Natural Computing Series

Kohei Nakajima
Ingo Fischer *Editors*

Reservoir Computing

Theory, Physical Implementations, and Applications

Springer

Natural Computing Series

More information about this series at http://www.springer.com/series/4190

Kohei Nakajima · Ingo Fischer
Editors

Reservoir Computing

Theory, Physical Implementations, and Applications

Springer

Editors
Kohei Nakajima
University of Tokyo
Tokyo, Japan

Ingo Fischer
Consejo Superior de Investigaciones
Científicas, IFISC (UIB-CSIC)
Palma, Spain

ISSN 1619-7127
Natural Computing Series
ISBN 978-981-13-1686-9 ISBN 978-981-13-1687-6 (eBook)
https://doi.org/10.1007/978-981-13-1687-6

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Foreword

Reservoir computing seems simple but is difficult, feels new but is old, opens horizons, and is brutally limiting. I will do my best in this foreword to leave the reader with many questions—to be answered, or maybe not, in the many chapters of this richly filled book.

The basic principle of reservoir computing (RC) is simple. Given: a *training* input signal $\mathbf{u}^{train}(t)$ paired with a desired target output signal $\mathbf{y}^{train}(t)$. Wanted: a filter (transducer) $\mathcal{F}$ which, when fed with input $\mathbf{u}^{train}(t)$, generates an output signal $\hat{\mathbf{y}}^{train}(t)$ which comes close to the target $\mathbf{y}^{train}(t)$. Approach: *Step 1.* Prepare a high-dimensional dynamical system $X(t)$, the *reservoir*, which can be driven by input $\mathbf{u}^{train}(t)$ and in which many state variables $x_i(t)$ (where $i = 1, \ldots, N$) can be observed and recorded. *Step 2.* Drive this system with input $\mathbf{u}^{train}(t)$ and record the corresponding reservoir-internal response signals $x_i^{train}(t)$. *Step 3.* Find (train, learn, and estimate) a *readout function* F which maps every recorded state vector $(x_1^{train}(t), \ldots, x_N^{train}(t))$ to an output $\hat{\mathbf{y}}^{train}(t)$ which approximates the training targets $\mathbf{y}^{train}(t)$. Finding such a readout F often boils down to a simple linear regression. Exploitation: feed new input signals $\mathbf{u}(t)$ to the reservoir, observe the reservoir-internal state vectors $\mathbf{x}(t) = (x_1(t), \ldots, x_N(t))$, and compute the output signal $\hat{\mathbf{y}}(t) = F(\mathbf{x}(t))$.

This basic scheme is very versatile. One can solve temporal input-output tasks for time series prediction, dynamical pattern generation, classification and segmentation, control, de-noising and channel equalization, rare event monitoring, and many more. One can apply RC to obtain practical engineering solutions in signal processing and control, robotics, communication technologies, machine learning, and AI; one can call upon RC models as an explanatory principle in theoretical neuroscience; and in mathematics, one can use RC as an entry point to identify and analyze a number of interesting phenomena in high-dimensional dynamical systems. But most importantly, one can in principle use *any* kind of nonlinear, high-dimensional dynamical system for the reservoir $X(t)$, regardless of whether it is an experimental probe of a freshly engineered nanomaterial, a quantum dot preparation, a replica of an octopus arm made from soft plastic and suspended in water, or a digital simulation of a neural

network all to be found in the scintillating collection of reservoirs that the reader will find in this book.

But... the closer one becomes involved with RC, the more difficult it gets, and if the one to embrace it in full contact, it gets almost impossibly difficult. Reservoirs are high-dimensional, input-driven, nonlinear, and often stochastic dynamical systems. A full theory of reservoir dynamics would be a full theory of everything that evolves in time. Only fragmentary insights into the unbounded phenomenal richness in general dynamical systems are currently available in mathematics, theoretical physics and biology, or the general complex systems sciences. Compared to what we *could* know about, observe in, and utilize from reservoir dynamics, we currently *do* know, see, and use almost nothing. It is easy to program a recurrent neural network with 100 neurons on a digital computer, declare it a reservoir, apply the basic RC scheme on a simple modeling task, exclaim "it works!" and call it good. This is how students worldwide get hooked on RC. However, when one gets pushed out of the comfort zone of the dozen or so ever-repeated "benchmark" tasks that pervade the RC literature, then reservoirs turn into feral beasts that take an enormous amount of patience and experience to tame. This applies, e.g., when the data are noisy or incomplete, have outliers or are nonstationary, have a wandering baseline or variable amplitude, are high-dimensional, or have multiple spatial or temporal scales, when stability conditions have to be guaranteed, the task demands continual online learning, or when the input data consist of rare events spiking out of a zero baseline. Moreover, problems arise, when there are many possible options for input and output signal re-coding (there always are), when one's computer allows only fast experimentation with small reservoirs, but one wants to extrapolate to large ones, or when one wants to automate the readout training. The promise of RC, one *need not* train the reservoir, turns into a problem: one *cannot* train the reservoir. There is an unlimited variability in task specifics, and there is an infinity of dynamical behaviors in candidate reservoir systems. In a haystack of possibilities one must find a reservoir whose native dynamics matches the demands of the task at hand. After two decades of RC research, we only have the faintest inklings of how to match reservoir dynamics with task dynamics. Many of my students choose a reservoir computing theme for their graduating thesis. I dare say that, when after much trial and error, they ultimately arrive at the point where "it works!" they don't understand *why* it works—and neither do I.

Many contemporary RC papers that I read or review introduce their subject still with "Reservoir computing is a new approach to train neural networks ...". Well... RC may be called "new" compared to Newton's and Leibniz's calculus, but by the standards of the fast-paced innovation cycles in machine learning it is rather old. The basic RC principle has been discovered and re-discovered many times, and I continue to become aware of earlier and earlier "first" sightings. This is how it goes with most ideas that are elementary and useful.

It is not customary to cite references in a foreword, but I take this as an opportunity to give due credit to RC pathfinders. The earliest perception of the RC principle that I am aware is Kirby and Day (1990), a 1-page conference abstract that was subsequently worked out by Kevin Kirby in a paper where he gives what I consider the

first concise and comprehensive account of the RC principle (Kirby 1991), with the readout from the reservoir (which he called *context reverberation subsystem*) trained by the perceptron learning algorithm. The problem of finding a "good" reservoir is clearly identified, and a sentence in the Conclusion section reads like prophesy: "This may encourage molecular electronic hardware implementations." Both papers remained entirely unnoticed (the single Google Scholar cite that I saw when I queried this in 2017 was a self-citation). In the same year 1991, Lambert Schomaker, in Chapter 7 of his Ph.D. thesis (Schomaker 1991) (separately published in Schomaker (1992)), described how a target output signal can be obtained by learning a linear readout from a random ensemble of spiking neural oscillators. I got to know about this work by an unlikely chance: after I was appointed at the University of Groningen in 2019, Lambert became my direct senior manager and he told me about his Ph.D. thesis in a casual conversation. I wonder how many other casual conversations with other senior colleagues worldwide would bring up similar surprises. Both Schomaker and Kirby refer back to earlier precursor ideas in their texts—clues for further studies in scientific archeology. The next independent discovery of RC that I know about occurred in cognitive neuroscience. Peter F. Dominey described a multi-module (human) brain circuit for sequence generation which included a simplified model of prefrontal cortex as a reservoir from which trainable readouts send information to the caudate nucleus (Dominey 1995). Up to the present day, and in close collaboration with other RC researchers, Dominey has been continuing to work out elaborate neuro-cognitive architectures with an RC core both for neuroscience modeling and for robotic/human-maching interaction applications. His chapter in this book gives a summary of a 25-year-long personal research mission.

The current RC literature mostly localizes the origin of RC in the propositions of *liquid state machines* by Wolfgang Maass and my *echo state networks* (Maass et al. 2002; Jaeger 2001). Wolfgang and I got to know of each other at the 2001 EU Advanced Course in Computational Neuroscience at the International Center for Theoretical Physics in Trieste, Italy, August 2001, where to our mutual surprise we found our own ideas almost identically reflected in the respective other's. We started to collaborate, soon joined by Benjamin Schrauwen who coined the term "reservoir computing" (or was it his brilliant Ph.D. student David Verstraeten? the first published paper where this term was used seems to be Verstraeten et al. (2005)). Benjamin rapidly built up an enormously productive RC research group at the University of Gent even before he was awarded his Ph.D. degree. I think several factors came together why reservoir computing took off only then. First, the three of us teamed up instead of defending proprietary RC islands. Second, for the first time the mathematical preconditions that make reservoirs functional were clearly spelled out through the *fading memory* and *separation property* in Wolfgang's models and the *echo state property* and an analysis of a reservoir's *memory capacity* in my work. Mathematical formulae gave authority to a wild-looking idea. Third, "it worked" really well in many demo tasks that met the taste and demands of the time—while training recurrent neural networks with other then-existing learning algorithms was difficult, unstable, and slow. The deep learning revolution superseded RC only toward the end of the 2010s when the intricacies of gradient-descent training of recurrent neural networks

finally became mastered. Reservoir computing research receded into a niche for a few years.

But RC research re-awakened and sprouted out again from this niche when RC principles were adopted first in the field of optical computing (see, e.g., chapters by Kanno et al. and Dambre et al.) and swiftly also in other domains of *physical reservoir computing* (surveyed in the chapter by Dale et al.). Most chapters in this book are a testimonial to the refreshing new thrust that RC has given to the wider fields of *unconventional/in-materio/natural/...* computing (I have a private list of about 15 different namings that have been branded in the last four decades or so). Materials scientists and non-digital device engineers from the most diverse makings continue to discover RC for themselves. As long as RC continues to be freshly discovered by colleagues in widening circles, there is still truth in when they say, in the introductory passages of articles, that RC is "... a new approach ...".

Ah, before I forget: there is one little technical thing that I want to point out to everyone who uses RC for the first time. So many neural-network-based RC papers state in their methods section that the spectral radius (largest absolute eigenvalue) of the network weight matrix should be less than unity to ensure the echo state property, a necessary condition to make RC work. This is a myth. A spectral radius $SR < 1$ is neither sufficient nor necessary for the echo state property (Yildiz et al. 2012), and a value much larger than 1 often gives the best performance. Please don't perpetuate this myth in your work! And while I am at it: another myth is that reservoirs work best when they are tuned to operate "at the edge of chaos", or "close to criticality". First, it's a misnomer, because the edge in question here is the edge of the echo state property, not the edge of chaos. If a reservoir slides across this edge, it doesn't necessarily (even not typically) enter a chaotic regime. Second, reservoirs are input-driven systems, and mathematicians still haven't entirely agreed on how to define chaos in input-driven systems. Finally, reservoirs "close to criticality" work well only for a certain class of learning tasks—the sort of tasks which are invariably re-iterated in articles on this subject—but reservoirs far on the stable side of that edge work much better for many other tasks. Cramer et al. (2020) point a spotlight on this affair. I really can't understand why this myth remains recited so often, given the massive counter-evidence from so many practical applications where carefully optimized reservoirs come out sitting safe and far away from this edge.

RC has the elegance of simplicity, which may be explanation enough why it inspires researchers in many fields. Of course, there are more substantial reasons why RC keeps blossoming, for instance, because it connects the neuro- with the computing sciences in stronger than purely metaphorical ways; or that it opens new doors for theoretical analyses of high-dimensional dynamical systems; or that materials scientists today really don't have many alternatives to make their unconventional substrates "compute".

But one should be aware that the powers of RC as a stand-alone carrier of "learning" or "computing" are decisively limited. Biological brains may be using RC in some places and some ways—it is unlikely that they don't because evolution will find and keep any trick that works—but brains use many other dynamical mechanisms and structuring principles and information encoding procedures as well, and I

don't think we have an idea yet even of *how* many. From my perspective of machine learning, AI and theory of computing, the strongest weakness (what a nice oxymoron) of pure RC is its inherent blindness to hierarchical multi-scale compositionality of data structures, processes. and architectures. Here, I understand compositionality in a strong sense which includes *bidirectional* interaction between higher modules or layers and their sub-modules or lower layers. An example are planning architectures for autonomous agents where higher planning modules generate longer-term plans that "call" lower sub-plan modules in a *top-down* direction, and stay informed about execution progress in a *bottom-up* direction of communication from the sub-modules. Another example are the Boltzmann machine or Friston's free-energy models of neural processing where higher layers send statistical biases to lower layers, and are informed about conditional feature distributions from below. In computer science, the object-oriented programming paradigm is the very manifestation of bidirectionally effective compositionality. The top-down actions can be interpreted in a variety of ways, for instance, as attention control, predictive context settings, or read/write signals in working memory systems. Such top-down modulations are essential for full-fledged cognitive information processing, but such bidirectionally effective hierarchical cognitive architectures cannot be realized by RC alone. Additional structures and algorithms are needed to coordinate intermodule communication (as in my attempt in Jaeger (2007) to design a hierarchical RC learning architecture that can discover temporal features on several timescales), or additional teacher signals for the individual modules must be created (as in Pascanu and Jaeger (2011) where we trained a kind of parser for visual text input that had a nested grammatical structure), or additional control mechanisms must be installed to modulate the reservoir dynamics "from above" (like *conceptors* (Jaeger 2017)). It is one of the strongest strengths of today's deep learning networks that such multi-directionally organized architectures, for instance, *neural Turing machines*, can be trained by "end-to-end" gradient descent, where the requisite local training signals are automatically generated. This said, I emphasize that RC can positively be applied with much benefit in certain multi-scale learning tasks using uni-directionally coupled stacks of reservoirs where lower, typically faster reservoirs connect upwards to higher, typically slower reservoirs. Gallicchio and Micheli (in this volume) survey such architectures, which they call *deep RC* systems.

RC research keeps advancing in many directions. I want to conclude with my personal favorite challenges for the next evolutionary steps in RC research:

- Coordinate RC modules in complex learning and information processing systems with the aid of additional mechanisms—this theme is also highlighted in the Conclusion of Dambre et al.'s chapter in this book.
- In physical RC, find ways to realize the readout and its training directly in the non-digital material substrate, instead of delegating it to a digital host computer.
- Leverage the infinite dimensionality of spatially extended nonlinear excitable media, extending the readout combination of a finite number of reservoir signals to an infinite-dimensional integration, convolution, or field transformation. In physical RC, one might envision spatially continuous two-layer substrates where

the bottom layer acts as a reservoir and the top layer would function as a continuous version of what today are the readout weight matrices.
- Find effective ways to cope with the unpleasant properties of physical reservoirs, such as device mismatch, parameter drift, temperature sensitivity, low numerical precision and stochasticity, and partial observability. Physical reservoirs may only become practically useful when appropriate auto-calibration or homeostatic regulatory mechanisms are realized in combination with numerically robust and swiftly self-adapting readout processes.
- Rigorously analyze which abstract dynamical characteristics of input and output data and task specifications should be reflected in which characteristics of reservoir dynamics. Currently available insights are mostly distilled from experimental studies of timescale profiles or frequency spectra in input data and provide no comprehensive guides for optimizing reservoir designs.

Many chapters in this collection include historical summaries of major RC research strands, and all tell enticing stories about what today's achievements are and are not. This book lets us see where we stand and invites us to imagine where we can go further. Being a veteran of the field, I feel enormously grateful for the massive labor of editors and authors to plant this landmark after 30 years of a voyage that will continue to feel fresh and young.

Herbert Jaeger
Bernoulli Institute for Mathematics
Computer Science and Artificial Intelligence
Cognitive Systems and Materials Center
(CogniGron)
University of Groningen
Groningen, The Netherlands

References

B. Cramer, D. Stöckel, M. Kreft, M. Wibral, J. Schemmel, K. Meier, V. Priesemann. Control of criticality and computation in spiking neuromorphic networks with plasticity. Nat. Commun. **11**, 2853, 2020

D. Verstraeten, B. Schrauwen, D. Stroobandt. Reservoir computing with stochastic bitstream neurons. In *Proceedings of the 16th Annual Prorisc Workshop*, pp. 454–459, 2005

H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks. GMD Report 148, GMD - German Nat. Res. Instit. Comp. Sci. 2001. https://www.ai.rug.nl/minds/uploads/EchoStatesTechRepErratum.pdf.

H. Jaeger. Discovering multiscale dynamical features with hierarchical echo state networks. Technical report 10, School of Engineering and Science, Jacobs University, 2007. URL https://www.ai.rug.nl/minds/uploads/hierarchicalesn_techrep10.pdf.

H. Jaeger, Using conceptors to manage neural long-term memories for temporal patterns. *JMRL*, **18**, 1–43, 2017

I. B. Yildiz, H. Jaeger, S. J. Kiebel, Re-visiting the echo state property. Neural Netw. **35**, 1–20, 2012

K. Kirby, Context dynamics in neural sequential learning. In *Proceedings Florida AI Research Symposium (FLAIRS)*, pp. 66–70, 1991

K. G. Kirby, N. Day, The neurodynamics of context reverberation learning. In *Proceedings Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Vol. 12 4*, pp. 1781–1782, 1990

L. R. B. Schomaker, *Simulation and Recognition of Handwriting Movements: A vertical approach to modeling human motor behavior*. Phd thesis, Nijmeegs Instituut voor Cognitie-onderzoek en Informatietechnologie, Nijmegen, 1991. URL https://repository.ubn.ru.nl/handle/2066/113914

L. R. B. Schomaker, A neural oscillator-network model of temporal pattern generation. Hum. Mov. Sci. **11**, 181–192, 1992

P. F. Dominey, Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. Biol. Cybern. **73**, 265–274, 1995

R. Pascanu, H. Jaeger. A neurodynamical model for working memory. Neural Netw. **240**(2), 199–207, 2011

W. Maass, T. Natschläger, H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Comput.**140**(11), 2531–2560, 2002

Preface

Reservoir Computing: Theory, Physical Implementations, and Applications is the first comprehensive book about reservoir computing (RC). RC was introduced in the early 2000s as a unified framework for recurrent neural network (RNN) training; it included a number of seminal models, such as echo-state networks (Jaeger 2001) and liquid state machines (Maass et al. 2002). Although RC originated in computational neuroscience and machine learning, in recent years, the use of RC has spread, and it has been introduced into a wide variety of fields, including nonlinear dynamical systems theory, physics, material science, biological science, and robotics (Fig. 1). One of the major reasons for this increase in relevance of RC is its conceptual simplicity. RC capitalizes on the nonlinear responses of a high-dimensional dynamical system, referred to as the reservoir. By restricting the learning process to the readout layer, RC resolved the difficulties and instabilities of RNN training, which had conventionally been implemented using the method of gradient descent. Using

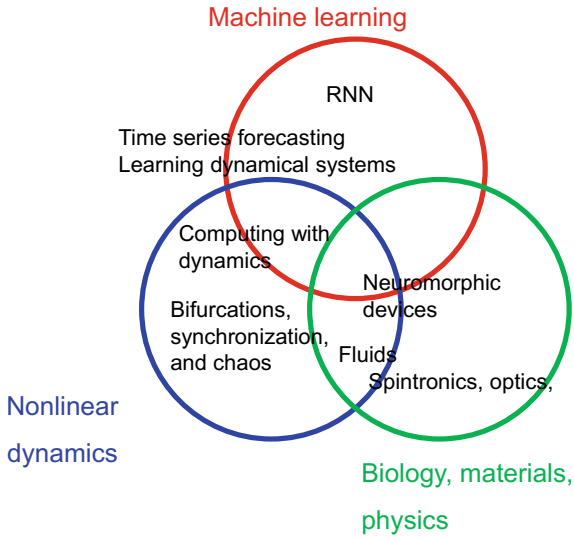


Fig. 1 Diagram summarizing how each field is connected through the concept of RC

gradient descent, all the network weights were trained, according to performance optimization needs and available training time. The particular RC concept allows us to exploit many "well-behaving" dynamical systems as reservoirs. This finding has resulted in important opportunities exploiting not only standard RNN but also many nonlinear dynamical systems and various physical dynamics found in nature as a computational resource.

Accordingly, recent RC trends and technologies exhibit two important directions. The first direction is to extend the framework of RC from the conventional RNN to a more abstract setup using the terms of nonlinear dynamical systems. With this broadened perspective, RC is not specific to the field of machine learning anymore but can be connected to a much wider class of systems. In particular, the connection and relationship between many technical terms developed in different fields and originating from different contexts have been revealed and bridged, which makes the RC technique accessible to various disciplines. For instance, the echo state property, which was originally proposed in the context of the echo-state networks, can also be related to generalized synchronization between the input stream and the reservoir dynamics. These bridges prove also effective and vital to the following second direction.

The second direction is the exploitation of physical dynamical systems as reservoirs; the framework for doing so is called physical reservoir computing (PRC). Because of the rapid development of computational technologies and sensing systems worldwide, novel schemes and devices are required to process massive amounts of data quickly in real time. In conventional computational architectures, due to the separation of the processing system and the memory system, there is a limit to the information processing speed, which is called a von Neumann bottleneck. This limit could be overcome using an approach inspired by biology or by using a dynamical-system-based implementation that realizes information processing and carries a memory of past input streams simultaneously; this is a typical non-von Neumann architecture. PRC is one of the main candidates for such architectures that researchers are currently focusing on. Many physical systems and materials have been already suggested and implemented as reservoir computing substrates. These systems include a wide range of physical systems exhibiting different spatiotemporal scales ranging from mechanical systems to optics, nanomaterials, spintronics, and quantum many-body systems. They are expected to be the substrates for next-generation neuromorphic devices that can process information natively at the edge according to the spatiotemporal scale, which is often termed edge computing. The variety of physical substrates provides a large diversity in the type of information processing that can be implemented. It is noticeable that this inspiration of PRC is, in fact, not a recent invention but has been around for a while since the genesis of RC approaches. Original attempts to implement PRC can be found in the ideas of the liquid computer (Natschläger et al. 2002) and the liquid brain (Fernando et al. 2003).

This book presents recent developments in the area of RC and is sub-structured into two major parts: theory and physical implementations. The book is a compilation of chapters contributed by different authors, who are leading experts in their respective fields. In detail, the book is structured as follows:

The first part (Part I) is devoted to theoretical aspects of RC. It starts with a wide perspective of aspects on how the real human brain processes information. In W. Singer's chapter, by comparing the recent system architecture of artificial intelligence and the real brain comprehensively, the important role of nonlinear dynamics in the cerebral cortex is discussed. In P. F. Dominey's chapter, it is argued that these dynamics generated in the cerebral cortex with structures of recurrency actually act as a reservoir. Subsequently, based on these properties of the real brain, A. Subramoney, F. Scherr, and W. Maass propose a novel architecture that can include meta-learning, called learning-to-learn, into the reservoir using plastic connections of weights. Deep architectures have also been introduced in the RC framework, and C. Gallicchio and A. Micheli provide a comprehensive overview of recent developments of deep reservoir computing. In the chapter by M. Inubushi, K. Yoshimura, Y. Ikeda, and Y. Nagasawa, the role of common-signal-induced synchronization on the information processing capability of the reservoir is discussed as a key to guaranteeing reproducible input-output relations. Finally, in the chapter by J. S. Pathak and E. Ott, recent progress on time series forecasting of large-scale spatiotemporal chaos introducing parallel spatial coupling in RC is presented, and the performance improvement is discussed in detail.

The second part (from Part II to Part VII) focuses on the physical implementations of RC, namely, PRC. M. Dale, J. F. Miller, S. Stepney, and M. Trefzer initiate the discussion on how to classify the appropriate physical substrates for computation in generic settings and propose a scheme and measures to systematically evaluate them. PRC is then introduced in the context of mechanical systems (Part III). H. Hauser has reviewed several case studies of PRC applications in robotics and discusses the importance of embodiment and the effectiveness of the approach for soft robotics. Guillaume Dion, Anouar Idrissi-El Oudrhiri, Bruno Barazani, Albert Tessier-Poirier, and Julien Sylvestre present PRC using MEMS.

Part IV begins by focusing on neuromorphic devices. F. Hadaeghi provides a systematic survey of neuromorphic electronic systems and their applications to RC and summarizes future challenges. In the chapter by S. Apostel, N. D. Haynes, E. Schöll, O. D'Huys, and D. J. Gauthier, field-programmable gate array implementations of an autonomous Boolean network for RC are demonstrated and analyzed in detail. R. Aguilera, H. O. Sillin, A. Z. Stieg, and J. K. Gimzewski present an atomic switch network as a substrate of RC implementations. The subsequent three chapters focus on spintronics approaches (Part V). M. Riou, J. Torrejon, and F. Abreu, et. al. present the recent development of neuromorphic applications for nanoscale spin-torque oscillators. T. Taniguchi, S. Tsunegi, and S. Miwa, et al. analyze the information processing capability (e.g., memory capacity) of a spin-torque oscillator as a reservoir, and H. Nomura, H. Kubota, and Y. Suzuki demonstrate an approach that uses a simple magnetic nano-dots array and discuss the possibility of implementing a larger scale array as a reservoir.

Part VI concentrates on photonic reservoir computing, which exploits optical systems as reservoirs. K. Kanno and A. Uchida demonstrate that the computational performance of a photonic reservoir can be improved by introducing a chaotic input mask signal, and they present an implementation of a miniature size photonic

integrated circuit. J. Dambre, A. Katumba, C. Ma, S. Sackesyn, F. Laporte, M. Freiberger, and P. Bienstman provide a brief history of integrated photonic reservoirs and introduce recent approaches designed to increase the computational power of the system.

Part VII is devoted to PRC developments in the field of quantum machine learning. In recent years, remarkable progress has been made in quantum computation and the development of quantum computer technology is heating up worldwide. Simultaneously, noisy intermediate-scale quantum devices, which include a number of qubits with no error correction capability and their applications are receiving attention from many physicists. Part VII begins with the chapter by K. Fujii and K. Nakajima that introduces a framework for quantum reservoir computing (QRC) from the basics. This chapter also introduces several approaches, such as quantum extreme learning machine (QELM) and quantum circuit learning, and demonstrates emulation tasks of chaotic attractors based on QRC. The chapter by M. Negoro, K. Mitarai, K. Nakajima, and K. Fujii presents the first implementation of a quantum reservoir using nuclear magnetic resonance (NMR) ensemble systems and successfully demonstrates QELM. The authors also discuss a future scenario for implementing QRC using NMR ensemble systems.

Last but not the least, we note that the year 2020, in which this book was prepared, was a difficult and challenging year for mankind in general and for the people involved in this book in particular. Many problems arose in the face of COVID-19, and the processes involved in creating this book were significantly delayed. During this difficult time, all the chapter authors, the Springer editor, the collaborators, and our families have been incredibly supportive and patient. We would like to sincerely thank them all and acknowledge their role in making this publication possible. It is our greatest pleasure to bring out this exciting book into the world.

Tokyo, Japan — Kohei Nakajima
Palma, Spain — Ingo Fischer
November 2020

References

H. Jaeger. The “echo state” approach to analysing and training recurrent neural networks-with an erratum note. Bonn, Germany: German Nat. Res. Cent. Inf. Technol. GMD Tech. Rep. **148**(34), 13 (2001)

W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Comput. **14**(11), 2531–2560 (2002)

T. Natschläger, W. Maass, H. Markram, The “liquid computer”: A novel strategy for real-time computing on time series. Spec. issue Found. Inf. Process. TELEMATIK, 8(ARTICLE), 39–43 (2002)

C. Fernando, S. Sojakka, Pattern Recognition in a Bucket. In: Banzhaf W., Ziegler J., Christaller T., Dittrich P., Kim J.T. (eds) Advances in Artificial Life. ECAL 2003. Lecture Notes in Computer Science, vol 2801. Springer, Berlin, Heidelberg (2003)

Contents

Part I
Fundamental Aspects and New Developments in Reservoir Computing

The Cerebral Cortex: A Delay-Coupled Recurrent Oscillator Network?

Wolf Singer

Abstract The refinement of machine learning strategies and deep convolutional networks led to the development of artificial systems whose functions resemble those of natural brains, suggesting that the two systems share the same computational principles. In this chapter, evidence is reviewed which indicates that the computational operations of natural systems differ in some important aspects from those implemented in artificial systems. Natural processing architectures are characterized by recurrence and therefore exhibit high-dimensional, non-linear dynamics. Moreover, they use learning mechanisms that support self-organization. It is proposed that these properties allow for computations that are notoriously difficult to realize in artificial systems. Experimental evidence on the organization and function of the cerebral cortex is reviewed that supports this proposal.

1 Introduction

The objects of the world, animate and inanimate, are composed of a relatively small repertoire of elementary components. Their virtually infinite diversity results from the variability of the relations among the components. Thus, an elegant and economical strategy for the description and classification of objects is to establish combinatory codes, to create symbols for the representation of the components and a code for the relations. The alphabet of 28 symbols suffices to compose world literature. It is for this reason that evolution has optimized cognitive systems to exploit the power of combinatorial codes, to represent elementary features of input patterns, to evaluate and encode the relations between these features and to generate minimally overlapping representations of particular feature constellations for classification. It is for the

W. Singer (✉)
Max Planck Institute for Brain Research (MPI), Frankfurt am Main, Germany
e-mail: wolf.singer@brain.mpg.de

Ernst Strüngmann Institute for Neuroscience (ESI) in Cooperation with Max Planck Society, Frankfurt am Main, Germany

Frankfurt Institute for Advanced Studies (FIAS), Frankfurt am Main, Germany

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_1

same reason that the design of artificial cognitive systems shares numerous structural and functional similarities with natural brains (Hochreiter and Schmidhuber 1997; Kar et al. 2018; Silver et al. 2017, 2018).

In many respects, evolution and man-made artefacts converged to similar solutions for the evaluation of relations and the representation of relational constructs. Both systems fulfill these functions with serial operations within hierarchically organized multilayer networks, whose nodes serve as integrators and are coupled through diverging and converging connections with adjustable gain.

However, there are also substantial differences. These concern the integrative functions of the nodes, the learning mechanisms required to obtain the desired input-output functions and above all the connectivity. In natural systems, feed-forward connections are complemented by massive feed-back (top-down projections) between layers and extremely dense reciprocal connections between nodes of the same layer. These re-entry connections are missing in most feed-forward artificial systems but in natural brains, they are more abundant than feed-forward connections (Markov et al. 2014; Bastos et al. 2015). This connectivity gives rise to very complex, high-dimensional, non-stationary dynamics, and there is evidence that these dynamic properties are exploited for computational strategies. In artificial systems implementation of these biological principles of computation is still at the very beginning.

2 Strategies for the Evaluation and Encoding of Relations

One strategy for the analysis and encoding of relations is based on convergent feed-forward circuits. This strategy is ubiquitous in natural systems. Nodes (neurons) of the input layer are tuned to respond to particular features of input patterns and their output connections are made to converge on nodes of the next higher layer. By adjusting the gain of these converging connections and the threshold of the target node, it is assured that the latter responds preferentially to only a particular conjunction of features in the input pattern (Hubel and Wiesel 1968; Barlow 1972). In this way, consistent relations among features become represented by the activity of conjunction-specific nodes. By iterating this strategy across multiple layers in hierarchically structured feed-forward architectures, complex relational constructs (cognitive objects) can be represented by conjunction-specific nodes of higher order. This basic strategy for the encoding of relations has been realized independently several times during evolution in the nervous systems of different phyla (molluscs, insects, vertebrates) and reached the highest degree of sophistication in the hierarchical arrangement of processing levels in the cerebral cortex of mammals (Felleman and van Essen 1991; Glasser et al. 2016; Gross et al. 1972; Tsao et al. 2006; Hirabayashi et al. 2013; Quian Quiroga et al. 2005). This strategy is also the hallmark of the numerous versions of artificial neuronal networks designed for the recognition and classification of patterns (Rosenblatt 1958; Hopfield 1987; DiCarlo and Cox 2007; LeCun et al. 2015). The highly successful recent developments in the field of artificial “deep

learning networks" (LeCun et al. 2015; Silver et al. 2017, 2018) capitalizes on the scaling of this principle in large multilayer architectures.

In natural systems a second, complementary strategy to encode relations consists of the temporary formation of cooperating assemblies of nodes responding collectively to particular constellations of related features. This strategy requires recurrent networks, relies on non-linear self-organizing dynamics, is realized in sensory systems of diverse species, including invertebrates (Laurent 1996; Rabinovich et al. 2001) and is particularly evolved in structures of vertebrate brains such as the hippocampus and the cerebral cortex. As proposed by Donald Hebb (1949), consistent relations among features can be encoded by forming distributed but functionally coherent assemblies in which individual, feature coding neurons cooperate. Neurons coding for features that typically co-occur and thereby define in their specific constellation particular regularities of natural objects get bound together into an assembly that, as a whole, represents the respective relations among particular features: At low levels of processing, these would correspond to particular feature constellations such as are captured by the Gestalt—principles for perceptual grouping. At higher levels of the processing hierarchy, such dynamic assemblies are thought to represent a concrete perceptual object or a more abstract cognitive object such as a category, a concept or an action plan. In this case, the *binding* of specific features is not achieved by convergence of feed-forward connections onto conjunction-specific neurons but by reciprocal connections between feature-selective nodes. These connections are endowed with correlation-dependent synaptic plasticity mechanisms (Hebbian synapses; see below) and strengthen when the interconnected nodes are frequently co-activated. Thus, nodes that are often co-activated because the features to which they are tuned frequently co-occur enhance their mutual interactions. As a result of these cooperative interactions, the vigour and/or coherence of the responses of the respective nodes is enhanced when they are activated by the respective feature constellation. In this way, consistent relations among the components of cognitive objects are translated into the weight distributions of the reciprocal connections between network nodes and represented by the joint responses of a cooperating assembly of neurons. Accordingly, the information about the presence of a particular constellation of features is not represented by the activity of a single conjunction-specific neuron but by the amplified or more coherent or reverberating responses of a distributed assembly of neurons.

Both relation-encoding strategies have advantages and disadvantages, and evolution has apparently opted for a combination of the two. Feed-forward architectures are well suited to evaluate relations between simultaneously present features, and they raise no stability problems because they lack reverberatory runaway dynamics. At first sight, one might also expect that they allow for fast processing because they rely exclusively on a series of simple summation and thresholding operations. However, discharge rates of cortical neurons are low and can carry only little information when integrated over short intervals. This problem could in principle be solved by configuring nodes as clusters of cells and to average across their activity (population coding). However, this strategy is costly in terms of hardware and energy. In addition, it is hampered by the fact that noise fluctuations of cortical activity are

correlated which limits the usefulness of averaging strategies (Averbeck et al. 2006). As will be discussed in later paragraphs, another possibility to reduce integration times is to raise the salience of neuronal responses by enhancing the synchronicity of neuronal discharges rather than their rate (Abeles 1991; Van Rullen et al. 2001, 2005). Another disadvantage of networks consisting exclusively of feed-forward connections is that they are less apt to handle relations among temporally segregated events because they lack short-term memory functions. Moreover, they are costly in terms of hardware requirements. Because specific constellations of features have to be represented explicitly by conjunction-specific neurons via the convergence of the respective feed-forward connections and because the dynamic range of the nodes is limited, a large number of nodes and processing levels are required to cope with the combinatorial complexity of possible feature constellations characterizing real-world objects (combinatorial explosion). Consequently, biological systems relying exclusively on feed-forward architectures can afford representation of only a limited number of behaviourally relevant relational constructs.

By contrast, assemblies of recurrently coupled, mutually interacting nodes (neurons) can cope very well with the encoding of temporal relations (sequences) because such networks exhibit fading memory due to reverberation and therefore can integrate temporally segregated information. Assembly codes are also much less costly in terms of hardware requirements, because individual feature-specific nodes can be recombined in flexible combinations into a very large number of different assemblies, each representing a different cognitive content (combinatorial code). In addition, coding space is dramatically widened because information about the statistical contingencies of features can be encoded not only in the synaptic weights of feed-forward connections but also in the weights of the recurrent and feed-back connections. Finally, the encoding of entirely new or the completion of incomplete relational constructs (associativity) is facilitated by the cooperativity inherent in recurrently coupled networks that allows for pattern completion and the generation of novel associations (generative creativity).

However, these advantages of assembly coding have a price. The implementation of this combinatorial coding strategy requires sophisticated mechanisms to control the dynamics of the recurrent network because (i) fast formation of assemblies requires a delicately regulated level of resting activity, (ii) networks can fall dead if global excitation drops below a critical level and (iii) they can engage in runaway dynamics and become epileptic if a critical level of excitation is reached. Nature takes care of this problem with a number of self-regulating mechanisms involving normalization of synaptic strength (Turrigiano and Nelson 2004), inhibitory interactions (E/I balance) (Yizhar et al. 2011) and control of global excitability by modulatory systems, which keep the network within a narrow working range just below criticality (Plenz and Thiagarajan 2007; Hahn et al. 2010).

Another and particularly challenging problem, known as the *superposition catastrophe*, is the segregation of simultaneously active assemblies. This problem arises whenever more than one object is present and when these objects and the relations among them need to be encoded within the same network layer. If assemblies were solely distinguished by enhanced activity of the constituting neurons, as proposed

by Hebb (1949), it becomes difficult to distinguish which of the more active neurons actually belong to which assembly, in particular, if objects share some common features and overlap in space. In this case, the corresponding feature-selective nodes would have to be shared by several assemblies (the "binding problem"). It has been proposed that this problem can be solved by multiplexing, i.e. by segregating the various assemblies in time (Milner 1992; von der Malsburg and Buhmann 1992; for reviews, see Singer and Gray 1995; Singer 1999). However, if assemblies were distinguished solely by enhanced discharge rate, this option is also problematic because readout of enhanced discharge rate requires temporal integration in down-stream structures. Given the low discharge rate of cortical neurons (see above), it might take several hundreds of milliseconds before the more active members of assemblies become distinguishable from less active neurons and hence multiplexing is achievable only on a slow time scale. This is incompatible with experimental evidence on processing speed (Van Rullen et al. 2005). Such slow multiplexing also jeopardizes the associative capacities of assembly coding as it becomes difficult to establish relations among simultaneously represented objects. It has been proposed, therefore, that the neurons temporarily bound into assemblies are distinguished not only by an increase of their discharge rate but also by the precise synchronization of their action potentials (Gray et al. 1989; Singer and Gray 1995; Singer 1999). Synchronization is as effective in enhancing the efficiency of neuronal responses in down-stream targets as is enhancing discharge rate (Bruno and Sakmann 2006). Thus, activation of target cells at the subsequent processing stage can be assured by increasing either the rate or the synchronicity of discharges in the converging input connections. The advantage of increasing salience by synchronization is that integration intervals for synchronous inputs are very short, allowing for instantaneous detection of enhanced salience. Hence, information about the relatedness of responses can be read out very rapidly. In extremis, single spikes can be labelled as salient and belonging to a particular assembly if synchronized with a precision in the millisecond range. Thus, assemblies defined by synchrony rather than rate increases can be multiplexed at a much faster rate than rate-coded assemblies without becoming confounded.

3 Learning Mechanisms

Artificial and natural systems differ not only with respect to their network architecture and the complexity of computations performed by the nodes but also with regard to the learning mechanisms. Natural learning mechanisms are exquisitely sensitive to temporal relations and exploit these for the establishment of associations. In contrast most artificial systems rely on some sort of supervised learning in which temporal relations play only a minor role if at all. In these systems, the gain of the feed-forward connections is iteratively adjusted until the activity patterns at the output layer represent particular input patterns with minimal overlap. To this end, very large samples of input patterns are generated, deviations of the output patterns from the desired result are monitored as "errors" and backpropagated through the network in order to

change the gain of those connections that contributed most to the error. In multilayer networks, this is an extremely challenging procedure and the breakthroughs of recent developments were due mainly to the design of efficient backpropagation algorithms.

These learning strategies differ substantially from those implemented in natural systems. The learning mechanisms in natural systems exploit the fundamental role of temporal relations for the definition of relatedness. Events exhibiting consistent temporal relations tend to be related. Simultaneously occurring events usually have a common cause or are interdependent because of interactions. If one event consistently precedes the other, the first is likely the cause of the latter and if there are no temporal correlations between the events, they are most likely unrelated. Accordingly, the molecular mechanisms developed by evolution for the establishment of associations are exquisitely sensitive to temporal relations between the activity patterns of interconnected nodes. The crucial variable that determines the occurrence and polarity of gain changes of the connections is the *temporal relation* between discharges in converging presynaptic inputs and/or between the discharges of presynaptic afferents and the activity of the postsynaptic neuron. In natural systems, most excitatory connections—feed-forward, feed-back and recurrent—as well as the connections between excitatory and inhibitory neurons are adaptive and can change their gain as a function of the correlation between pre- and postsynaptic activities. The molecular mechanisms that translate electrical activity in lasting changes of synaptic gain evaluate correlation patterns with a precision in the range of tens of milliseconds and support both strategies for the representation of relations: the experience-dependent generation of conjunction-specific neurons in feed-forward architectures and the formation of assemblies.

A large number of empirical studies have led to the formulation of rules that capture the relation between the polarity of synaptic modifications and the temporal patterning of activity of the connected neurons. These are addressed as the BCM (Bienenstock et al. 1982), ABS (Artola et al. 1990) and STDP (Markram et al. 1997; Bi and Poo 1998) rules. Interestingly, these rules apply both to the experience-dependent selection of circuits during development and to learning-dependent changes of synaptic gain in the adult.

In summary, synaptic gain increases for (reciprocal) connections among pairs of cells that are frequently activated in temporal contiguity or when one cell successfully drives the respective other or when converging inputs are active in synchrony. Conversely, synaptic gain decreases for connections among pairs of cells whose activity is un- or anti-correlated or for connections that discharge shortly after the target neuron has been driven by other inputs. Inputs also weaken when they are active while the target cell is inhibited or when they are silent, while the target cell is strongly activated by other inputs (heterosynaptic depression). Thus, use-dependent synaptic modifications are not only sensitive to the coherence of converging activity but also to causal relations. The gain of excitatory connections increases if their activity can be causally related to the activation of the postsynaptic neuron and weakens when this is not the case. Moreover, the strong dependence of synaptic modifications on cooperativity between pre- and postsynaptic activities assigns a particularly important role to synchronized states. Synchronous activity is expected to provide a particularly

favourable condition for the strengthening of synaptic connections. This prediction is supported by the finding that synaptic plasticity in the visual cortex is facilitated by entrainment of the recurrent network in synchronous gamma oscillations (Galuske et al. 2019).

These use-dependent synaptic modifications permit the nervous system to learn about the statistical contingencies of features and events in the external world and to generate internal models of relational constructs. The fundamental nature of these learning mechanisms is a likely reason for the striking conservation of the molecular mechanisms supporting activity-dependent modifications of synaptic transmission. However, these mechanisms can evaluate temporal relations only over intervals of a few hundred milliseconds at most. Therefore, additional mechanisms have been implemented to enable analysis of correlations over longer time spans such as are required for the detection and encoding of contingencies separated by long intervals. These involve memory functions at different time scales, ranging from the fading memory of recurrent networks over short- to long-term memory mechanisms in devoted structures of the brain.

4 Association of Signals Lacking Temporal Structure

All learning rules identified so far rely on the evaluation of temporal relations between the activity patterns of the interconnected nodes. Hence, the generation of both conjunction-specific neurons and assemblies requires that the activity patterns used for the selective strengthening of connections have temporal structure. This raises the question on how relations between events that lack temporal structure are analysed and how these events can become associated selectively by the established, time-sensitive learning mechanisms. The encoding of relations between external stimuli poses no problem as long as these stimuli have a temporal structure because sensory systems signal the temporal structure of stimuli with extreme precision (Buracas et al. 1998; Reinagel and Reid 2002). Simulation studies, partly based on the concept of synfire chains proposed by Moshe Abeles (1991), confirmed that conventional integrate-and-fire neurons are capable of transmitting temporal information with the required precision (Mainen and Sejnowski 1995; Diesmann et al. 1999).

Additional mechanisms are required, however, when selective associations have to be established between neuronal responses that lack precise temporal structure. Such is the case for sensory responses to stimuli that lack temporal structure or for activity that is generated internally in the context of memory recall or imagery. Evidence for the implementation of such mechanisms is indeed available. These impose a temporal structure on neuronal responses which allows the brain to utilize the existing mechanisms of coincidence detection and synaptic plasticity in order to analyse and store relations between signals that initially lack a temporal structure.

5 The Generation of Temporally Structured Activity

Neuronal mechanisms capable of generating temporally structured activity are diverse, abundant and evolutionarily ancient. A common and highly conserved strategy to generate temporally structured activity is the oscillatory patterning of activity, the basic principle of parsing time, used in virtually all clocks. Both, certain neurons, addressed as pacemaker or clock neurons, as well as most neuronal networks have a high propensity to engage in oscillatory activity (Gray and Singer 1989; Börgers and Kopell 2008; Whittington et al. 2000; Buzsáki and Wang 2012). These oscillations cover a broad frequency range, from below 0.1 to more than 200 Hz, and they tend to occur in typical frequency bands that are characteristic for particular brain structures and brain states. As reviewed recently (Buzsáki et al. 2013), these frequency bands are surprisingly well conserved across different species and even across different phyla, suggesting that they reflect some basic dynamics of nerve cells and/or circuits and are adapted to serve particular cognitive and/or executive functions. Common to all oscillatory processes is that neurons undergo periodic changes of excitability. Phases of increased excitability, often associated with action potential generation alter with phases of low excitability (Fries et al. 2007). In addition to endowing the discharge sequences of individual neurones (nodes) with a precise temporal structure, this oscillatory modulation is also ideally suited to introduce precise but variable temporal relations between the discharge patterns of interconnected network nodes. The reason is the propensity of oscillators to resonate and to be entrainable by periodically modulated inputs. As observed as early as 1665 by Van Huygens, a Dutch watch maker, very weak interactions suffice to synchronize coupled oscillators if their preferred frequencies are similar. If the difference between preferred frequencies increases, stronger coupling is required to assure synchrony with stable phase locking and if the frequency difference increases beyond a critical point, synchronization becomes unstable. Phase offset gradually increases, and this may lead to intermittent phase resetting or a complete breakdown of synchrony. These complex and highly non-linear relations have been analysed in numerous theoretical studies (Winfree 1967; Aronson et al. 1990; Kuramoto 1990) and are summarized in the so-called Arnold tongue regime (Glass and Sun 1994). A graphical representation of synchronization behaviour relating the difference in preferred frequency with increasing coupling strength leads to a “tongue”-shaped surface of possible synchronization regimes, the Arnold tongue.

In addition to coupling strength and preferred oscillation frequency, synchronization probability also depends critically on the conduction velocity of the coupling connections that varies over a wide range in neuronal networks. If conduction delays exceed a critical value, synchronization breaks down and interactions may lead to drifting phase behaviour or a complete shutdown of the oscillations of one or all of the coupled oscillators (Aronson et al. 1990; Reddy et al. 1998; Vicente et al. 2008; Niebur et al. 1991; for a review, see Pajevic et al. 2014). Thus, synchronization by reciprocal coupling can only be achieved over larger distances if coupling connections are fast conducting or if oscillation frequency is reduced.

6 Synchrony as a Common Signature of Relatedness

Such reciprocal coupling between oscillatory circuits is a common motif in recurrently coupled neuronal networks such as the cerebral cortex. Thus, the Arnold tongue formalism should be applicable to describe the relation between coupling strength and synchronization probability. Experimental evidence from the visual cortex indicates that this is indeed the case. Nodes tuned to features that have a high probability to co-occur in natural scenes are more strongly coupled than nodes tuned to features that are rarely contiguous (Pecka et al. 2014; Gilbert and Wiesel 1989; Stettler et al. 2002; Bosking et al. 1997). This selectivity of coupling is to a large extent due to experience-dependent synaptic plasticity (Singer and Tretter 1976; Löwel and Singer 1992; Smith et al. 2015) whereby the statistical contingencies of features in the outer world are translated into the synaptic weight distributions of the recurrent connections among the feature-selective nodes (Iacaruso et al. 2017). As demonstrated several decades ago, the consequence of anisotropic coupling strength is that nodes responding to features that often co-occur and hence are likely related, e.g. because they are part of a particular cognitive object, synchronize their oscillatory responses (Gray et al. 1989). Thus, these nodes become distinguishable of members of an assembly (see above) that signals the relatedness of the respective features. This observation has been at the origin of the "*binding by synchrony hypothesis*" which posits that responses to related features which should be bound according to common Gestalt—principles become synchronized (for a review, see Singer and Gray 1995; Singer 1999). Thus, precise temporal relations between neuronal discharges (phase synchrony or consistent phase shifts) appear to serve as a code of relatedness in a rather general sense. They are used by learning mechanisms to generate both conjunction-specific neurons and Hebbian assemblies, and they are used during actual processing of information to label related responses. To use temporal contiguity as a general code of relatedness is a parsimonious solution as it permits use of the same code of relatedness both for signal processing and learning.

In conclusion, the established time-sensitive mechanisms of synaptic plasticity permit translation of temporal relations into lasting modifications of network architectures. The required temporal structure of neuronal responses is either inherited from the temporal structure of stimuli or is generated through an oscillatory patterning of neuronal responses by internal mechanisms. The latter permit the selective association of neuronal populations whose responses are not time-locked to stimuli but caused by stimuli lacking temporal structure or are generated internally, e.g. during imagery or recall of memories. Evidence that this option is likely exploited has been provided by investigations on memory consolidation in human subjects (Miltner et al. 1999; Fell et al. 2011; Axmacher et al. 2008) and animals (Yamamoto et al. 2014; for additional citations see Singer 2017).

7 Critical Issues

It has been argued that synchronization of oscillatory activity cannot have a functional role because establishing synchrony would be too slow to be compatible with processing speed. Even more problematic is that synchronous oscillations, when assessed with conventional Fourier or wavelet analyses, are difficult to detect in responses to complex stimuli such as cluttered scenes or images lacking clear high-contrast boundaries (Lima et al. 2010). Moreover, it has been argued that the strong dependence of oscillation frequency and power on stimulus parameters is incompatible with the idea that spike synchronization can be used to encode semantic relations as postulated by the "binding by synchrony" hypothesis (Singer 1993; Singer and Gray 1995) or to gate communication through coherence as postulated by the CTC hypothesis (Fries 2005). These arguments are made explicit in Atallah and Scanziani (2009), Burns et al. (2010, 2011), Ray and Maunsell (2010), Jia et al. (2013a, b) and are reviewed in Ray and Maunsell (2015) and Palmigiano et al. (2017).

However, recent experimental evidence from recordings in awake, behaviourally trained monkeys and simulation studies let these arguments appear in a different light. They revealed that synchronization phenomena are indeed very volatile and characterized by non-stationarity, frequency variability, short duration and rapid phase shifts when animals explore natural scenes. But these studies have also shown that even brief bouts of coherent activity are informative. They allow the decoding of the contents of working memory (Lundqvist et al. 2016), facilitate the communication between cortical areas (Siegel et al. 2008, Bastos et al. 2015), support the dynamic formation of functional networks (Buschman and Miller 2007) and determine the direction of information flow (Lowet et al. 2017). These experimental observations are in agreement with the dynamics of simulated recurrently coupled networks of spiking neurons. Korndörfer et al. (2017) have recently demonstrated that in such networks, neurons engage very rapidly in synchronous discharges when activated by structured input, and that the synchronization probability is determined by the strength of coupling. In this study, no explicit oscillatory properties of the nodes were implemented but the spiking neurons had the usual refractory period and hence shared features of relaxation oscillators.

Another comprehensive simulation study investigated the effect of synchronization on information transfer in recurrent oscillator networks (Palmigiano et al. 2017). The authors simulated a delay-coupled recurrent network with spiking excitatory and inhibitory neurons that shared essential connectivity motifs of the superficial layers of the cerebral cortex. The discharge statistics of individual neurons (nodes) were stochastic but the averaged excitability fluctuations of local clusters of neurons exhibited bursts of synchronous oscillations in the 40 Hz range that lasted only a few cycles. The interesting and unexpected outcome of measurements of information transfer in this delay-coupled oscillatory network was that phase shifts could lead to a rapid reversal of the direction of information transfer. This phase-dependent gating of information flow was particularly effective and flexible when the network operated in a regime characterized by transient rather than sustained oscillations.

The short duration of the oscillatory bursts allowed for fast opening and closing of transmission channels by frequency tracking. The authors concluded that "features that at first sight appear to be noncompliant with information routing may actually provide the brain with a particularly flexible routing mechanism". A similar conclusion was reached in a study by Lowet et al. (2017) who investigated with massive parallel recordings the dynamics of synchronized oscillations in the visual cortex of awake monkeys. The observed dynamics resembled in great detail those reproduced by the simulated network and confirm essential predictions of the hypothesis that the superficial layers of the cerebral cortex can be considered as a delay-coupled recurrent oscillator network, whose dynamics follow the Arnold tongue formalism.

Taken together, both the results of electrophysiological experiments and simulation studies indicate that synchronization of spike discharges can be fast enough to serve feature binding/perceptual grouping within the short inter-saccadic fixation intervals (see also Lowet et al. 2016). In addition, the transient and variable nature of synchronized gamma oscillations characteristic for free viewing conditions and the processing of complex scenes is advantageous for the numerous functions assigned to synchronization. The fast fluctuations between synchronized and uncorrelated states support flexible binding of distributed feature selective nodes into functionally coherent assemblies (Singer 1999), allow for the rapid and flexible definition of relations between distributed neuronal responses as is required for attention-dependent input selection (Fries et al. 2001a) and permit the flexible and selective routing of signals on the backbone of the fixed connectome, as is required for the task-dependent formation of functional networks (Roelfsema et al. 1997; Siegel et al. 2008, 2015). Recent experimental results obtained in awake monkeys by Lowet et al. (2016), Bosman et al. (2009) on the effect of microsaccades and Brunet et al. (2015) on oscillations during free viewing of natural scenes are fully compatible with this view.

Another argument questioning the advantages of assembly codes is that the readout of assemblies requires again conjunction-specific neurons that receive convergent input from the nodes constituting an assembly and convert relational information again into a labelled line code. In this case, so the argument, assembly codes provide no advantage. However, it is also conceivable that exactly this alternation between labelled line and assembly coding, i.e. the generation of conjunction-specific neurons and the formation of assemblies, is an ideal way to capitalize on the respective advantages of the two strategies for the encoding of relations. Assemblies can be read out by connecting a subpopulation of assembly nodes to readout neurons. The activity of these readout neurons will increase when the number and the discharge rate of the feeding nodes increases and in particular, when the synchrony (coherence) of the discharges of the respective nodes increases. Thus, in a sense such readout neurons function both as conjunction-specific neurons and as classifiers. They can either be used to directly control effectors, in which case the classifier function would dominate. Or they could serve as nodes of the recurrent network at the next higher processing level. In this case, they figure as conjunction-specific neurons responding selectively to the combination of features represented by the dynamic assembly at the preceding processing level. The problem of processing speed arising in purely

rate-coded feed-forward architectures vanishes since the synchronous activity of assemblies can ignite conjunction-specific readout neurons without requiring time-consuming temporal summation. Because of the exquisite sensitivity of neurons to coincident input (Salinas and Sejnowski 2000), a single barrage of synchronized discharges from the nodes of an assembly suffices to drive the conjunction-specific neurons at the next higher level. In addition, the advantages of recurrent computation are preserved. The associativity of recurrent connectivity can be exploited to disambiguate relations among features through flexible binding and to thereby cope with the combinatorial explosion of possible feature constellations. And the reverberating dynamics of recurrent networks permits to handle the encoding of sequences. Thus, major downsides of the strategy to encode relations by convergence in feed-forward architectures are compensated by combining this architecture with recurrent networks.

Moreover, as far as we understand the system´s coding strategy, processing is distributed from the very early sensory structures all the way to the motor output. Thus, there is no bottleneck that would ultimately require explicit condensation and representation of relational constructs in the activation of individual cells (grandmother cells). Rather, the connectome of the cerebral cortex permits to map assemblies onto assemblies in an iterative way. Thus, the great advantage of assembly coding, the ability to cope with the infinite number of possible relations of features by dynamic binding can be maintained throughout the whole processing stream.

8 Computing in High-Dimensional State Space

Recurrent networks exhibit highly complex non-linear dynamics, especially if the nodes are configured as oscillators and if the coupling connections impose delays—as is the case for natural networks. These dynamics provide a very high-dimensional state space that can be exploited for the implementation of functions that go beyond those discussed above: the encoding of temporal sequences, the storage and ultra-fast retrieval of vast amounts of information and the fast and effective classification of complex spatio-temporal input patterns. In the following, some of these options will be discussed and substantiated with recently obtained experimental evidence.

The non-linear dynamics of recurrent networks are exploited for computation in certain AI systems, the respective strategies being addressed as “echo state, reservoir or liquid computing” (Lukoševičius and Jaeger 2009; Buonomano and Maass 2009; D’Huys et al. 2012; Soriano et al. 2013). In most cases, the properties of recurrent networks are simulated in digital computers, whereby only very few of the features of biological networks are captured. The nodes act as simple integrators and the coupling connections lack most of the properties of their natural counterparts. They operate without delay, lack specific topologies and their gain is non-adaptive. Moreover, most artificial recurrent networks lack inhibitory interneurons that constitute 20% of the neurons in natural systems and interact in highly selective ways with the excitatory neurons. Moreover, as the updating of network states has to be performed

sequentially according to the clock cycle, many of the analog computations taking place in natural networks can only be approximated with iterations if at all. Therefore, attempts are made to emulate the dynamics of recurrent networks with analog technology. An original and hardware-efficient approach is based on optoelectronics. Laser diodes serve as oscillating nodes, and these are reciprocally coupled through glass fibres whose variable length introduces variations of coupling delays (Soriano et al. 2013). All these implementations have in common to use the characteristic dynamics of recurrent networks as a medium for the execution of specific computations. Because the dynamics of recurrent networks resemble to some extent the dynamics of liquids—hence the term "liquid computing"—the basic principle can be illustrated by considering the consequences of perturbing a liquid. If objects impact at different intervals and locations in a pond of water, they generate propagating waves whose parameters reflect the size, impact speed and location of the objects. The wave patterns fade with a time constant determined by the viscosity of the liquid, interfere with one another and create a complex dynamic state. This state can be analysed by measuring at several locations in the pond the amplitude, frequency and phase of the respective oscillations and from these variables a trained classifier can subsequently reconstruct the exact sequence and nature of the impacting "stimuli". Similar effects occur in recurrent networks when subsets of nodes are perturbed by stimuli that have a particular spatial and temporal structure. The excitation of the stimulated nodes spreads across the network and creates a complex dynamic state, whose spatio-temporal structure is determined by the constellation of initially excited nodes and the functional architecture of the coupling connections. This stimulus-specific pattern continues to evolve beyond the duration of the stimulus due to reverberation and eventually fades. The network state returns to its initial state. This evolution of the network dynamics can be traced by assessing the activity changes of the nodes and is usually represented by time-varying, high-dimensional vectors or trajectories. As these trajectories differ for different stimulus patterns, segments exhibiting maximal distance in the high-dimensional state space can be selected to train classifiers for the identification of the respective stimuli.

This computational strategy has several advantages: (i) Low-dimensional stimulus events are projected into a high-dimensional state space where non-linearly separable stimuli become linearly separable; (ii) the high dimensionality of the state space can allow for the mapping of more complicated output functions (like the XOR) by simple classifiers and (iii) information about sequentially presented stimuli persists for some time in the medium (fading memory). Thus, information about multiple stimuli can be integrated over time, allowing for the representation of sequences. These properties make artificial recurrent networks extremely effective for the classification of input patterns that have both spatial and temporal structures and share overlapping features in low-dimensional space. Moreover, because these networks self-organize and produce spatio-temporally structured activity patterns, they have generative properties and can be used for pattern completion, the formation of novel associations and the generation of patterns for the control of movements. Consequently, an increasing number of AI systems now complement the feed-forward strategy implemented in deep learning networks with algorithms inspired by recurrent networks. One of these

powerful and now widely used algorithms is the Long Short-Term Memory (LSTM) algorithm, introduced decades ago by Hochreiter and Schmidhuber (1997) and used in systems such as AlphaGo (Silver et al. 2017, 2018). The surprising efficiency of these systems that excels in certain domains of human performance has nurtured the notion that brains operate in the same way. If one considers, however, how fast brains can solve certain tasks despite their comparatively extremely slow components and how energy-efficient they are, one is led to suspect the implementation of additional strategies.

And indeed, natural recurrent networks differ from their artificial counterparts in several important features which is the likely reason for their amazing performance. In sensory cortices, the nodes are feature-selective, i.e. they can be activated only by specific spatio-temporal stimulus configurations. The reason is that they receive convergent input from selected nodes of the respective lower processing level and thus function as conjunction-specific units in very much the same way as the nodes in feed-forward multilayer networks. In low areas of the visual system, for example, the nodes are selective for elementary features such as the location and orientation of contour borders while in higher areas of the processing hierarchy, the nodes respond to increasingly complex constellations of elementary features. In addition, the nodes of natural systems, the neurons, possess an immensely larger spectrum of integrative and adaptive functions than the nodes currently used in artificial recurrent networks. And finally, the neurons and/or their embedding microcircuits are endowed with the propensity to oscillate.

The recurrent connections within the respective layers also differ in important respects from those implemented in most artificial networks. Because of the slow velocity of signals conveyed by neuronal axons, interactions occur with variable delays. These delays cover a broad range and depend on the distance between interconnected nodes and the conduction velocity of the respective axons. Furthermore and most importantly, the connections are endowed with plastic synapses whose gain changes according to the correlation rules discussed above. Connections strengthen between nodes whose activity is often correlated, and they weaken between nodes whose activity is mostly uncorrelated. As a consequence, coupling is highly anisotropic. Nodes tuned to features that often co-occur in natural environments tend to be more strongly coupled than nodes responding to features that rarely occur simultaneously. Thus, through both experience-dependent pruning of connections during early development and experience-dependent synaptic plasticity, statistical contingencies between features of the environment get internalized and stored not only in the synaptic weights of feed-forward connections to feature-selective nodes but also in the weight distributions of the recurrent connections. Thus, in low levels of the processing hierarchy the weight distributions of the recurrent coupling connections reflect statistical contingencies of simple and at higher levels of more complex constellations of features. In other words, the hierarchy of reciprocally coupled recurrent networks contains a model of the world that reflects the frequency of co-occurrence of typical relations among the features/components of composite perceptual objects. Recent simulation studies have actually shown that the performance of an artificial recurrent network is substantially improved if the recurrent

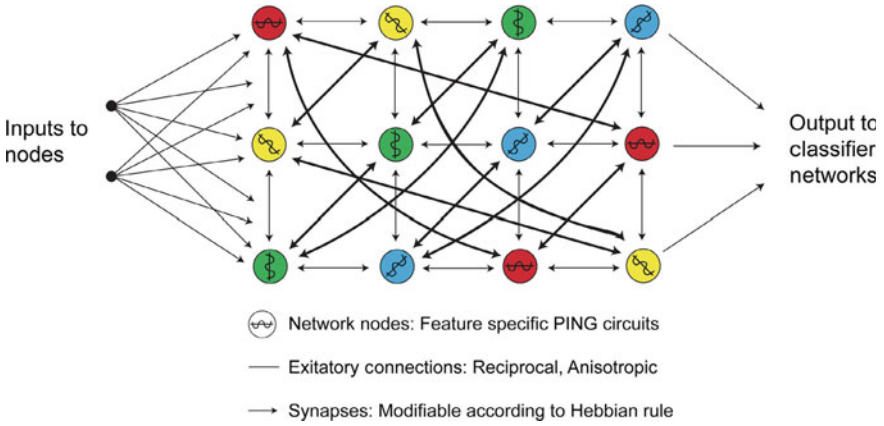


Fig. 1 Schematic representation of wiring principles in supragranular layers of the visual cortex. The coloured discs (nodes) stand for cortical columns that are tuned to specific features (here stimulus orientation) and have a high propensity to engage in oscillatory activity due to the intrinsic circuit motif of recurrent inhibition. These functional columns are reciprocally coupled by a dense network of excitatory connections that originate mainly from pyramidal cells and terminate both on pyramidal cells and inhibitory interneurons in the respective target columns. Because of the genetically determined span of these connections, coupling decreases exponentially with the distance between columns. However, these connections undergo use-dependent selection during development and remain susceptible to Hebbian modifications of their gain in the adult. The effect is that the weight distributions of these connections and hence the coupling strength among functional columns (indicated by thickness of lines) reflect the statistical contingencies of the respective features in the visual environment (for further details, see text). (From Singer (2018) Neuronal oscillations: unavoidable and useful? Europ J Neurosci 48:2389–2398.)

connections are made adaptive and can "learn" about the feature contingencies of the processed patterns (Lazar et al. 2009; Hartmann et al. 2015). A simplified representation of the essential features of the delay-coupled oscillator network supposed to be realized in the superficial layers of the visual cortex is shown in Fig. 1.

9 Information Processing in Natural Recurrent Networks, a Proposal

Theories of perception, formulated more than a 100 years ago (von Helmholtz 1867) and a plethora of experimental evidence indicate that perception is the result of a constructivist process. Sparse and noisy input signals are disambiguated and interpreted on the basis of an internal model of the world. This model is used to reduce redundancy, to detect characteristic relations between features, to bind signals evoked by features constituting a perceptual object, to facilitate the segregation of figures from background and to eventually enable identification and classification. The store containing such an elaborate model must have an immense capacity given that the

interpretation of ever-changing sensory input patterns requires knowledge about the vast number of distinct feature conjunctions characterizing perceptual objects. Moreover, this massive amount of prior knowledge needs to be arranged in a configuration that permits ultra-fast readout to meet the constraints of processing speed. Primates perform on average four saccades per second. This implies that new visual information is sampled approximately every 250 ms (Maldonado et al. 2008; Ito et al. 2011) and psychophysical evidence indicates that attentional processes sample visual information at comparable rates (Landau 2018). Thus, the priors required for the interpretation of a particular sensory input need to be made available within fractions of a second.

How the high-dimensional non-linear dynamics of delay-coupled recurrent networks could be exploited to accomplish these complex functions is discussed in the following paragraph.

A hallmark of natural recurrent networks such as the cerebral cortex is that they are spontaneously active. The dynamics of this resting activity must reflect the weight distributions of the structured network and hence must harbour the entirety of the stored "knowledge" about the statistics of feature contingencies, i.e. the latent priors used for the interpretation of sensory evidence. This predicts that resting activity is high dimensional and represents a vast but constrained manifold inside the universe of all theoretically possible dynamical states. Once input signals become available they are likely to trigger a cascade of effects: They drive in a graded way a subset of feature-sensitive nodes *and* thereby perturb the network dynamics. If the evidence provided by the input patterns matches well the priors stored in the network architecture, the network dynamics will collapse to a specific substate that provides the best match with the corresponding sensory evidence. Such a substate is expected to have a lower dimensionality and to exhibit less variance than the resting activity, to possess a specific correlation structure and be metastable due to reverberation among nodes supporting the respective substate. Because these processes occur within a very high-dimensional state space, substates induced by different input patterns are likely to be well segregated and therefore easy to classify. As the transition from the high-dimensional resting activity to substates is likely to follow stimulus-specific trajectories, classification of stimulus-specific patterns should be possible once trajectories have sufficiently diverged and before they reach a fix point. These points of divergence should be reached faster for input patterns that match particularly well with the internal priors.

10 Experimental Evidence

Experimental studies testing such a scenario are still rare and have become possible only with the advent of massive parallel recordings from the network nodes. So far, however, the few predictions that have been subject to experimental testing appeared to be confirmed. The covariance structure of resting activity does indeed reflect the anisotropic layout of the recurrent connections (Kenet et al. 2003; Fries et al. 2001b;

Bosking et al. 1997), is modified by experience during early development (Smith et al. 2018) and learning throughout life (Lewis et al. 2009; Kundu et al. 2013) and recapitulates features of the environment (Berkes et al. 2011). Evidence also indicates that the resting dynamics of cortical networks are high dimensional (Moca et al. 2019) and characteristic of complex systems operating close to criticality (Plenz and Thiagarajan 2007; Hahn et al. 2010; Priesemann et al. 2014). Multisite recordings reveal in addition that spontaneous activity is organized in spatio-temporal patterns such as travelling waves and propagating avalanches whose amplitude distribution can be fitted with power law functions (Plenz and Thiagarajan 2007; Hahn et al. 2010; Ermentrout and Kleinfeld 2001). When the network is activated by sensory stimuli, this high-dimensional resting state gets constrained. As indicated by a decrease of the Fano factor and a reduction of fractal dimensionality, the variance and the dimensionality of the dynamics decrease (Churchland et al. 2010; Bányai et al. 2019; Moca et al. 2019). This is compatible with the formation of metastable, coherent substates.

Confirmed is also the prediction that the stimulus-specific substates outlast the duration of the stimuli due to network reverberation and that stimulus-specific states induced by sequences of different stimuli can become superimposed but remain segregated in high-dimensional space. Responses to successively presented visual stimuli (letters and numbers) were recorded with matrix electrodes simultaneously from a random sample of ~60 neurons in the cat primary visual cortex ,and linear classifiers were trained on short segments (5–100 ms) of the activity vectors of a training set of responses and then these classifiers were used to identify the nature of the presented stimuli in a test set (Nikolic et al. 2009). These experiments revealed that (i) the information about a particular stimulus persists in the activity of the network for up to a second after the end of the stimulus (fading memory), (ii) the information about sequentially presented stimuli superimposes so that two subsequent stimuli and the order of their presentation can be correctly classified with a linear classifier sometime after the end of the second stimulus, suggesting that the network is capable of performing non-linear XOR operations and (iii) the information about stimulus identity is distributed across many neurons (>30) and encoded in the rate vector *and* the temporal correlation structure of the responses.

Evidence has also been obtained that sensory signals evoked by natural scenes and hence matching the priors stored in the network cause a collapse of high-dimensional network dynamics into metastable subregions of the state space that are stimulus specific. These substates are distinguished by enhanced coherence (covariance) of discharge rate and synchronization of oscillatory activity, reduced variability and lower dimensionality (Churchland et al. 2010; Bányai et al. 2019; Moca et al. 2019). For low levels of the visual system, a particularly good match with stored priors is achieved with grating stimuli. Gratings match several Gestalt criteria (priors), namely those of continuity, colinearity, similarity in feature space (in this case, in the orientation domain), regularity and—if the grating drifts—common fate. These stimuli elicit particularly stable and coherent states characterized by sustained, well synchronized oscillatory activity in the gamma frequency range (Gray and Singer 1989; Gray et al. 1989). This confirms the prediction that stimuli matching particularly well with stored priors lead to a fast collapse of systems dynamics towards

stable, highly coherent and low-dimensional substates characterized by synchronous gamma oscillations (see also Vinck and Bosman 2016). Interestingly, under certain experimental conditions, stimulus-induced substates seem to develop only with some delay (~100–200 ms) after the initial phasic response of the network nodes. This is suggested by the fact that classification of stimulus-specific states and segregation of the principal components of the response vectors are better at later phases of the response, when the stimulus is no longer present and the amplitude of the responses already decaying. Stimuli that do not match the stored priors (e.g. "unnatural" stimuli) evoke substates that are less coherent, and their correlation structure is less stimulus specific (Banyai et al. 2019). Accordingly, these states are more difficult to classify (Andreea Lazar, pers. communication).

Evidence also indicates that the cortical network "learns" in a non-supervised way about stimulus statistics and exploits this knowledge in order to optimally segregate the representations of different stimuli. Repeated presentation of stimuli has been shown to cause changes in the network with the consequence that familiar stimuli evoke substates that are better classifiable than those evoked by less familiar stimuli. The reason is that substates evoked by familiar stimuli are better segregated in high-dimensional dynamic space (Lazar et al. 2018). As predicted by the dependence of the structure of resting state activity on the functional architecture of the recurrent network (see above), these experience-dependent modifications of the network are reflected in resting state activity: The vectors specific for highly familiar stimuli are spontaneously replayed (Lazar pers. communication).

There are also indications that top-down mechanisms related to attention and expectancy constrain the dynamic space of recurrent networks at lower levels of processing. Stimuli that are cued as being relevant for behavioural reactions and reward evoke more strongly synchronized oscillatory responses than behaviourally irrelevant stimuli (Fries et al. 2001a; Lima et al. 2011). Also, one observes a ramp-up of the power of synchronized gamma oscillations and a change in the dimensionality of the network dynamics following the presentation of a cue that instructs the animal about the sequence of future events (Moca et al. 2019). These top-down influences are likely to change the correlation structure of the activity vectors in anticipation of a forthcoming cognitive task, and this preparation could accelerate the formation of specific substates once sensory evidence becomes available. This possibility awaits further experimental testing.

11 Learning Mechanisms

As discussed above, the learning mechanisms implemented in biological systems differ from those used to train artificial networks. Promising alternatives to the back-propagation algorithm have recently been proposed for the training of recurrent networks, and these strategies share some features with the learning mechanisms implemented in natural systems (Bellec et al. 2019). However, essential differences

remain. The main reason is the lack of hardware solutions that mimic the functions of adaptive (Hebbian) synapses and the gating of synaptic plasticity by global, value assigning systems. Adaptive synapses are ideally suited to mediate unsupervised learning processes in feed-forward and recurrent networks (see above; for references, see Galuske et al. 2019). However, they can also support supervised learning because their adaptive functions are controlled by modulatory systems. These systems evaluate the outcome of behaviour and are closely related to reward assigning networks using dopamine as a transmitter (Schultz 2016; Schultz et al. 1997) and to networks controlling arousal and attention, using acetylcholine and norepinephrine as a transmitter (Singer 1995; Bear and Singer 1986; Galuske et al. 2019). The signals provided by these systems are distributed throughout the brain by widely branching axonal networks and exert a gating function for synaptic plasticity both during development and adult learning. These signals interfere with the crucial variables determining the occurrence and polarity of use-dependent synaptic gain changes: the membrane potential of neurons and the molecular cascades translating neuronal activity into long-term changes of synaptic transmission. In the absence of these modulatory gating signals, synaptic modifications do not occur. This prevents spurious activity patterns to induce dysfunctional modifications of the functional architecture of neuronal networks. For lasting changes to occur, it is required that these systems are tonically active and maintain a critical level of excitability. This is the case when brains are awake and attentive. In order to gate synaptic changes as a function of behavioural outcomes, some of the modulatory systems have in addition a "now print" function. They emit "reward" signals when a particular behaviour had the desired effect, and this leads to the selective strengthening of synapses that were involved in bringing about the respective behaviour. As the outcome of a behaviour can only be evaluated once it is executed and the activation of the involved synapses has returned to baseline, the now print signal must be able to act retrogradely in time. This is achieved by a process called "synaptic tagging" (Redondo and Morris 2011; Frey and Morris 1997). Activation patterns that are sufficiently well structured and coherent to induce synaptic modifications—which is usually the case when networks have converged to "meaningful" substates—induce short lasting modifications of synaptic gain, and these are associated with the generation of molecular tags. In the absence of now print signals, both the gain changes and the tags fade with time. However, when the now print signals are emitted once the behavioural outcome is known, the gain change of the tagged synapses becomes consolidated and long lasting. A further mechanism for the consolidation of synaptic gain changes is replay. This mechanism seems to consolidate modifications irrespective of whether they have been induced by supervised or unsupervised learning. In both cases, the functional architecture of the networks is altered and reflects the novel weight distributions of recurrent connections. As alluded to above, the correlation structure of resting activity does reflect the functional architecture of the recurrent network and evidence is available both from recordings in the hippocampus and the cerebral cortex; those activity vectors associated with previous experience are replayed during resting activity and certain sleep phases, and can be correctly classified (Diba and Buzsáki 2007; Lazar

pers. communication). In the hippocampus, this replay is associated with an oscillatory patterning of resting activity and compressed in time, i.e. the firing sequences of distributed neurons characteristic for previous behaviour are reproduced but at a faster time scale. Evidence is available that this "rehearsal" of activity vectors produced by previous behaviour (experience) serves the consolidation of memories (Ego-Stengel and Wilson 2010).

Another important feature of use-dependent synaptic plasticity in natural systems is that mechanisms are implemented that counteract saturation of synaptic gain and runaway dynamics. Neurons down-regulate their activity in response to an increase of excitatory drive. This normalization process preserves the relative weight distribution of converging inputs and thus the specificity of learning-dependent synaptic gain changes but keeps the average discharge rate of neurons constant (Turrigiano and Nelson 2004).

12 Concluding Remarks

Despite considerable effort, there is still no unifying theory of cortical processing. As a result, numerous experimentally identified phenomena lack a cohesive theoretical framework. This is particularly true for the dynamic phenomena reviewed here because they cannot easily be accommodated in the prevailing concepts that emphasize serial feed-forward processing and labelled line codes. However, the cortical connectome with its preponderance of reciprocal connections and the rich dynamics resulting from these reciprocal interactions suggest that additional processing strategies are implemented. Theoretical considerations based on the connectivity of cortical networks and recent evidence suggest that neuronal systems, especially the neocortex, combine two complementary strategies for the analysis and encoding of relations among the features of composite perceptual objects (feature binding): The generation of conjunction-specific neurons in hierarchically organized feed-forward architectures (labelled line code) and the formation of dynamically configured assemblies of transiently cooperating neurons (assembly code).

Here, we have proposed a concept that assigns specific functions to oscillations, synchrony and the more complex dynamics emerging from a delay-coupled recurrent network. This concept is fully compatible with the robust evidence for the encoding of relations by anatomical convergence (labelled line codes) but complements this mechanism by a scenario in which the precise temporal relations among the discharges of coupled neurons serve as complementary code for the definition of semantic relations both in signal processing and learning. In addition, a computational strategy is introduced that capitalizes on the high-dimensional coding space offered by delay-coupled recurrent networks. In this conceptual framework, information is distributed and encoded both in the discharge rate of individual nodes (labelled lines) and to a substantial degree also in the precise temporal relations among the discharge sequences of distributed nodes. The core of the hypothesis is that the dynamic interactions within delay-coupled recurrent oscillator networks (i) endow responses with

the precise temporal structure required for the encoding (binding) and learning of semantic relations, (ii) exhibit complex, high-dimensional correlation structures that reflect the weight distributions of the coupling connections and serve as space for the storage of a vast amount of information (internal model of the world, priors), (iii) permit ultra-fast comparison between input patterns and stored priors through rapid convergence towards stimulus-specific dynamic substates, and iv) allow for easy classification of the results of this comparison because input-specific substates occupy well-segregated loci in the high-dimensional state space. The analysis of the correlation structure of these high-dimensional response vectors is still at the very beginning. However, methods are now available for massive parallel recording from large numbers of network nodes in behaving animals. It is to be expected, therefore, that many of the predictions derivable from the novel concept will be amenable to experimental testing in the near future.

References

M. Abeles, *Corticonics* (Cambridge University Press, Cambridge, 1991)

D.G. Aronson, G.B. Ermentrout, N. Kopell, Amplitude response of coupled oscillators. Phys. D **41**, 403–449 (1990)

A. Artola, S. Bröcher, W. Singer, Different voltage-dependent thresholds for the induction of long-term depression and long-term potentiation in slices of the rat visual cortex. Nature **347**, 69–72 (1990)

B.V. Atallah, M. Scanziani, Instantaneous modulation of gamma oscillation frequency by balancing excitation with inhibition. Neuron **62**, 566–577 (2009)

B.B. Averbeck, P.E. Latham, A. Pouget, Neural correlations, population coding and computation. Nat. Rev. Neurosci. **7**, 358–366 (2006)

N. Axmacher, D.P. Schmitz, T. Wagner, C.E. Elger, J. Fell, Interactions between medial temporal lobe, prefrontal cortex, and inferior temporal regions during visual working memory: a combined intracranial EEG and functional magnetic resonance imaging study. J. Neurosci. **28**, 7304–7312 (2008)

M. Bányai, A. Lazar, L. Klein, J. Klon-Lipok, M. Stippinger, W. Singer, G. Orbán, Stimulus complexity shapes response correlations in primary visual cortex. Proc. Natl. Acad. Sci. U.S.A. **116**, 2723–2732 (2019)

H.B. Barlow, Single units and sensation: a neurone doctrine for perceptual psychology? Perception **1**, 371–394 (1972)

A.M. Bastos, J. Vezoli, C.A. Bosman, J.-M. Schoffelen, R. Oostenveld, J.R. Dowdall, P. De Weerd, H. Kennedy, P. Fries, Visual areas exert feedforward and feedback influences through distinct frequency channels. Neuron **85**, 390–401 (2015)

M.F. Bear, W. Singer, Modulation of visual cortical plasticity by acetylcholine and noradrenaline. Nature **320**, 172–176 (1986)

G. Bellec, F. Scherr, E. Hajek, D. Salaj, R. Legenstein, W. Maass, Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets (2019), https://arxiv.org/abs/2553450:1-34

P. Berkes, G. Orbán, M. Lengyel, J. Fiser, Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. Science **331**, 83–87 (2011)

G.Q. Bi, M.M. Poo, Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. J. Neurosci. **18**, 10464–10472 (1998)

E.L. Bienenstock, L.N. Cooper, P.W. Munro, Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. J. Neurosci. **2**, 32–48 (1982)
C. Börgers, N.J. Kopell, Gamma oscillations and stimulus selection. Neural Comput. **20**, 383–414 (2008)
W.H. Bosking, Y. Zhang, B. Schofield, D. Fitzpatrick, Orientation selectivity and the arrangement of horizontal connections in tree shrew striate cortex. J. Neurosci. **17**, 2112–2127 (1997)
C.A. Bosman, T. Womelsdorf, R. Desimone, P. Fries, A microsaccadic rhythm modulates gamma-band synchronization and behavior. J. Neurosci. **29**, 9471–9480 (2009)
N. Brunet, C. Bosman, M. Roberts, R. Oostenveld, T. Womelsdorf, P. De Weerd, P. Fries, Visual cortical gamma-band activity during free viewing of natural images. Cereb. Cortex **25**, 918–926 (2015)
R.M. Bruno, B. Sakmann, Cortex is driven by weak but synchronously active thalamocortical synapses. Science **312**, 1622–1627 (2006)
D.V. Buonomano, W. Maass, State-dependent computations: spatiotemporal processing in cortical networks. Nat. Rev. Neurosci. **10**, 113–125 (2009)
G. Buracas, A. Zador, M. Deweese, T. Albright, Efficient discrimination of temporal patterns by motion-sensitive neurons in primate visual cortex. Neuron **20**, 959–969 (1998)
S.P. Burns, D. Xing, M.J. Shelley, R.M. Shapley, Searching for autocoherence in the cortical network with a time-frequency analysis of the local field potential. J. Neurosci. **30**, 4033–4047 (2010)
S.P. Burns, D. Xing, R.M. Shapley, Is gamma-band activity in the local field potential of V1 cortex a "clock" or filtered noise? J. Neurosci. **31**, 9658–9664 (2011)
T.J. Buschman, E.K. Miller, Top-down versus bottom-up control of attention in the prefrontal and posterior parietal cortices. Science **315**, 1860–1862 (2007)
G. Buzsáki, X.-J. Wang, Mechanisms of gamma oscillations. Annu. Rev. Neurosci. **35**, 203–225 (2012)
G. Buzsáki, N. Logothetis, W. Singer, Scaling brain size, keeping timing: evolutionary preservation of brain rhythms. Neuron **80**, 751–764 (2013)
M.M. Churchland et al., Stimulus onset quenches neural variability: a widespread cortical phenomenon. Nat. Neurosci. **13**, 369–378 (2010)
O. D'Huys, I. Fischer, J. Danckaert, R. Vicente, Spectral and correlation properties of rings of delay-coupled elements: Comparing linear and nonlinear systems. Phys. Rev. E **85**(056209), 1–5 (2012)
K. Diba, G. Buzsáki, Forward and reverse hippocampal place-cell sequences during ripples. Nat. Neurosci. **10**, 1241–1242 (2007)
J.J. DiCarlo, D.D. Cox, Untangling invariant object recognition. Trends Cogn. Sci. **11**, 333–341 (2007)
M. Diesmann, M.-O. Gewaltig, A. Aertsen, Stable propagation of synchronous spiking in cortical neural networks. Nature **402**, 529–533 (1999)
V. Ego-Stengel, M.A. Wilson, Disruption of ripple-associated hippocampal activity during rest impairs spatial learning in the rat. Hippocampus **20**, 1–10 (2010)
G.B. Ermentrout, D. Kleinfeld, Traveling electrical waves in cortex: insights from phase dynamics and speculation on a computational role. Neuron **29**, 33–44 (2001)
J. Fell, E. Ludowig, B.P. Staresina, T. Wagner, T. Kranz, C.E. Elger, N. Axmacher, Medial temporal theta/alpha power enhancement precedes successful memory encoding: Evidence based on intracranial EEG. J. Neurosci. **31**, 5392–5397 (2011)
D.J. Felleman, D.C. van Essen, Distributed hierarchical processing in the primate cerebral cortex. Cereb. Cortex **1**, 1–47 (1991)
U. Frey, R.G.M. Morris, Synaptic tagging and long-term potentiation. Nature **385**, 533–536 (1997)
P. Fries, A mechanism for cognitive dynamics: neuronal communication through neuronal coherence. Trends Cogn. Sci. **9**, 474–480 (2005)
P. Fries, J.H. Reynolds, A.E. Rorie, R. Desimone, Modulation of oscillatory neuronal synchronization by selective visual attention. Science **291**, 1560–1563 (2001a)

P. Fries, S. Neuenschwander, A.K. Engel, R. Goebel, W. Singer, Rapid feature selective neuronal synchronization through correlated latency shifting. Nat. Neurosci. **4**, 194–200 (2001b)
P. Fries, D. Nikolic, W. Singer, The gamma cycle. Trends Neurosci. **30**, 309–316 (2007)
R.A.W. Galuske, M.H.J. Munk, W. Singer, Relation between gamma oscillations and neuronal plasticity in the visual cortex. Proc. Natl. Acad. Sci. USA, 116, 23317–23375
C.D. Gilbert, T.N. Wiesel, Columnar specificity of intrinsic horizontal and corticocortical connections in cat visual cortex. J. Neurosci. **9**, 2432–2442 (1989)
L. Glass, J. Sun, Periodic forcing of a limit-cycle oscillator: fixed points, Arnold tongues, and the global organization of bifurcations. Phys. Rev. E **50**(6), 5077–5084 (1994)
M.F. Glasser, T.S. Coalson, E.C. Robinson, C.D. Hacker, J. Harwell, E. Yacoub, K. Ugurbil, J. Andersson, C.F. Beckmann, M. Jenkinson, S.M. Smith, D.C. Van Essen, A multi-modal parcellation of human cerebral cortex. Nature **536**, 171–178 (2016)
C.M. Gray, W. Singer, Stimulus-specific neuronal oscillations in orientation columns of cat visual cortex. Proc. Natl. Acad. Sci. U.S.A. **86**, 1698–1702 (1989)
C.M. Gray, P. König, A.K. Engel, W. Singer, Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties. Nature **338**, 334–337 (1989)
C.G. Gross, C.E. Rocha-Miranda, D.B. Bender, Visual properties of neurons in inferotemporal cortex of the macaque. J. Neurophysiol. **35**, 96–111 (1972)
G. Hahn, T. Petermann, M.N. Havenith, Y. Yu, W. Singer, D. Plenz, D. Nikolic, Neuronal avalanches in spontaneous activity in vivo. J. Neurophysiol. **104**, 3312–3322 (2010)
C. Hartmann, A. Lazar, B. Nessler, J. Triesch, Where's the noise? Key features of spontaneous activity and neural variability arise through learning in a deterministic network. PLoS Comput. Biol. **11**(12), e1004640, 1–35 (2015)
D.O. Hebb, *The Organization of Behavior* (Wiley, New York, 1949)
T. Hirabayashi, D. Takeuchi, K. Tamura, Y. Miyashita, Microcircuits for hierarchical elaboration of object coding across primate temporal areas. Science **341**, 191–195 (2013)
S. Hochreiter, J. Schmidhuber, Long short-term memory. Neural Comput. **9**, 1735–1780 (1997)
J.J. Hopfield, Learning algorithms and probability distributions in feed-forward and feed-back networks. Proc. Natl. Acad. Sci. U.S.A. **84**, 8429–8433 (1987)
D.H. Hubel, T.N. Wiesel, Receptive fields and functional architecture of monkey striate cortex. J. Physiol. (Lond.) **195**, 215–243 (1968)
M.F. Iacaruso, I.T. Gasler, S.B. Hofer, Synaptic organization of visual space in primary visual cortex. Nature **547**, 449–452 (2017)
J. Ito, P. Maldonado, W. Singer, S. Grün, Saccade-related modulations of neuronal excitability support synchrony of visually elicited spikes. Cereb. Cortex **21**, 2482–2497 (2011)
X. Jia, D. Xing, A. Kohn, No consistent relationship between gamma power and peak frequency in macaque primary visual cortex. J. Neurosci. **33**, 17–25 (2013a)
X. Jia, S. Tanabe, A. Kohn, Gamma and the coordination of spiking activity in early visual cortex. Neuron **77**, 762–774 (2013b)
K. Kar, J. Kubilius, K. Schmidt, E.B. Issa, J.J. DiCarlo, Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior, 24 June 2018 (2018). http://dx.doi.org/10.1101/354753:1-20
T. Kenet, D. Bibitchkov, M. Tsodyks, A. Grinvald, A. Arieli, Spontaneously emerging cortical representations of visual attributes. Nature **425**, 954–956 (2003)
C. Korndörfer, E. Ullner, J. Garcia, G. Pipa, Cortical spike synchrony as measure of input familiarity. Neural Comput. **29**, 2491–2510 (2017)
B. Kundu, D.W. Sutterer, S.M. Emrich, B.R. Postle, Strengthened effective connectivity underlies transfer of working memory training to tests of short-term memory and attention. J. Neurosci. **33**, 8705–8715 (2013)
Y. Kuramoto, Collective synchronization of pulse-coupled oscillators and excitable units. Phys. D **50**, 15–30 (1990)

A.N. Landau, Neuroscience: a mechanism for rhythmic sampling in vision. Curr. Biol. **28**, R830–R832 (2018)
G. Laurent, Dynamical representation of odors by oscillating and evolving neural assemblies. Trends Neurosci. **19**, 489–496 (1996)
A. Lazar, G. Pipa, J. Triesch, SORN: a self-organizing recurrent neural network. Front. Comput. Neurosci. **3**(23), 1–9 (2009)
A. Lazar, C. Lewis, P. Fries, W. Singer, D. Nikolic, Visual exposure optimizes stimulus encoding in primary visual cortex, 20 December 2018 (2018). http://dx.doi.org/10.1101/502328:1-24
Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature **521**, 436–444 (2015)
C.M. Lewis, A. Baldassarre, G. Committeri, G.L. Romani, M. Corbetta, Learning sculpts the spontaneous activity of the resting human brain. Proc. Natl. Acad. Sci. U.S.A. **106**, 17558–17563 (2009)
B. Lima, W. Singer, N.-H. Chen, S. Neuenschwander, Synchronization dynamics in response to plaid stimuli in monkey V1. Cereb. Cortex **20**, 1556–1573 (2010)
B. Lima, W. Singer, S. Neuenschwander, Gamma responses correlate with temporal expectation in monkey primary visual cortex. J. Neurosci. **31**, 15919–15931 (2011)
S. Löwel, W. Singer, Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity. Science **255**, 209–212 (1992)
E. Lowet, M.J. Roberts, C.A. Bosman, P. Fries, P. De Weerd, Areas V1 and V2 show microsaccade-related 3–4-Hz covariation in gamma power and frequency. Eur. J. Neurosci. **43**, 1286–1296 (2016)
E. Lowet, M.J. Roberts, B. Gips, P. De Weerd, A. Peter, A quantitative theory of gamma synchronization in macaque V1. eLife **6**, e26642, 1–44 (2017). elifesciencesorg
M. Lukoševičius, H. Jaeger, Reservoir computing approaches to recurrent neural network training. Comput. Sci. Rev. **3**, 127–149 (2009)
M. Lundqvist, J. Rose, P. Herman, S.L. Brincat, T.J. Buschman, E.K. Miller, Gamma and beta bursts underlie working memory. Neuron **90**, 152–164 (2016)
Z.F. Mainen, T.J. Sejnowski, Reliability of spike timing in neocortical neurons. Science **268**, 1503–1506 (1995)
P. Maldonado, C. Babul, W. Singer, E. Rodriguez, D. Berger, S. Grün, Synchronization of neuronal responses in primary visual cortex of monkeys viewing natural images. J. Neurophysiol. **100**, 1523–1532 (2008)
N.T. Markov et al., A weighted and directed interareal connectivity matrix for macaque cerebral cortex. Cereb. Cortex **24**, 17–36 (2014)
H. Markram, J. Lübke, M. Frotscher, B. Sakmann, Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. Science **275**, 213–215 (1997)
P.M. Milner, The functional nature of neuronal oscillations. Trends Neurosci. **15**, 387 (1992)
W.H.R. Miltner, C. Braun, M. Arnold, H. Witte, E. Taub, Coherence of gamma-band EEG activity as a basis for associative learning. Nature **397**, 434–436 (1999)
V. Moca, L. Klein, H. Klon-Lipok, R. Muresan, W. Singer, Attention effects on the complexity of cortical dynamics (in prep.)
E.N. Niebur, H.G. Schuster, D.M. Kammen, Collective frequencies and metastability in networks of limit cycle oscillators with time delay. Phys. Rev. Lett. **67**, 2753–2756 (1991)
D. Nikolic, S. Häusler, W. Singer, W. Maass, Distributed fading memory for stimulus properties in the primary visual cortex. PLoS Biol. **7**(e1000260), 1–19 (2009)
S. Pajevic, P.J. Basser, R.D. Fields, Role of myelin plasticity in oscillations and synchrony of neuronal activity. Neuroscience **276**, 135–147 (2014)
A. Palmigiano, T. Geisel, F. Wolf, D. Battaglia, Flexible information routing by transient synchrony. Nat. Neurosci. **20**, 1014–1022 (2017)
M. Pecka, Y. Han, E. Sader, T.D. Mrsic-Flogel, Experience-dependent specialization of receptive field surround for selective coding of natural scenes. Neuron **84**, 457–469 (2014)
D. Plenz, T.C. Thiagarajan, The organizing principles of neuronal avalanches: cell assemblies in the cortex? Trends Neurosci. **30**, 99–110 (2007)

V. Priesemann, M. Wibral, M. Valderrama, R. Pröpper, Q.M. Le Van, T. Geisel, J. Triesch, D. Nikolic, M.H.J. Munk, Spike avalanches *in vivo* suggest a driven, slightly subcritical brain state. Front. Syst. Neurosci. **8**(108), 1–17 (2014)
R. Quian Quiroga, L. Reddy, G. Kreiman, C. Koch, I. Fried, Invariant visual representation by single neurons in the human brain. Nature **435**, 1102–1107 (2005)
M. Rabinovich, A. Volkovskii, P. Lecanda, R. Huerta, H.D.I. Abarbanel, G. Laurent, Dynamical encoding by networks of competing neuron groups: Winnerless competition. Phys. Rev. Lett. **87**(068102), 1–4 (2001)
S. Ray, J.H.R. Maunsell, Differences in gamma frequencies across visual cortex restrict their possible use in computation. Neuron **67**, 885–896 (2010)
S. Ray, J.H.R. Maunsell, Do gamma oscillations play a role in cerebral cortex? Trends Cogn. Sci. **19**, 78–85 (2015)
D.V.R. Reddy, A. Sen, G.L. Johnston, Time delay induced death in coupled limit cycle oscillators. Phys. Rev. Lett. **80**, 5109–5112 (1998)
R.L. Redondo, R.G.M. Morris, Making memories last: the synaptic tagging and capture hypothesis. Nat. Rev. Neurosci. **12**, 17–30 (2011)
P. Reinagel, R.C. Reid, Precise firing events are conserved across neurons. J. Neurosci. **22**, 6837–6841 (2002)
P.R. Roelfsema, A.K. Engel, P. König, W. Singer, Visuomotor integration is associated with zero time-lag synchronization among cortical areas. Nature **385**, 157–161 (1997)
F. Rosenblatt, The perceptron. A probabilistic model for information storage and organization in the brain. Psychol. Rev. **65**, 386–408 (1958)
E. Salinas, T.J. Sejnowski, Impact of correlated synaptic input on output firing rate and variability in simple neuronal models. J. Neurosci. **20**, 6193–6209 (2000)
W. Schultz, Dopamine reward prediction-error signalling: a two-component response. Nat. Rev. Neurosci. **17**, 183–195 (2016)
W. Schultz, P. Dayan, P.R. Montague, A neural substrate of prediction and reward. Science **275**, 1593–1599 (1997)
M. Siegel, T.H. Donner, R. Oostenveld, P. Fries, A.K. Engel, Neuronal synchronization along the dorsal visual pathway reflects the focus of spatial attention. Neuron **60**, 709–719 (2008)
M. Siegel, T.J. Buschman, E.K. Miller, Cortical information flow during flexible sensorimotor decisions. Science **348**, 1352–1355 (2015)
D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, Mastering the game of Go without human knowledge. Nature **550**, 354–359 (2017)
D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, D. Hassabis, A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science **362**, 1140–1144 (2018)
W. Singer, Synchronization of cortical activity and its putative role in information processing and learning. Annu. Rev. Physiol. **55**, 349–374 (1993)
W. Singer, Development and plasticity of cortical processing architectures. Science **270**, 758–764 (1995)
W. Singer, Neuronal synchrony: a versatile code for the definition of relations? Neuron **24**, 49–65 (1999)
W. Singer, Synchronous oscillations and memory formation, in *Learning Theory and Behavior* ed. by J.H. Byrne. Learning and Memory: A Comprehensive Reference, vol. 1, 2nd edn. (Academic, Oxford, 2017), pp 591–597
W. Singer, C.M. Gray, Visual feature integration and the temporal correlation hypothesis. Annu. Rev. Neurosci. **18**, 555–586 (1995)
W. Singer, F. Tretter, Unusually large receptive fields in cats with restricted visual experience. Exp. Brain Res. **26**, 171–184 (1976)
G.B. Smith, A. Sederberg, Y.M. Elyada, S.D. Van Hooser, M. Kaschube, D. Fitzpatrick, The development of cortical circuits for motion discrimination. Nat. Neurosci. **18**, 252–261 (2015)

G.B. Smith, B. Hein, D.E. Whitney, D. Fitzpatrick, M. Kaschube, Distributed network interactions and their emergence in developing neocortex. Nat. Neurosci. **21**, 1600–1608 (2018)
M.C. Soriano, J. Garcia-Ojalvo, C.R. Mirasso, I. Fischer, Complex photonics: Dynamics and applications of delay-coupled semiconductors lasers. Rev. Mod. Phys. **85**, 421–470 (2013)
D.D. Stettler, A. Das, J. Bennett, C.D. Gilbert, Lateral connectivity and contextual interactions in macaque primary visual cortex. Neuron **36**, 739–750 (2002)
D.Y. Tsao, W.A. Freiwald, R.B.H. Tootell, M.S. Livingstone, A cortical region consisting entirely of face-selective cells. Science **311**, 670–674 (2006)
G.G. Turrigiano, S.B. Nelson, Homeostatic plasticity in the developing nervous system. Nat. Rev. Neurosci. **5**, 97–107 (2004)
R. Van Rullen, A. Delmore, S. Thorpe, Feed-forward contour integration in primary visual cortex based on asynchronous spike propagation. Neurocomputing **38**, 1003–1009 (2001)
R. Van Rullen, R. Guyonneau, S.J. Thorpe, Spike times make sense. Trends Neurosci. **28**, 1–4 (2005)
R. Vicente, L.L. Gollo, C.R. Mirasso, I. Fischer, G. Pipa, Dynamical relaying can yield zero time lag neuronal synchrony despite long conduction delays. Proc. Natl. Acad. Sci. U.S.A. **105**, 17157–17162 (2008)
M. Vinck, C.A. Bosman, More gamma more predictions: Gamma-synchronization as a key mechanism for efficient integration of classical receptive field inputs with surround predictions. Front. Syst. Neurosci. **10**(35), 1–27 (2016)
C. von der Malsburg, J. Buhmann, Sensory segmentation with coupled neural oscillators. Biol. Cybern. **67**, 233–242 (1992)
H. von Helmholtz, *Handbuch der Physiologischen Opt.* (Leopold Voss Verlag, Hamburg, 1867)
M.A. Whittington, R.D. Traub, N. Kopell, B. Ermentrout, E.H. Buhl, Inhibition-based rhythms: experimental and mathematical observations on network dynamics. Intern. J. Psychophysiol. **38**, 315–336 (2000)
A.T. Winfree, Biological rhythms and the behavior of populations of coupled oscillators. J. Theor. Biol. **16**, 15–42 (1967)
J. Yamamoto, J. Suh, D. Takeuchi, S. Tonegawa, Successful execution of working memory linked to synchronized high-frequency gamma oscillations. Cell **157**, 845–857 (2014)
O. Yizhar, L.E. Fenno, M. Prigge, F. Schneider, T.J. Davidson, D.J. O'Shea, V.S. Sohal, I. Goshen, J. Finkelstein, J.T. Paz, K. Stehfest, R. Fudim, C. Ramakrishnan, J.R. Huguenard, P. Hegemann, K. Deisseroth, Neocortical excitation/inhibition balance in information processing and social dysfunction. Nature **477**, 171–178 (2011)

Cortico-Striatal Origins of Reservoir Computing, Mixed Selectivity, and Higher Cognitive Function

Peter Ford Dominey

Abstract The computational richness and complexity of recurrent neural networks is well known, and has yet to be fully exploited and understood. It is interesting that in this context, one of the most prevalent features of the cerebral cortex is its massive recurrent connectivity. Despite this central principle of cortical organization, it is only slowly becoming recognized that the cortex is a reservoir. Of course there are mechanisms in the cortex that allow for plasticity. But the general model of a reservoir as a recurrent network that creates a high dimensional temporal expansion of its inputs which can then be harvested for extracting the required output is fully achieved by the cortex. Future research will find this obvious. This chapter provides a framework for more clearly understanding this conception of cortex and the corticostriatal system.

1 Introduction

As Jaeger and colleague note (Lukosevicius and Jaeger 2009, Lukoševičius et al. 2012) reservoir computing was independently invented three times, by Dominey in the context of the primate prefrontal cortex (Dominey 1995; Dominey et al. 1995), by Maass as the liquid state machine (Maass et al. 2002), and by Jaeger (Jaeger 2001, Jaeger and Haas 2004) as the echo state network. This chapter reviews the origins of reservoir computing in the context of the neurophysiology of the primate cortex and striatum—the corticostriatal system. It further describes how this modeling, initially developed to simulate detailed electrophysiology of the primate cortex, went on to simulate human behavior in a number of sensorimotor sequencing tasks, language comprehension and production, and even narrative processing. The link back to primate neurophysiology is made, based on the subsequent characterization of a highly prevalent feature of neural coding in the primate cortex, called mixed selectivity. This is the observation that when agents perform reasonably complex tasks that have different internal and externally driven states, the single unit responses

P. F. Dominey (✉)
INSERM UMR1093-CAPS, Université Bourgogne Franche-Comté, UFR des Sciences du Sport, F-21000 Dijon, France
e-mail: peter.dominey@inserm.fr

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_2

of individual neurons encode non-linear mixtures of these states, thus called mixed selectivity (Rigotti et al. 2013).

One of the most prevalent characteristics of cortical neuroanatomy is the overwhelming abundance of very short-distance, local, recurrent connections between pyramidal cells and inhibitory interneurons (Markov et al. 2011). While we are so accustomed to seeing this, it didn't have to be that way. Pyramidal cells might have received their thalamic inputs and kept to themselves, or communicated only over long-distance connections. But instead their largest preference is to communicate with their local neighbors. This must be for something, and that is what we shall describe here. Part of the clue is the overwhelming aspect of sequential or temporal structure in our existence. In the rest of this section, we briefly review the neurophysiological and behavioral phenomena that motivated one flavor of reservoir computing.

1.1 Initial Motivation—Barone and Joseph

In the 1980s, there was a massive activity and success in characterizing the functional neuroanatomy of the interactions between cortex and basal ganglia—the corticostriatal system—particularly in the domain of fast eye movements, the oculomotor saccade. Regions of the peri-arcuate frontal cortex including the frontal eye fields were carefully studied as non-human primates (macaque monkeys) made saccades to visual targets, and memorized targets, revealing properties related to the visual targets, memory of the target, the saccade itself (Bruce & Goldberg 1985, Goldberg and Bushnell 1981). Similarly, the role of the basal ganglia—the caudate nucleus of the striatum—the input node (Hikosaka 1989, Hikosaka et al. 1989a, b, c), and the substantia nigra pars reticulata (Hikosaka and Wurtz 1983a, b, c, d), relay to the output structure, the superior colliculus in all aspects of saccades—visual response, delayed memory response, and saccade response had been very well characterized. One of the characteristic coding properties observed throughout the cortex, striatum, SNr, and superior colliculus was a topography of saccade amplitude and direction. That is, neurons coded for a particular saccade amplitude and direction, and their neighbors coded for similar metrics. Because the retinal coding and the saccade response to the target overlap, this coding can be considered in terms of a spatial receptive field, a motor response field, and a retinal error. We developed the first complete model of how the cortex and basal ganglia cooperate to produce a variety of oculomotor saccade behavior that provided a coherent framework for the interpretation of this neurophysiological data (Dominey and Arbib 1992).

During this period researchers in Lyon France took what was at that time a very radical step, and examined the role of the corticostriatal system when monkeys were trained to learn and reproduce spatio-temporal sequences of saccades and manual button presses to the spatial sequence elements. In this pioneering work, Barone and Joseph discovered a fascinating coding of space and sequential order (Barone and Joseph 1989). That is, certain neurons had a spatial preference that was modulated

by the sequential order. Such coding began to provide clues on how the underlying system might work. The drive to understand this behavioral neurophysiology was the principal motivation for the modeling work that led to one branch of reservoir computing.

1.2 Further Motivation—Temporal Structure

Another clue on how the system works came from additional studies of sequence learning, this time in humans. In this domain of sequential behavior, a number of human studies revealed the importance of temporal structure, or rhythm, in the identity of a behavioral sequence. A classic manifestation of this was observed in the serial reaction time task, where subjects were asked to respond to stimuli and their reaction times were measured. Unknown to subjects, the stimuli could be ordered in more or less complex sequences, and subjects reliably displayed robust learning of these sequences (Stadler 1993). Interestingly, the temporal structure of delays imposed between successive stimuli was learned as an integral part of the sequence, laying another requirement on models of sequence learning.

1.3 Abstract Structure and Language

Looking toward the origins of these capacities for sequence learning, developmental psychologist developed clever methods to measure sequence learning in infants as young as nine months old. Saffran et al. (1996) demonstrated that within minutes, these infants could learn the serial order of sound sequences. Nazzi et al. (1998) tested nine-month-old children's ability to discriminate temporal structure of different language rhythm classes, and again observed a robust ability for discriminating rhythm classes in these children. Finally, Marcus et al. (1999) tested seven-month-old infant's ability to learn abstract patterns of sounds like we-ga-ga or ba-ni-ni that correspond to an abstract rule ABB. They demonstrated that remarkably, the children could then use this learning to discriminate whether new sequences followed an ABB vs ABA rule. This was exciting evidence that pre-linguistic infants mastered a form of abstract generative rule, that could be used in the service of language learning. This again provides further requirements on a neurophysiological system for sensorimotor sequence learning.

1.4 Renaissance: Neural Dynamics, Mixed Selectivity, and Higher Cognitive Function

A final piece of the puzzle of reservoir computing has recently arisen in studies of primate physiology that extend what was initially observed in Barone and Joseph (1989). In particular, Rigotti et al. (2013) clearly established the link between this multidimensional coding of different dimensions of cognitive task execution, and the ability of the primates to perform the task. As will be developed below, this mixed selectivity is a signature of the neural dynamics of reservoir computing (Enel et al. 2016, Rigotti et al. 2013).

Given this brief overview of some of the motivations for developing a brain-inspired sequence learning system, we now take the next step in characterizing the genesis of the model.

2 Corticostriatal Foundations

The primate corticostriatal system consists in two of the most remarkable aspects of the primate brain. The first is the undeniable prevalence of local recurrent connections in the entire cortex, and the second is the massive projection from cortex to the sizable sub-cortical structure, the striatum, the input node of the basal ganglia. Here, we begin to examine the neuroanatomy, neurophysiology, and resulting behavior of this system that lead to the development of one branch of reservoir computing.

2.1 Barone and Joseph, and the Primate Behavioral Neurophysiology of Sequence Learning

In the heyday of the exploration of the corticostriatal oculomotor system, Barone and Joseph (1989) recorded neurons in the frontal cortex, around the arcuate sulcus, near the frontal eye fields where Goldberg and others found saccade related activity. But they asked a daring new question. Barone and Joseph trained their primates to observe different spatial sequences of three successively presented spatial targets on an array in front of the animal. During the static phase of the trial, the animal was required to maintain the gaze on a central fixation point. Then, in the dynamic phase, the removal of the fixation point, and simultaneous onset of all three targets was the signal for the animal to make a saccade to the first target, and then touch it. Successive presentation of the remaining targets was the go-signal for the second and final saccades of the three-element sequence. The animals underwent several months of training in order to learn to perform this task. Their behavior was successively shaped: First making saccades to a single element, then sequences of two element, and finally three elements. After the behavior was attained the neural activity was

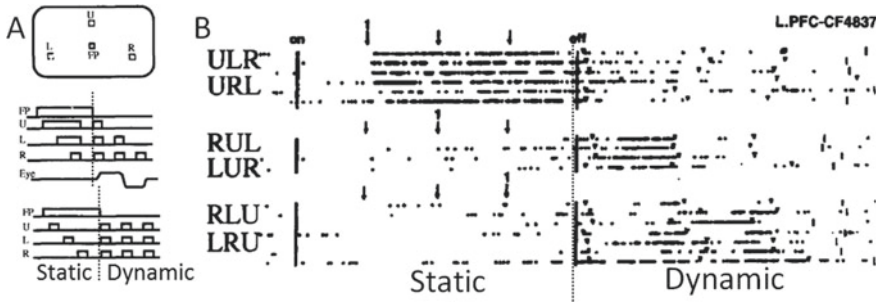


Fig. 1 Behavior and neurophysiology in the sequential saccade task. **A** Upper panel illustrates three peripheral visual target buttons L, U and R, and the central fixation point. Six possible three-element sequences were used. Second and third left panels illustrate the temporal success of these stimuli in the static and dynamic phases. During the static phase, the animal must maintain gaze on the fixation point. During the dynamic phase, the successive temporal onset of the targets signals the animal to saccade and then touch the first, second, and third element of the memorized sequence. **B** Visual-tonic cell. Traces of action potentials (spikes, neural firing) of a neuron that responded during the static phase with a spatial selectivity for the upper target, but only when it was first in the sequence

recorded, using single electrodes that penetrated into the cortex around the arcuate sulcus.

Half of the 300 cells that they recorded displayed a complex combination of spatial and sequential dimensions of the task. Visual tonic cells showed a visual response to a given target location in the static phase. However, this spatial selectivity was modulated by the rank of that element in the sequence. This is a form of mixed selectivity (Rigotti et al. 2013) that mixes spatial location and rank in the sequence during the static phase. Such a unit is illustrated in Fig. 1. Another particularly interesting type of response was seen in Context cells which responded to a saccade to a given target (L, U or R) in the dynamic phase, but in a way that was dependent on where it occurred in the sequence. Such neurons thus displayed a form of mixed selectivity, mixing spatial location and saccade rank. One can imagine how such units might be used in a population code to determine the next element in the ongoing sequence.

2.2 Requirements on the System—Mixture of Inhibition and Excitation, Modulated by Time

How can we imagine the wiring of a neural network so as to allow the network to perform the task, and to generate these neural responses that mix spatial location and sequence rank in doing so? In order to produce neurons with a response like the visual tonic cell seen in Fig. 1, we would imagine that neurons should receive excitatory retinotopic or spatially selective input (for spatial selectivity), and inhibitory input

from other spatially selective neurons (for rank selectivity). In Fig. 1, this neuron would get excitatory input for location U, and inhibitory input from neurons that themselves get excitatory inputs from L and R. Prewiring such neurons could be done for a given task, but there must be a more general architectural solution that can yield these results.

2.3 Primate Cortical Neuroanatomy

The presence of spatial selectivity in these frontal oculomotor areas is well documented, and is likely due to spatially organized inputs from the parietal cortex in the dorsal stream (Bruce and Goldberg 1984). This can explain the spatially selective responses here, but what about the context effects? If we look at cortical neuroanatomy, one of the most prevalent features is the existence of local short-range connections between neighboring neurons (Goldman-Rakic 1987; Markov et al. 2011). Via these local recurrent connections, neurons with a visual tonic response to a given spatial target could project to a local neighbor that has a spatial response to a different target location. If this projection is inhibitory (via an intervening inhibitory interneuron), then we have a circuit to produce these context or mixed selective effects. Here, we have the foundation then for the notion that cortex can be modeled as a population of neurons that receive excitatory input from an external source, and that has local recurrent excitatory and inhibitory connections.

2.4 Primate Corticostriatal System—Plasticity

The question remains, how could representations generated in this cortical network be used for sequence learning? The first part of the response to this question is derived from what is perhaps the second most prevalent aspect of cortical neuroanatomy, which is the massive projection from all of cortex to the striatum, which is the input nucleus of the basal ganglia (Selemon and Goldman-Rakic 1985). That is, all of the cerebral cortex projects in an orderly way onto the striatum. What's more, there is significant overlap and intermingling of the projections of different cortical regions into the striatum. Alexander et al. (1986) characterized how this massive corticostriatal system can be divided into a set of segregated (but partially overlapping)) pathways dedicated to sensorimotor and cognitive functions, based on the different corticostriatal topography. Of particular interest to us, the saccade-related neurons in the frontal eye fields, and neurons in the dorsolateral prefrontal cortex (as recorded by Barone and Joseph) project to overlapping zones in the anterior part of the caudate nucleus of the striatum (Selemon and Goldman-Rakic 1988). This means that sequence-related activity in the cortex can project onto oculomotor saccade output structures in the striatum, thus binding sequence context onto saccade motor behavior.

The final element in this puzzle comes from one of the major neurophysiological properties of the striatum, which is that it is a site for major plasticity, under the control of reward-related dopamine (Ljungberg et al. 1992; Schultz et al. 1993). Indeed, under the control of dopamine, corticostriatal synapses are candidates for long-term depression and long-term potentiation (Calabresi et al. 2007; Centonze et al. 2001; Di Filippo et al. 2009). That is, through reward-related learning, the system can learn by trial and error to associate the dynamic state of the recurrent prefrontal network with the appropriate output in the caudate nucleus of the striatum.

3 Corticostriatal Reservoir Computing

We now have the required ingredients to assemble a neurophysiologically valid model of sequence learning that should be able to learn the Barone and Joseph sequencing task, while at the same time generating and exploiting mixed selectivity (manifest as visual-tonic and context activity, as observed in the prefrontal neurons by Barone and Joseph).

3.1 Model Implementation—Birth of the Reservoir

We developed the first model of reservoir computing in Dominey (1995), Dominey et al. (1995), initially in the context of the primate corticostriatal sequence learning physiology. The model builds on the initial model of the corticostriatal saccade system (Dominey and Arbib 1992), by introducing the recurrent PFC system (the reservoir), and the reward-related learning in PFC-caudate connections (the readout). Illustrated in Fig. 2, individual cortical and sub-cortical layers were implemented as 5×5 arrays of leaky integrator neurons with sigmoidal output functions.

In the oculomotor circuit, visuospatial input enters the system and activates the posterior parietal cortex which projects with a topographic input to the oculomotor frontal eye fields (FEF). FEF has an excitatory projection to the caudate, and the output system, the Superior Colliculus. In parallel, the caudate has a topographic inhibitory projections to the substantia nigra pars reticulate (SNr), which itself has a tonic inhibitor activation on the Superior Colliculus (Chevalier and Deniau 1990). In a standard saccade, FEF activation activates Caudate, which dis-inhibits the tonic inhibition of SNr on Superior Colliculus. This removal of inhibition, combined with the excitatory input from FEF, allows the Superior Colliculus to activate and generate the desired saccade.

Reservoir Genesis: This classic oculomotor circuit was then enhanced with the ability to make conditional behavior—to choose the correct saccade target from among several based on learning. We first studied how different visual cues could be associated with different targets, and we then studied how spatial sequences could be associated with output saccade sequences, which required the Prefrontal cortex

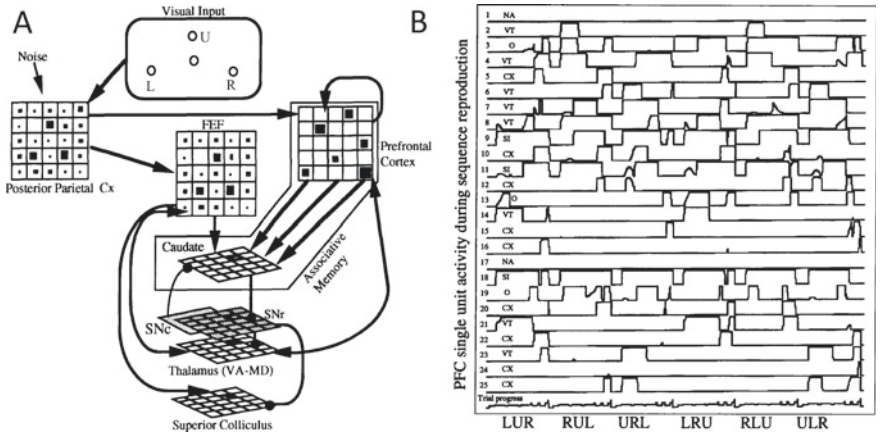


Fig. 2 **A** Corticostriatal model for sequence learning. Classic oculomotor circuit. Spatial input to Posterior Parietal Cortex, FEF, Caudate, with disinhibition of Superior Colliculus via the dual inhibition of Caudate and SNr, and generation of saccade in Superior Colliculus. Prefrontal sequencing circuit (reservoir). In parallel, sequence state coding in the recurrent network projects to Caudate. Through trial and error learning, at each saccade choice time, activation patterns in Prefrontal Cortex become associated with the correct saccade choice at that time, via reward-related dopamine in the Caudate that strengthens Prefrontal Cortex—Caudate connections (Modified from Dominey et al. 1995). **B** PFC neuron activity during static and dynamic phase for the six three-element Up, Left, Right sequences used in Barone and Joseph (1989) (Modified from Dominey 1995)

sequencing circuit, the reservoir (Dominey et al. 1995). The Prefrontal Cortex is modeled as recurrent network made of two 5 × 5 layers of neurons. The first, PFC, receives the topographic input from Parietal Cortex, and project topographically to the second, PFC_D which projects back to the first with random and mixed excitatory and inhibitory connections. We had initially considered modeling plasticity in these internal PFC- PFC_D connections. However, a crucial aspect of our modeling was to respect the temporal structure of the primate behavior. Simulation time steps correspond to 5 ms of real-time. The sequencing task takes several seconds, with stimuli presented for hundreds of milliseconds, delays, then onset of the dynamic phase, with several seconds unfolding as the animal chooses the three successive sequences. In simulation, this corresponds to an order of 1000 time steps for a single sequence presentation and recall. Modifying recurrent connections is computationally expensive and non-trivial, if we want to keep track of the role of a recurrent connection over 1000 time steps into the past (Pearlmutter 1995). *We thus choose to keep the recurrent connections fixed, with a mixture of inhibitory and excitatory recurrent connections to provide a rich spatio-temporal dynamics within the system, and to use neural plasticity to bind states of activity in the recurrent network with the desired responses in the output layer, in our case, corresponding to the caudate or striatum (Dominey* 1995*, Dominey* et al. 1995*).* This was the genesis of the first implementation of reservoir computing. The rich dynamics can be seen in Fig. 2B.

The PFC layer projects to Caudate, forming the readout. The PFC-Caudate connections are 1:N and are initialized as random, and they are subject to modification according to a simple learning rule. Each time the model is asked to make a choice (by the go-signal, offset of the central target corresponding to the Fixation Point), when the output activation in the Superior Colliculus reaches a threshold, that output is evaluated. If it corresponds to the correct choice, the PFC-Caudate connections to the Caudate neuron involved in that choice are strengthened, otherwise they are weakened. Then, weight normalization is performed to conserve the total amount of synaptic weight in the PFC-Caudate weight matrix. This synaptic plasticity in the PFC-Caudate connections is perfectly justified by synaptic plasticity under the control of reward-related dopamine (Calabresi et al. 2007; Centonze et al. 2001; Di Filippo et al. 2009).

3.2 Modeling Barone and Joseph (1989)

The model is trained in a progressive manner. Each trial consists of a static phase where the input sequence is presented and the model generates no output. During this phase, however, the PFC is driven through a stimulus- driven state activation trajectory, which is unique for each input sequence. Then during the dynamic phase, the inhibitory fixation point is removed, the three targets are presented and the model must chose the correct response, with a positive synaptic change on the PFC-Caudate weights for the Caudate neuron corresponding to the choice if the choice is correct and negative synaptic change if incorrect. During initial training, only the correct target is presented at each go-signal, which allows an initial shaping of the correct behavior, then in subsequent epochs, all three targets are presented and the model must choose. Figure 2B illustrates the trajectory of activation of the PFC neurons during successive trials with each of the six target sequences.

We thus observed that the model could learn the sequencing task, and perform like the trained primates. We also saw that there was rich activity in the recurrent PFC network. But what remained, most interesting, was to perform a more detailed analysis of the activity patterns of individual neurons in the recurrent network, and compare them to those observed in the primate. Indeed, this approach is specific to these studies of reservoir computing in the context of understanding the neurophysiology of the primate cortex. We analyzed the data in Fig. 2 and categorized cells as Context, Visual-tonic, signal related, and other. Interestingly, we found the same proportions of these response types as found in the study of Barone and Joseph. Figure 3 illustrates typical examples of Visual-Tonic and Context cells from the primate study of Barone and Joseph and comparison with examples of cell types from our model. Examining the comparison in detail, it is actually quite striking how similar the activation patterns of the real and simulated neurons are.

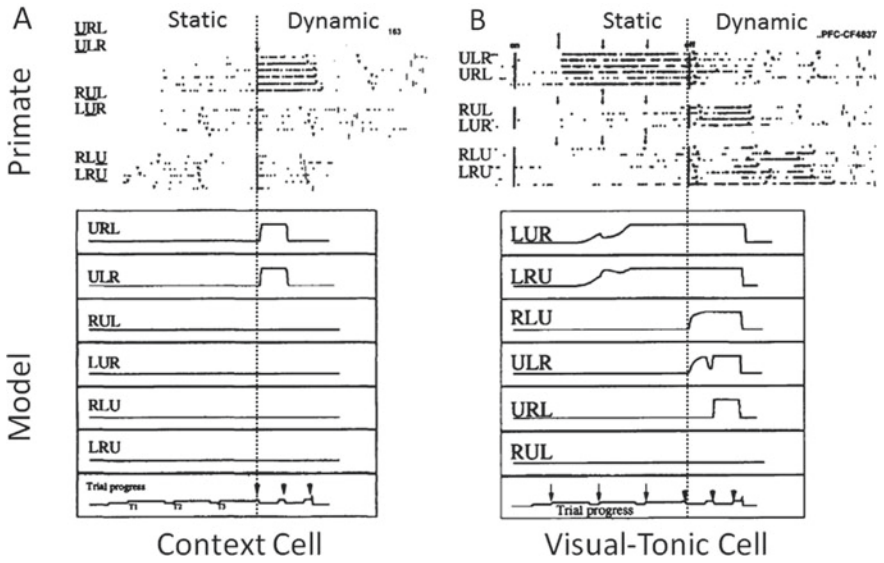


Fig. 3 Comparison of primate and model PFC neural activity. **A** Context cell. These cells respond after the saccade to a particular target, but depend on its rank in the target. Primate and model neurons that respond after saccades to the upper (U) target, but only when it is first in the sequence. **B** Visual-Tonic cells display a response to the visual presentation of a target, but depend on its location in the sequences. Primate and model neurons respond after a given target (U for the primate and L for the model) but only when it is first in the sequence. In both primate and model there is complex activity later in the dynamic phase

3.3 Complex Sequence Learning

Motivated by the success of the model in learning the six sequences, performing like the monkey, and generating neural activity as seen in the primate cortex, we then set out to further explore the sequence learning capabilities of the system.

Figure 4 illustrates the activation in the PFC reservoir during the presentation of a complex sequence ABCDABCEABCFABCGABCH that has a repeating subsequence (ABC) with different successive successors D, E, F, G, and H. This is a challenging sequence because it requires a memory of the past 4 elements to determine the correct choice after ambiguous element C. Observing the last row in panel 4B, where the activation vector for the final choice H is compared to all other states, we see that the cosine is near 1 or D, E, F and G, all of which had the same repeating subsequence that preceded, but that each had a different unique element prior to that subsequence. This indicates that the reservoir has sufficient memory to distinguish what happened each time before the intervening repeating subsequence ABC.

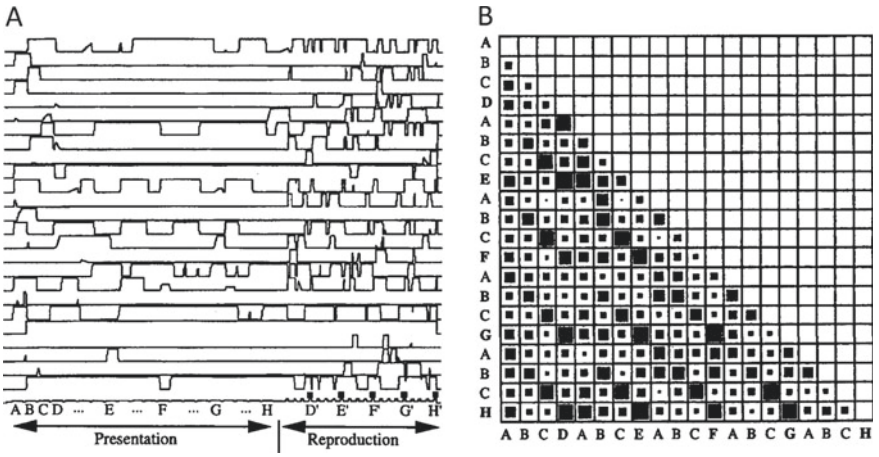


Fig. 4 **A** Reservoir activity in during presentation and execution of a complex sequence. **B** Similarity matrix comparing the cosines of activation vectors at each choice epoch in the reproduction phase. Strong filling indicates cosine near 1, corresponding to similar (and potentially confusing) state vectors

3.4 Temporal Structure

In the initial development of the reservoir (Dominey 1995; Dominey et al. 1995), a deliberate choice was made to use fixed connections in the recurrent network, so as to provide rich dynamics that could represent the spatio-temporal structure of arbitrary problems. The motivation was the difficulty of applying learning rules to the recurrent connections, which often involve a cut-off of the temporal sensitivity (Pearlmutter 1995). Thus, one of the most robust and inherent characteristics of the resulting network is its inherent sensitivity to time, delays, and temporal structure. This can be observed by stimulating the recurrent network with a brief input, and then observing the resulting state trajectory in the absence of further external input. An example of this can be seen in Fig. 5A. There, we can observe the reliable and reproducible state trajectory following a fixed input that could be used to discriminate temporal delays. The system was trained to respond with different outputs according to which one of three delays was imposed following the same initial input, before the go-signal to respond was provided. This was a first form of interesting temporal behavior demonstrated by the network (Dominey 1998b).

A second form of temporal behavior is revealed by reduced reaction times for responses to sequentially presented stimuli, due to increased excitation with significant learning. We recall that all of the neurons in the system are leaky integrators with sigmoid output functions. The leaky integrator must charge to reach firing threshold, and thus, the delay in this activation is a function of the strength of the inputs. After significant learning, the PFC-caudate connections become increased and lead to a faster activation of the caudate neurons, resulting in reduced reaction times in the

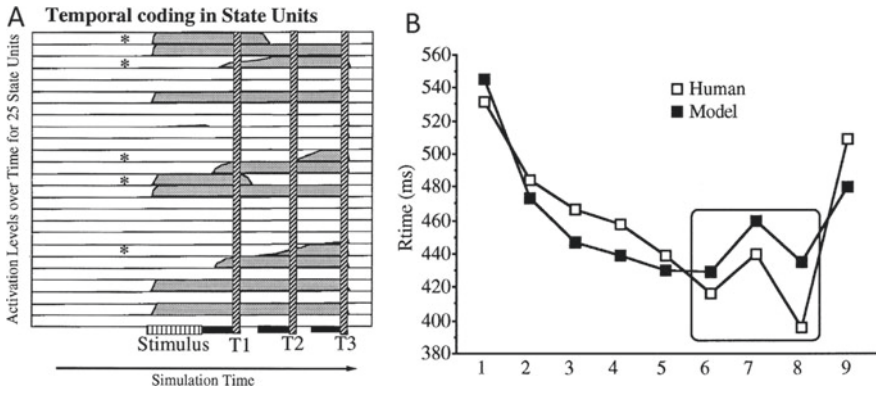


Fig. 5 Temporal Structure coding in the dynamic reservoir. **A** Activity in 25 State units (half of the reservoir population of 50) during and after a short input/perturbation. Prolonged and delayed activity reflects the reservoir dynamics. Different patterns at T1, T2 and T3 are reliably re-produced and thus allow the system to learn to discriminate these different time intervals. **B** Human and model response times in a serial reaction time task. Blocks 1–6 and 8 use a repeating sequence with each input element associated with a specific temporal structure. Block 7 uses the same sequence, with different temporal structure. This increases the response times, as if the sequence itself had been changed. Block 9 uses a different sequence, also producing increased reaction times

output generated by the system, for well-learned behavior. This allowed us to simulate human behavior in tasks where reaction time was a measure of sequence learning (Dominey 1998a, b). The classic paradigm for this is the SRT or serial reaction time task, where subjects are to respond as quickly as possible to stimuli that are presented in typically long (several hundred) blocks of responses. Unknown to the subjects, the stimuli can be organized in more or less complex sequences, and the difference in reaction times for short blocks of trials that follow the learned sequence, vs. in a random order, is a measure of learning. A typical profile is illustrated in Fig. 5B. There, RTs are shown for 6 blocks of 80 trials with the repeating 10-trial sequence B–C–B–D–C–A–D–A–C–D using temporal structure where elements A and D are always preceded by RSIs of 200 ms and B and C by RSIs of 1000 ms. In block 7 the same sequence is used, and the temporal structure is modified by swapping the 200 and 1000 ms delays.

What we observe, both for humans and the temporal recurrent network model, is that the sequence is learned as a multimodal structure combining serial and temporal structure. This is a demonstration that the temporal recurrent network (TRN) inherently possesses this sensitivity to serial and temporal structure, due to the dynamic network properties.

3.5 Abstract Structure

This allowed us to conclude that the TRN was inherently sensitive to serial and temporal structure. In the late 1990s, efforts were being made to assess human infants sensitivity to such dimensions of sequential structure. Clever behavioral methods were adopted to assess sequence learning in infants as young as 7–9 months of age, by measuring an increase in their rate of sucking on a pacifier in response to novel stimuli. Thus, the infants could be habituated with audio sequences of a particular type, and then presented with test sequences that either adhered to or varied from the habituated sequences, and the behavioral response in case of a changed sequence was evidence that the infant had learned (Jusczyk 1997). It was thus demonstrated by Saffran et al. (1996) that 8 month olds could learn sequential structure of simple 3 element sound sequences, and by Nazzi et al. (1998) that they were sensitive to the temporal rhythmic structure of different language rhythm classes. We adapted these tasks to the TRN and demonstrated indeed that the model could reproduce the learning effects that were observed in the children (Dominey and Ramus 2000). Interestingly, however, there was another dimension of sequential structure that the TRN model failed to capture.

In a protocol similar to that used by Saffran, Marcus et al. (1999) generated syllable sequences according to simple rules ABA vs ABB, and then tested whether the infants could discriminate new sequences as adhering to the learned rules. Thus, in the test phase, the infant would be exposed to sound sequences that it had never heard before, but that did (or did not) adhere to the rule that they had learned. Intuitively, we anticipated that the TRN would fail in this task, because there is no mechanism or representation that can detect the structure of repetition that characterizes the difference between ABA and ABB. The TRN can learn sequence identity but not these underlying rules. In order to account for such behavior, the system would require some form of working memory and a recognition function to compare the current element with previous elements in a sequence. We implemented such a mechanism that augmented the TRN, thus resulting in a dual model with TRN and ANR for Abstract Recurrent Network (Dominey et al. 2003), that we referred to as the Abstract Temporal Recurrent network (ATRN) illustrated in Fig. 6.

Using the model illustrated in Fig. 6, we tested the learning of sequences like ABCBAC, and then tested with sequences like DEFEDF. The STM and recognition mechanisms perceived both of these sequences as having a structure u u u n-2 n-4 n-3, where u indicates that the current element is unrecognized with respect to the contents of the STM, and n-x indicates that it is a repetition of the element n-places back in the STM. Thus, in terms of the abstract structure, the two sequences ABCBAC and DEFEDF are the same. This allows the system to explain and simulate the results of abstract structure learning as observed by Marcus. We performed a number of human experiments and finally concluded that there are dissociable neurophysiological systems (corresponding to the TRN and ARN) for learning the surface and abstract structure of sensorimotor sequences (Dominey et al. 1998).

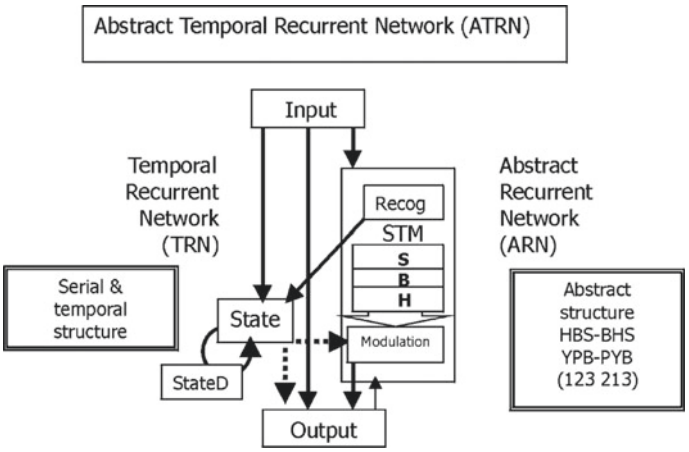


Fig. 6 Combined Temporal and Abstract Recurrent Network. In the ARN, sequence elements are stored in a short-term memory that always contains the last 5 elements seen. This is compared with current element, in order to detect repetition structure, as in 123–213. This recognition (abstract) structure is input to the recurrent State-StateD network, which now can learn the serial order, or the abstract structure, or both

4 Higher Cognitive Function I—Language

4.1 Language Acquisition and Grammatical Constructions

Given the ability to model infant's behavior in learning serial, temporal, and abstract structure of language-like sound sequences (Dominey and Ramus 2000), we felt that these precursors to language learning could be exploited in the simulation of some aspects of language acquisition. The question was how to map language processing into this domain of sequence learning. In response to this we identified a behavioral paradigm for testing language in aphasic patients, developed by (Caplan et al. 1985) that was well suited. In this thematic role assignment task, the patients heard a sentence, and then they had to identify (by pointing to pictures) the agent, object, and recipient of the action described in the sentence, in that order. Thus, this language comprehension task had a sequence processing aspect: the sentence was presented as a sequence of words, then the response was the sequence agent, object, recipient (the thematic roles) in that order. The challenge was to correctly extract these thematic roles, which did not necessarily map on to a fixed order in the sentence. In active sentences like "The giraffe gave the elephant to the monkey", the nouns giraffe, elephant and monkey appear already in the required agent, object, recipient order. However, in sentences like "The elephant was given to the monkey by the giraffe", the nouns came in the wrong order, and the participant had to use the grammatical structure of the sentence to determine the thematic roles. We translated this protocol

into a sequence learning task, where given an input sequence, the system had to produce the nouns in the order Agent, Object, Recipient. Here are two examples.

Example 1 Input: The A was given to the B by the C Response: C A B Rule: ABC-CAB

Example 2 Input: A gave B to C Response: A B C Rule: ABC-ABC

Looking at Examples 1 and 2, the inputs are different. In particular, the presence and order of the closed class elements was, to and by is different for these two sentences, and can be used to distinguish between them.

The solution is to send the closed class words to the TRN while the repetitive structure of the open class words is detected and encoded in the ARN. This way the structure of the closed class words forms a pattern that can be associated with the abstract rule. Interestingly, this approach corresponds to a well-validated hypothesis of linguistic development. The competition hypothesis holds that, across languages, different cues like word order and grammatical marking, are used to encode thematic roles (Bates and MacWhinney 1987; Li and Macwhinney 2013). While we initially had a very clear idea about the neuroanatomy of the corticostriatal system for the core of the reservoir, the mapping of the additional elements of the language model onto the neurophysiology required some effort, and the result is revealed in Fig. 7. This neuroanatomical reality was informed by the proposed models of Friederici and Hagoort (Friederici 2002, 2012; Hagoort 2005).

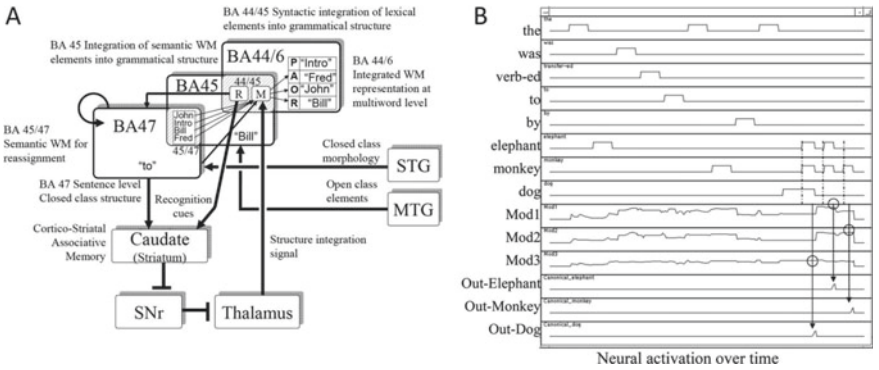


Fig. 7 Here, we map the elements of the reservoir and abstract structure processing onto human neurophysiology. **A** Grammatical function words and closed class morphology enter from superior temporal gyrus and enter into the recurrent reservoir in BA47. Open class elements from middle temporal gyrus enter a semantic working memory in BA45. The grammatical construction that is uniquely identified by the closed-class driven pattern of activity in BA47 modulates the open class working memory of BA45 into the structured meaning representation in BA44/46. **B** Display of main functional elements of the model during processing of the sentence "The elephant was given to the monkey by the dog." Note the Mod1, 2, and 3 neurons that modulate their corresponding working memory elements, which generates the output sequence "dog, elephant, monkey" corresponding to the agent object and recipient of the sentence

In Fig. 7, the TRN function is carried out by STG—BA47 pathway where closed class grammatical structure is processed in the recurrent network. The ARN with its short-term memory and matching function is carried out by the MTG—BA45 pathway for open class elements.

It is noteworthy that this proposed hybrid system is able to perform the described language processing capabilities, but in addition, it can be used to perform non-linguistic tasks that require the manipulation of abstract structure such as artificial grammar learning (Dominey and Inui 2009; Dominey et al. 2009).

It is remarkable that the proposed model was able to perform thematic role assignment in natural language, and also able to learn non-linguistic abstract sequencing tasks. This means that from the limited perspective of the model, language has the same processing status as that of any cognitive sequence system in which configurations of function elements govern the application of systematic transformations, as in domains including music, mathematics, logical theorem proving and artificial symbol manipulation tasks (Dominey et al. 2003). This predicts that (1) similar brain activation should be observed when humans process linguistic and non-linguistic abstract structure, and (2) impairments that affect abstract structure in language should equally affect it in non-linguistic processing. These predictions were confirmed using ERP (Hoen and Dominey 2000), and fMRI (Hoen et al. 2006) in healthy subjects, and by observing correlated impairments in aphasic patients for abstract and grammatical structure processing (Dominey et al. 2003; Dominey and Lelekov 2000; Lelekov et al. 2000).

In the context of human neurophysiology, this research allows us to propose an additional circuit in the framework of Alexander et al. (1986), that is a language or grammatical structure circuit (Dominey 2013; Dominey and Inui 2009; Dominey et al. 2009).

4.2 The Reservoir Convergence

Up to this point, we were exploring language learning with the modeling infrastructure that had been developed initially to model the primate cortex, including online, trial by trial learning. In addition, our research team had not yet made the link to the related work of Jaeger and Maass. But this was about to change. As part of the European Information and Communication Technologies project Organic, Jaeger and Maass invited our team to join the proposal. One result of this was that we began to use more powerful simulation tools (Verstraeten et al. 2012), including radically more efficient learning mechanisms for the cortico-striatal readout, which allowed the use of much larger training corpora (Hinaut and Dominey 2013). These corpora were organized as sentence meaning pairs, with sentences coded word by word, and meaning coded by specification of the mapping between the ordered open class words, and their semantic roles (as illustrated in Fig. 8).

In this work with Hinaut, for the first time we thus exploited larger corpora, and advanced training methods that allowed us to demonstrate the ability of the model

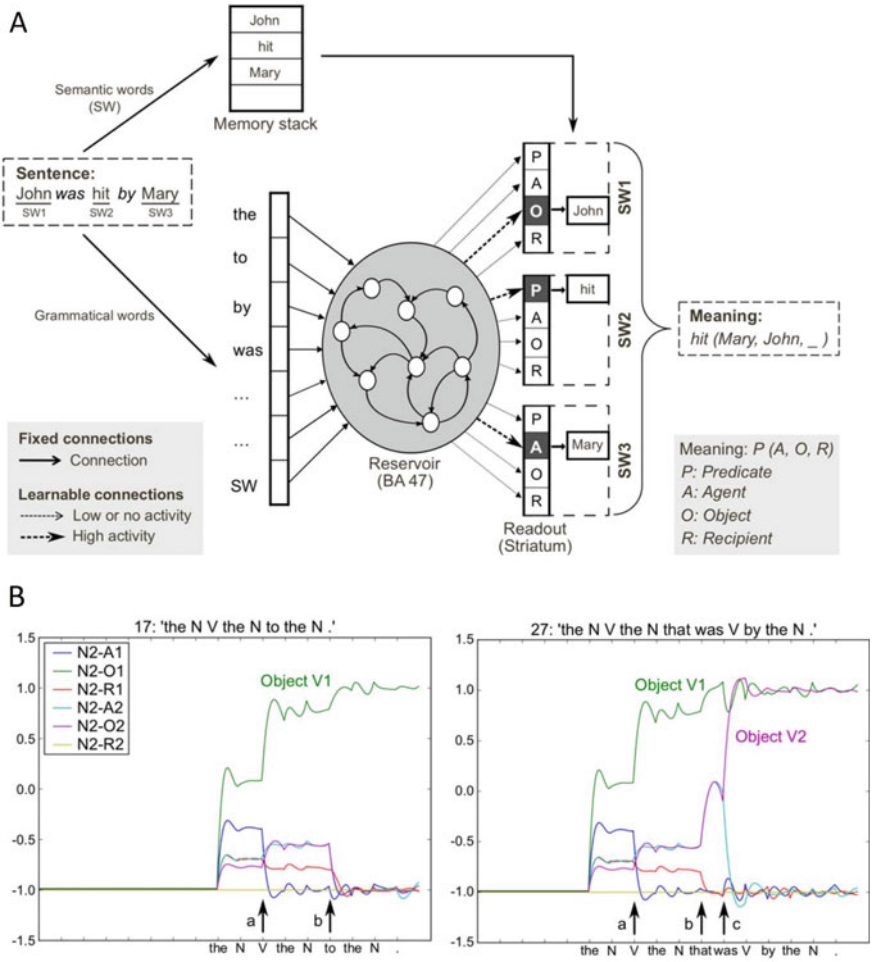


Fig. 8 Language processing reservoir from Hinaut & Dominey (2013). Again, closed class elements activate neurons that project to the recurrent reservoir BA47. **A** Open class elements stored in the BA45 working memory. On the right, a set of four readout neurons per open class element code its potential role as predicate, agent, object, or recipient (there are actually two of these 4-element readouts per open class word, to accommodate sentences with two verbs). **B** Activity of readout neurons coding the role of SW2, the second semantic word in the sentence. Left, for the sentence "The boy gave the book to the teacher" we see that neuron coding for the Object role is active from the outset, that is it predicts the object role, and in this case it is correct. The Right panel illustrates activity for the sentence "The cat chased the boy that was bit by the dog". There we see that the system immediately votes for SW2 to be the object of the first verb. It determines that SW2 is also the object of the second verb but only after the word "was" appears. This reflects the real-time or on-line processing, as well as the parallel processing of multiple possibilities in parallel

to generalize to new constructions that it had not been exposed to before (Hinaut and Dominey 2013). This was a crucial step in the argument that language could be processed by reservoirs, as the ability to generalize to new grammatical structures is a crucial aspect of human language processing. Significantly, the neurophysiological validity of the comprehension model was confirmed in a human clinical evaluation (Szalisznyó et al. 2017). Similarly, we were able to demonstrate a link between the real-time processing of multiple possible sentence parses in the readout neurons of the reservoir, and language-related ERPs (the P600) observed in human sentence processing (Hinaut and Dominey 2013). To complete the grammatical construction story, we applied the mechanism in the opposite sense for language production (Hinaut et al. 2015).

4.3 Narrative

The key idea in the language processing modeling is that the linear string of words contains two forms of information. The basic semantic elements are coded in the semantic or open class words (nouns, verbs, etc.), while the structural relations between these elements (including who did what to whom) is coded by cues including grammatical markers (was, to, by, etc.) and word order, as specified in the cue competition model (Bates and MacWhinney 1987; Li and Macwhinney 2013). This approach has been limited to the grammatical construction. At the level of the construction, grammatical words specify the relations between open class (semantic) elements in one or two events. For example, "The dog that chased the cat bit the boy" and "The dog was chased by the cat that bit the boy" express two quite different meanings that are coded by the grammatical function words.

Considering narrative, we can extend the notion of grammatical construction to that of narrative construction, where narrative function words (before, after, because, since, then, etc.) define relations between open class elements across multiple events in a situation model. This is illustrated in Fig. 9, where the Narrative Cx Model uses separate reservoirs for comprehension and production, and mapping to and from a representation of multiple related events in the Situation Model. The narrative comprehension reservoir is very similar to the sentence comprehension reservoir, but it has two sets of readout neurons, coding events and narrative relations, respectively. The coding in events is as in the sentence model, that is, readout neurons specify the semantic role for each open class element in the sentence. The coding of narrative relations consists of an additional set of readout neurons that encode the narrative function words. In the system developed by Mealier et al. (2017), these narrative relations were extracted and used to create narrative links in the situation model. Thus, in processing the narrated sentence "I gave you the toy because you wanted it", the system creates the causal link labeled "because" between the representations for the given event and the want event, as schematically illustrated above.

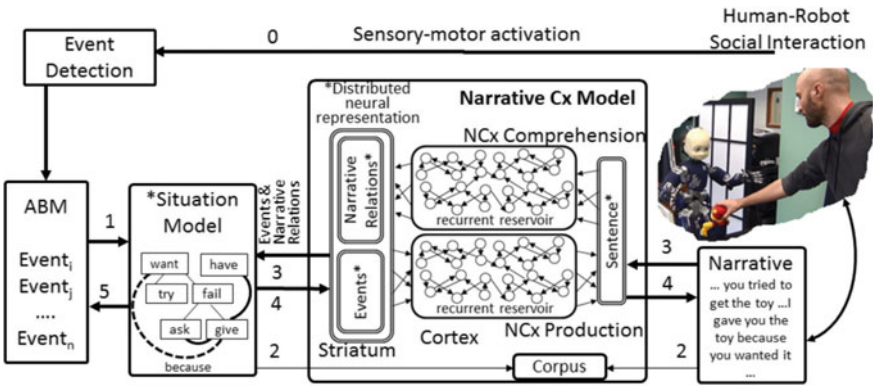


Fig. 9 Human–robot interaction generates events that are stored in the ABM. These are used to generate a graph where events are linked by temporal and causal relations to build the situation model. Human narration is processed by the Narrative Construction (NCx) comprehension model which extracts events and narrative function words. These are automatically mapped onto events in the Situation model, and the narrative relations are added to the SM, thus producing narrative enrichment. In the opposite sense, a SM created from observation can be narrated by extracting events and narrative relations from the SM into the NCx production model, to generate the corresponding sentence(s) in a narrative to communicate the SM. See Mealier et al. (2017) for details

5 Higher Cognitive Function II. Executive Function Mixed Selectivity

When a reservoir has been trained to perform such high-level cognitive functions, it is of interest to understand the underlying neural dynamics. We initiated such analyses in our comparison of primate neurophysiology underlying complex sequential behavior from Barone and Joseph (1989), and the comparison with the single-unit activity in our reservoir model that performed the same task (Dominey et al. 1995). One of the characteristics of the neural activity in the primate cortex and in the model was the complex mixture of different aspects of the task dimensions, in particular spatial location and sequential order. This form of complex neural activity has been recently examined more closely and characterized as mixed selectivity.

5.1 Revelation of Mixed Selectivity in Primate Cortex

In groundbreaking work that has had major impact in creating a new way of considering cortical activity, Rigotti et al. (2013) characterized mixed selectivity in the activity of single units in primate cortex, and demonstrated the necessity of this coding for successful execution of tasks requiring higher cognitive function. In a task that required seeing a sequence of two visual images and then having to either recognize or recall those images, they observed that frontal cortical neurons encoded complex

non-linear mixtures of image identity, rank in the sequence, and the task that was to be performed. Interestingly, they showed that the higher the dimensionality of the population coding, the better the performance on the task. This research revealed that the phenomenon of mixing task-related aspects in single-unit activity (like Barone and Joseph's location and rank mixing) is a required and inherent property of cortical activity during complex task performance.

5.2 Explore Exploit

We set out to examine reservoir dynamics and mixed selectivity in a complex task that requires exploration and exploitation. We based this approach on the research of Quilodran et al. (2008), who trained macaque monkeys to perform a task, illustrated in Fig. 10A, where one of four spatial targets was rewarded, and the subject had to explore by trial and error to find the rewarded target. After the first reward, the exploration phase ended and the subject could repeat that target for a reward several times in the exploitation phase, before the signal to change indicated the beginning of a new exploration. Quilodran et al. described neural responses related to feedback in the exploration, and the shift between exploration and exploitation. In a reanalysis of the data, we (Enel et al. 2016) determined that the dACC neurons display significant mixed selectivity. ANOVA on the factors target choice, phase, and task epoch revealed a broad distribution of neurons that had significant main effects (i.e., selectivity for one of the three task variables), and two and three-way interactions. Thus, 16% of dACC neurons displayed a dynamic mixed selectivity, revealed by a significant three-way interaction.

5.3 Modeling Explore Exploit

We trained a 1000 unit reservoir model (illustrated in Fig. 10) to perform the explore/exploit task. Essentially the model learns to shift (explore) when no rewarded is provided, to stay (exploit) when the choice is rewarded, and when exploring, to not start with the same target that was just rewarded. The model learns to perform this task well (Enel et al. 2016). We then analyzed the neural activity within the reservoir and compared this to the dACC neurons. The similarity in the neural coding was striking. The distribution of neurons that coded the epoch (explore/exploit), the task phase, and the target choice, between the model and the dACC were quite similar. Most striking, however, was the similar distribution of the different interactions between epoch, phase, and choice. These observations argue that the cortex behaves computationally as a reservoir (Fig. 11).

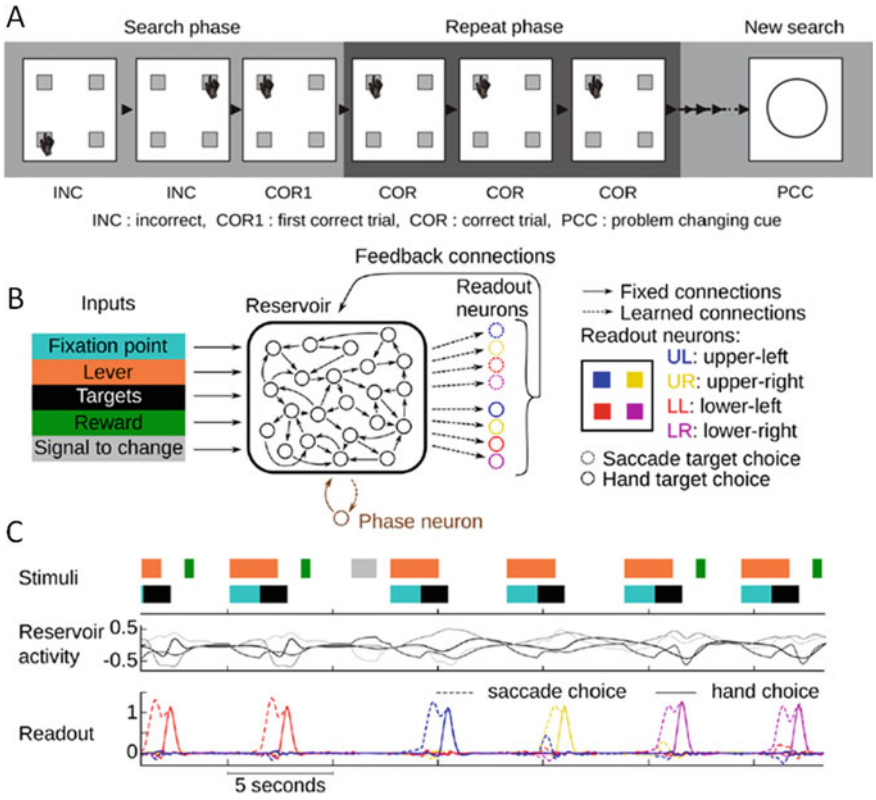


Fig. 10 Reservoir processing of Explore-Exploit task. **A** Task—subject must search for one of four targets that is rewarded (explore). After the first reward, the subject can repeat several times (exploit), until the Signal To Change. **B** Mapping of this task to reservoir: inputs are Fixation point, lever, targets, reward, and signal to changes. Readout Outputs are saccade and touch to the four possible targets. **C** Timing of inputs, reservoir activity (in a few sample cells) and outputs over several trials, with two correct rewarded trials, then the signal to change, and a search of three trials before the new rewarded target is found and repeated

5.4 Adaptation in the Face of Diversity

It has been noted that living cortex is plastic, and that adaptive change is likely a central component of cortical physiology. In contrast, by definition, reservoirs are not plastic. From our perspective, this is not a statement that living cortex is not plastic, but rather an approach to investigate how far we can explain nature in the simplified model of the reservoir. Within this context there are clearly methods to investigate the role of plasticity. Because phase (exploration vs. exploitation) is so central to this task, we investigated the introduction of an adaptation in the form of a neuron that could learn to fire during the exploration phase, and be silent during exploitation. The activity of this phase neuron is illustrated in Fig. 12. We also

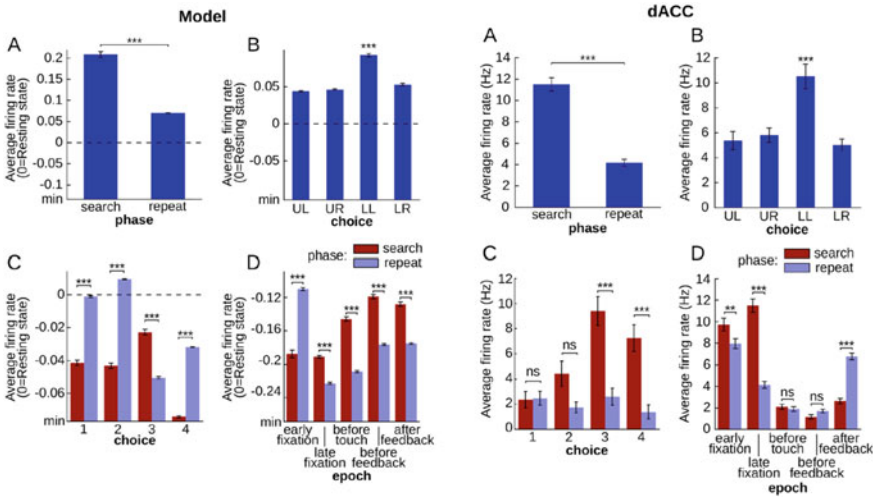


Fig. 11 Simple task-related activity, and non-linear mixed selectivity in reservoir and primate cingulate cortex. **A** and **B**, single units from Model and dACC that illustrate simple phase and choice responses. **C** Model and dACC units with choice selectivity that is modulated by phase (mixed selectivity). **D** Model and dACC units that display complex mixed selectivity for task epoch, and phase

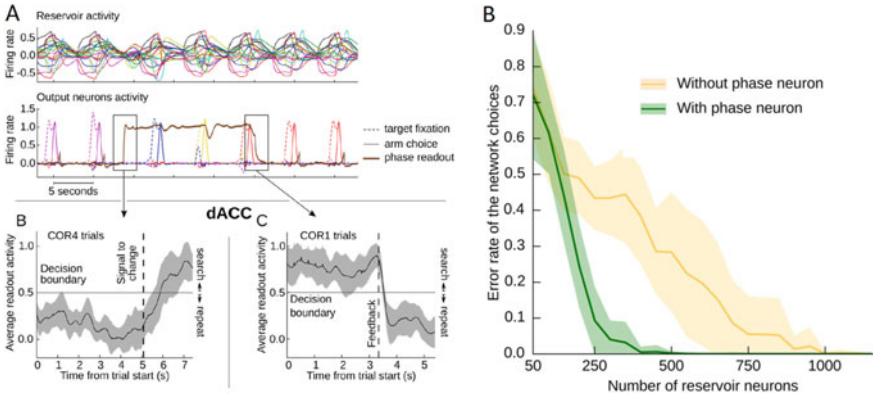


Fig. 12 **A** Decoding of a phase neuron (active during search but not repeat) from reservoir and dACC. **B** Performance of models with and without phase neuron. When phase neuron feeds its task-crucial input back into the reservoir, less than half the number of neurons are required to perform the task

demonstrated that a similar phase neuron can be trained to decode the phase from the population of 85 neurons recorded in the primate dACC. This is despite fact that within the recorded neurons, no single unit was found that displayed this phase encoding behavior. Most interestingly, when this phase information was fed back

into the reservoir, a dramatic improvement in performance could be realized, as revealed by a significant reduction in the number of neurons required in the reservoir to perform the task. Similar benefits of introducing such memory state neurons into the reservoir context has been elegantly demonstrated by Pascanu and Jaeger (2011).

5.5 From Main Effects to Mixed Selectivity

In the initial days of primate neurophysiology, tasks were often reduced to a single behavioral dimension, e.g., making a saccade to a spatial target, and the scientist typically then interrogated single units to determine if they encode the task-related variable. In such cases, many neurons are found to encode the main effect—that is, the neuron has an easily identifiable response to a clearly relevant aspect of the task, e.g., saccade amplitude and direction (Bruce and Goldberg 1985). However, had the animal been performing a more complex task, as in the saccade sequencing task, then more complex responses could have been looked for, and would likely have been found, as observed by Barone and Joseph. In our own work, we observed that the primate cortex and the reservoir indeed display both task-related responses, and complex, dynamic mixed selectivity (Enel et al. 2016). Interestingly, while it is difficult to interpret the function of mixed selectivity in single units, when they are interrogated as a population, the crucial role of mixed selectivity is revealed. It represents a high dimensional projection of task variables and their history, from which diverse pertinent aspects of the problem at hand can be read out of the population (Rigotti et al. 2013; Sussillo and Barak 2013).

6 Cortico-Hippocampal Interaction for Synthesis of Novel Adaptive Experiences

The previous section indicates that the recurrent reservoir maintains a representation of state, including externally visible and hidden internal states as required performing a complex task. Here, we explore another property of reservoir computing, which is the ability to concatenate reservoir state trajectories. When a reservoir with output feedback learns an input sequence, it will traverse a trajectory of internal states. If the end of one state trajectory overlaps with the beginning of another, then the reservoir, when launched on the first will naturally follow it and then continue—via the overlap—into the second trajectory. We recently examined how this property could explain a form of optimization behavior observed in navigating rodents.

When rats are placed in a large (e.g., 1 m diameter) open space with 5–6 baited food wells irregularly distributed in space, they display an interesting form of spatial optimization. The initial trials typically yield trajectories that are not well organized and meandering. Over successive trials, the trajectories become more structured, and finally spatially optimal. This has been characterized as a form of solving the traveling salesman optimization problem (de Jong et al. 2011). During this kind of behavior, activation of hippocampal place cells encodes the position of the animal as it navigates (Moser et al. 2008). Interestingly, between trials, hippocampal place cell activations also recalls previous navigation paths, and future plans. We simulated aspects of these processes by training a reservoir to auto-generate spatial sequences of place cell activation patterns, corresponding to the trajectories illustrated in Fig. 13i (Cazin et al. 2018), in a network where input to the network was the current location in a place cell code, and the readout codes the next location in the navigation sequence, also in a place cell code. The novelty was that the model was trained on the replay of short "snippets" of place cell activation sequences that were drawn from its recent navigation trajectories. For illustration, short paths linking the marked rewards are colored red, while long and inefficient paths are marked in blue. We introduced a place cell activation replay mechanism that favors replay of rewarded locations, and also propagates reward backwards along the spatial trajectory. This is illustrated in Fig. 13ii. Because there is a limited reward propagation budget, for long and inefficient sequences there is not enough reward and the sequence will thus be under-represented in the replay. For short efficient sequences between rewards, there is ample reward, and so the sequence will be well represented in the replay that is used to train the reservoir. This is illustrated in the replay profile in Fig. 13iv, which illustrates that snippets on short trajectories between rewards will have a high probability of being replayed. This means that in the end the reservoir is selectively

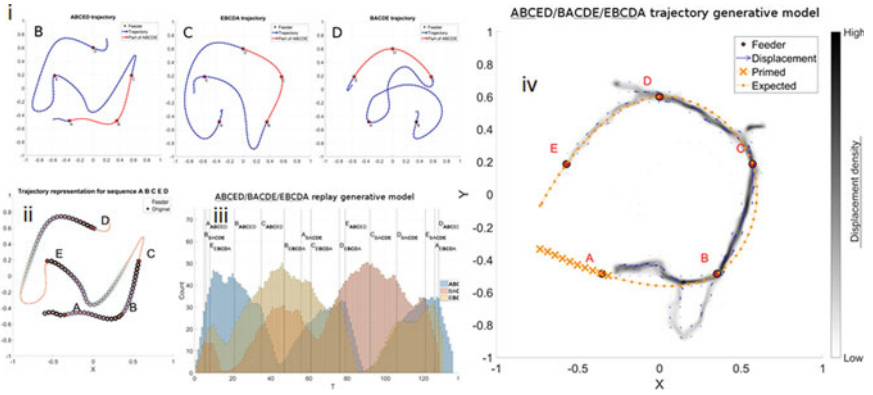


Fig. 13 Reservoir synthesizes an efficient trajectory from multiple ambiguous inputs. i Three spatial navigation sequences between baited feeders (marked). Red indicates "efficient" paths, and blue "non-efficient". ii Reward propagation from rewarded locations forms a spatial gradient. iii Illustration of spatial gradient that favors efficient trajectories. iv. Histogram of generated trajectories illustrates that efficient subsequences have been regrouped to form the overall efficient sequence

exposed predominantly to the efficient subsequences, and can concatenate them to creatively generate the composite optimal sequence that was never seen in its entirety during training.

This work thus introduces a simple form of reinforcement learning into the reservoir computing domain, which may have important impact in future research.

7 Discussion

In 1995, we set out to build a neural network model that respected the neuroanatomy of the primate corticostriatal system, that could learn to perform a visuospatial sequencing task as defined in the primate (Barone and Joseph 1989), and that could explain the complex mix of space and time that was represented in the prefrontal neurons of animals that performed this task. In order to avoid the computational complexity of learning in recurrent connections, we chose to keep these connections fixed (with rich dynamics) and to have plasticity only in the corticostriatal (readout) connections (Dominey 1995; Dominey et al. 1995). Historically, this was the first implementation of reservoir computing.

The resulting model was a success according to our objectives. We then set out to explore the computational capabilities of this system, and began to discover its versatility. In this review, we have seen sensitivity to serial, temporal and abstract structure, the ability to learn and generalize over patterns of grammatical elements for learning grammatical constructions as form to meaning mappings. We have seen the application to complex non-linguistic tasks in the context of exploration/exploitation, and in optimization in spatial navigation. Strikingly, in all of these contexts, the reservoir (and sometimes the readout) activity has been demonstrated to bear striking likeness to that observed in the living cortex. This contributes to the position that cortex is a system of reservoirs organized in parallel interdigitated subsystems, and that progress in future understanding of human cognition will be made through further development and analysis of cooperating reservoir systems.

One of the main areas where we have still not completed this demonstration is in the domain of meaning, but there is hope. Embodied theories of cognition hold that meaning is constituted of reactivation of sensorimotor experiential traces, generating a form of simulation of the intended meaning (Barsalou 2008; Barsalou et al. 2003). This has been revealed in human brain imagery studies that identify the system used for understanding images and sentences that depict everyday human activity, as corresponding to a widespread activation of the sensorimotor system, the spatial cognition system and representations of self and others (Jouen et al. 2018, 2015).

From the reservoir perspective, we can consider that meaning is a dynamic neural state trajectory that traverses a sequence of neural states that is similar to those traversed in the actual experience of a given event. This can be considered in the context of perception and prediction during actions, such that at the same time that the nervous system is navigating a complex high dimensional space in real-time, it is predicting the perceived outcome of its own actions and of those of other agents in

the world. This predictive mechanism will thus be the same for real-time predictive control, and for off-line simulation of lived experiences. The system should continuously be predicting/generating commands for what will happen next. This should be modulated by external contextual cues as well as internal contextual cues. Highly sensorized and actuated humanoid robots are an obvious candidate for providing such a framework, and our current research addresses the generation of meaning in reservoir networks in this context (Mealier et al. 2017; Moulin-Frier et al. 2018).

8 Conclusions

The power of recurrent connections has been clearly demonstrated in the research reported in the different chapters of this volume. When we see that the cortex is a massive system built up of locally constructed reservoir circuits, that are also interconnected by a vast network of longer connections, and that all of this cortex projects with a rich segregated yet interleaving topography to the striatum to allow for highly rich readout—when we appreciate all this, we see a massive computing potential in the cortico-striatal system. However, we also see that we must fashion our thinking about neural computation so as to be able to fathom how this very characteristic organization can be marshaled to allow the highly precise and abstract thought that the human is capable of. Executive function, working memory, variable binding and passing (or their functional equivalents) are all implemented in this framework, and we still have work to construct the integrated computational framework to accomplish this (Graves et al. 2014). In this context, Jaeger (2017) has developed the conceptor framework that allows the imposition of a form of biasing structure on reservoirs, which thus allows the management and control of reservoir dynamics for multiple different stored patterns. Such mechanisms provide a direction for future research on how reservoir dynamics can be controlled for precision cognitive functions.

The parallel corticostriatal circuits described by (Alexander et al. 1986) implement a set of cooperating reservoirs for diverse sensorimotor and cognitive functions. We extended this to include a language or grammatical structure circuit (Dominey 2013; Dominey and Inui 2009; Dominey et al. 2009), and an executive function circuit (Enel et al. 2016). Future research should investigate the interaction of multiple such parallel and overlapping reservoir systems and their cooperation within an integrated, real-time cognitive system.

References

G.E. Alexander, M.R. DeLong, P.L. Strick, Parallel organization of functionally segregated circuits linking basal ganglia and cortex. Annu. Rev. Neurosci. **9**, 357–381 (1986)

P. Barone, J.P. Joseph, Prefrontal cortex and spatial sequencing in macaque monkey. Exp. Brain Res. **78**, 447–464 (1989)

L.W. Barsalou, Grounded cognition. Annu. Rev. Psychol. **59**, 617–645 (2008)
L.W. Barsalou, W. Kyle Simmons, A.K. Barbey, C.D. Wilson, Grounding conceptual knowledge in modality-specific systems. Trends Cogn. Sci. **7**, 84–91 (2003)
E. Bates, B. MacWhinney, Competition, variation, and language learning, in *Mechanisms of Language Acquisition*, ed. by B. MacWhinney, E. Bates (Erlbaum, Hillsdale, NJ, 1987), pp. 157–193
C.J. Bruce, M.E. Goldberg, Physiology of the frontal eye fields. Trends Neurosci. **7**, 436–441 (1984)
C.J. Bruce, M.E. Goldberg, Primate frontal eye fields. I. Single neurons discharging before saccades. J. Neurophysiol. **53**, 603–635 (1985)
P. Calabresi, B. Picconi, A. Tozzi, M. Di Filippo, Dopamine-mediated regulation of corticostriatal synaptic plasticity. Trends Neurosci. **30**, 211–219 (2007)
D. Caplan, C. Baker, F. Dehaut, Syntactic determinants of sentence comprehension in aphasia. Cognition **21**, 117–175 (1985)
N. Cazin, M.L. Alonso, P.S. Chiodi, T. Pelc, B. Harland et al., Prefrontal cortex creates novel navigation sequences from hippocampal place-cell replay with spatial reward propagation 466920 (2018)
D. Centonze, B. Picconi, P. Gubellini, G. Bernardi, P. Calabresi, Dopaminergic control of synaptic plasticity in the dorsal striatum. Eur. J. Neurosci. **13**, 1071–1077 (2001)
G. Chevalier, J.M. Deniau, Disinhibition as a basic process in the expression of striatal functions. Trends Neurosci. **13**, 277–280 (1990)
L.W. de Jong, B. Gereke, G.M. Martin, J.-M. Fellous, The traveling salesrat: insights into the dynamics of efficient spatial navigation in the rodent. J. Neural Eng. **8**, (2011)
M. Di Filippo, B. Picconi, M. Tantucci, V. Ghiglieri, V. Bagetta et al., Short-term and long-term plasticity at corticostriatal synapses: implications for learning and memory. Behav. Brain Res. **199**, 108–118 (2009)
P.F. Dominey, Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. Biol. Cybern. **73**, 265–274 (1995)
P.F. Dominey, Influences of temporal organization on sequence learning and transfer: Comments on Stadler (1995) and Curran and Keele (1993). J. Exp. Psychol. Learn. Mem. Cogn. **24**, 14 (1998a)
P.F. Dominey, A shared system for learning serial and temporal structure of sensori-motor sequences? Evidence from simulation and human experiments. Brain Res. Cogn. Brain Res. **6**, 163–172 (1998b)
P.F. Dominey, Recurrent temporal networks and language acquisition-from corticostriatal neurophysiology to reservoir computing. Front. Psychol. **4**, 1–14 (2013)
P.F. Dominey, M.A. Arbib, A cortico-subcortical model for generation of spatially accurate sequential saccades. Cereb. Cortex **2**, 153–175 (1992)
P.F. Dominey, T. Lelekov, Nonlinguistic transformation processing in agrammatic aphasia. Behav. Brain Sci. **23**, 30-+ (2000)
P.F. Dominey, F. Ramus, Neural network processing of natural language: I. Sensitivity to serial, temporal and abstract structure of language in the infant. Lang. Cogn. Process. **15**, 87–127 (2000)
P.F. Dominey, T. Inui, Cortico-striatal function in sentence comprehension: insights from neurophysiology and modeling. Cortex **45**, 1012–1018 (2009)
P.F. Dominey, M.A. Arbib, J.P. Joseph, A model of corticostriatal plasticity for learning oculomotor associations and sequences. J. Cogn. Neurosci. **7**, 25 (1995)
P.F. Dominey, T. Lelekov, J. Ventre-Dominey, M. Jeannerod, Dissociable processes for learning the surface structure and abstract structure of sensorimotor sequences. J. Cogn. Neurosci. **10**, 734–751 (1998)
P.F. Dominey, M. Hoen, J.M. Blanc, T. Lelekov-Boissard, Neurological basis of language and sequential cognition: evidence from simulation, aphasia, and ERP studies. Brain Lang. **86**, 207–225 (2003)
P.F. Dominey, T. Inui, M. Hoen, Neural network processing of natural language: II. Towards a unified model of corticostriatal function in learning sentence comprehension and non-linguistic sequencing. Brain Lang. **109**, 80–92 (2009)

P. Enel, E. Procyk, R. Quilodran, P.F. Dominey, Reservoir computing properties of neural dynamics in prefrontal cortex. PLoS Comput. Biol. **12**, (2016)
A.D. Friederici, Towards a neural basis of auditory sentence processing. Trends Cogn. Sci. **6**, 78–84 (2002)
A.D. Friederici, The cortical language circuit: from auditory perception to sentence comprehension. Trends Cogn. Sci. (2012)
M.E. Goldberg, M.C. Bushnell, Behavioral enhancement of visual responses in monkey cerebral cortex. II. Modulation in frontal eye fields specifically related to saccades. J. Neurophysiol. **46**, 773–787 (1981)
P.S. Goldman-Rakic, Circuitry of primate prefrontal cortex and regulation of behavior by representational memory. Handb. Neurophysiol. **5**, 40 (1987)
A. Graves, G. Wayne, I. Danihelka, Neural turing machines (2014), arXiv:1410.5401
P. Hagoort, On Broca, brain, and binding: a new framework. Trends Cogn. Sci. **9**, 416–423 (2005)
O. Hikosaka, Role of basal ganglia in saccades. Rev. Neurol. (Paris) **145**, 580–586 (1989)
O. Hikosaka, R.H. Wurtz, Visual and oculomotor functions of monkey substantia nigra pars reticulata. I. Relation of visual and auditory responses to saccades. J. Neurophysiol. **49**, 1230–1253 (1983a)
O. Hikosaka, R.H. Wurtz, Visual and oculomotor functions of monkey substantia nigra pars reticulata. II. Visual responses related to fixation of gaze. J. Neurophysiol. **49**, 1254–1267 (1983b)
O. Hikosaka, R.H. Wurtz, Visual and oculomotor functions of monkey substantia nigra pars reticulata. III. Memory-contingent visual and saccade responses. J. Neurophysiol. **49**, 1268–1284 (1983c)
O. Hikosaka, R.H. Wurtz, Visual and oculomotor functions of monkey substantia nigra pars reticulata. IV. Relation of substantia nigra to superior colliculus. J. Neurophysiol. **49**, 1285–1301 (1983d)
O. Hikosaka, M. Sakamoto, S. Usui, Functional properties of monkey caudate neurons. I. Activities related to saccadic eye movements. J. Neurophysiol. **61**, 780–798 (1989a)
O. Hikosaka, M. Sakamoto, S. Usui, Functional properties of monkey caudate neurons. II. Visual and auditory responses. J. Neurophysiol. **61**, 799–813 (1989b)
O. Hikosaka, M. Sakamoto, S. Usui, Functional properties of monkey caudate neurons. III. Activities related to expectation of target and reward. J. Neurophysiol. **61**, 814–832 (1989c)
X. Hinaut, P.F. Dominey, Real-time parallel processing of grammatical structure in the fronto-striatal system: a recurrent network simulation study using reservoir computing. PLoS ONE **8**, 1–18 (2013)
X. Hinaut, F. Lance, C. Droin, M. Petit, G. Pointeau, P.F. Dominey, Corticostriatal response selection in sentence production: Insights from neural network simulation with reservoir computing. Brain Lang. **150**, 54–68 (2015)
M. Hoen, P.F. Dominey, ERP analysis of cognitive sequencing: A left anterior negativity related to structural transformation processing. NeuroReport **11**, 3187–3191 (2000)
M. Hoen, M. Pachot-Clouard, C. Segebarth, P.F. Dominey, When Broca experiences the Janus syndrome: an ER-fMRI study comparing sentence comprehension and cognitive sequence processing. Cortex **42**, 605–623 (2006)
H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks-with an erratum note'. GMD Technical Report 148. German National Research Center for Information Technology Bonn, Germany (2001)
H. Jaeger, Using conceptors to manage neural long-term memories for temporal patterns. J. Mach. Learn. Res. **18**, 1–43 (2017)
H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**, 78–80 (2004)
A. Jouen, T. Ellmore, C. Madden, C. Pallier, P. Dominey, J. Ventre-Dominey, Beyond the word and image: characteristics of a common meaning system for language and vision revealed by functional and structural imaging. NeuroImage **106**, 72–85 (2015)

A. Jouen, T. Ellmore, C. Madden-Lombardi, C. Pallier, P. Dominey, J. Ventre-Dominey, Beyond the word and image: II-Structural and functional connectivity of a common semantic system. NeuroImage **166**, 185–197 (2018)
P.W. Jusczyk, *The Discovery of Spoken Language* (MIT Press, Cambridge, 1997)
T. Lelekov, N. Franck, P.F. Dominey, N. Georgieff, Cognitive sequence processing and syntactic comprehension in schizophrenia. NeuroReport **11**, 2145–2149 (2000)
P. Li, B. Macwhinney, Competition model. Encycl. Appl. Linguist. (2013)
T. Ljungberg, P. Apicella, W. Schultz, Responses of monkey dopamine neurons during learning of behavioral reactions. J. Neurophysiol. **67**, 145–163 (1992)
M. Lukosevicius, H. Jaeger, Reservoir computing approaches to recurrent neural network training. Comput. Sci. Rev. **3**, 22 (2009)
M. Lukoševičius, H. Jaeger, B. Schrauwen, Reservoir computing trends. KI-Künstliche Intell. **26**, 365–371 (2012)
W. Maass, T. Natschlager, H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. **14**, 2531–2560 (2002)
G.F. Marcus, S. Vijayan, S. Bandi Rao, P.M. Vishton, Rule learning by seven-month-old infants. Science **283**, 77–80 (1999)
N. Markov, P. Misery, A. Falchier, C. Lamy, J. Vezoli et al., Weight consistency specifies regularities of macaque cortical networks. Cereb. Cortex **21**, 1254–1272 (2011)
A.-L. Mealier, G. Pointeau, S. Mirliaz, K. Ogawa, M. Finlayson, P.F. Dominey, Narrative constructions for the organization of self experience: proof of concept via embodied robotics. Front. Psychol.: Lang. (2017)
E.I. Moser, E. Kropff, M.-B. Moser, Place cells, grid cells, and the brain's spatial representation system. Annu. Rev. Neurosci. **31**, 69–89 (2008)
C. Moulin-Frier, T. Fischer, M. Petit, G. Pointeau, J.-Y. Puigbo et al., DAC-h3: a proactive robot cognitive architecture to acquire and express knowledge about the world and the self. IEEE Trans. Cogn. Dev. Syst. **10**, 1005–1022 (2018)
T. Nazzi, J. Bertoncini, J. Mehler, Language discrimination by newborns: toward an understanding of the role of rhythm. J. Exp. Psychol. Hum. Percept. Perform. **24**, 756–766 (1998)
R. Pascanu, H. Jaeger, A neurodynamical model for working memory. Neural Netw. **24**, 199–207 (2011)
B.A. Pearlmutter, Gradient calculations for dynamic recurrent neural networks: A survey. IEEE Trans. Neural Netw. **6**, 1212–1228 (1995)
R. Quilodran, M. Rothe, E. Procyk, Behavioral shifts and action valuation in the anterior cingulate cortex. Neuron **57**, 314–325 (2008)
M. Rigotti, O. Barak, M.R. Warden, X.-J. Wang, N.D. Daw et al., The importance of mixed selectivity in complex cognitive tasks. Nature (2013)
J.R. Saffran, R.N. Aslin, E.L. Newport, Statistical learning by 8-month-old infants. Science **274**, 1926–1928 (1996)
W. Schultz, P. Apicella, T. Ljungberg, Responses of monkey dopamine neurons to reward and conditioned stimuli during successive steps of learning a delayed response task. J. Neurosci. **13**, 900–913 (1993)
L.D. Selemon, P.S. Goldman-Rakic, Longitudinal topography and interdigitation of corticostriatal projections in the rhesus monkey. J. Neurosci. **5**, 776–794 (1985)
L.D. Selemon, P.S. Goldman-Rakic, Common cortical and subcortical targets of the dorsolateral prefrontal and posterior parietal cortices in the rhesus monkey: evidence for a distributed neural network subserving spatially guided behavior. J. Neurosci. **8**, 4049–4068 (1988)
M.A. Stadler, Implicit serial learning: questions inspired by Hebb (1961). Mem. Cogn. **21**, 819–827 (1993)
D. Sussillo, O. Barak, Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. Neural Comput. **25**, 626–649 (2013)

K. Szalisznyó, D. Silverstein, M. Teichmann, H. Duffau, A. Smits, Cortico-striatal language pathways dynamically adjust for syntactic complexity: a computational study. Brain Lang. **164**, 53–62 (2017)

D. Verstraeten, B. Schrauwen, S. Dieleman, P. Brakel, P. Buteneers, D. Pecevski, Oger: modular learning architectures for large-scale sequential processing. J. Mach. Learn. Res. **13**, 2995–2998 (2012)

Reservoirs Learn to Learn

Anand Subramoney, Franz Scherr, and Wolfgang Maass

Abstract The common procedure in reservoir computing is to take a "found" reservoir, such as a recurrent neural network with randomly chosen synaptic weights or a complex physical device, and to adapt the weights of linear readouts from this reservoir for a particular computing task. We address the question of whether the performance of reservoir computing can be significantly enhanced if one instead optimizes some (hyper)parameters of the reservoir, not for a single task but for the range of all possible tasks in which one is potentially interested, before the weights of linear readouts are optimized for a particular computing task. After all, networks of neurons in the brain are also known to be not randomly connected. Rather, their structure and parameters emerge from complex evolutionary and developmental processes, arguably in a way that enhances the speed and accuracy of subsequent learning of any concrete task that is likely to be essential for the survival of the organism. We apply the Learning-to-Learn (L2L) paradigm to mimic this two-tier process, where a set of (hyper)parameters of the reservoir are optimized for a whole family of learning tasks. We found that this substantially enhances the performance of reservoir computing for the families of tasks that we considered. Furthermore, L2L enables a new form of reservoir learning that tends to enable even faster learning, where not even the weights of readouts need to be adjusted for learning a concrete task. We present demos and performance results of these new forms of reservoir computing for reservoirs that consist of networks of spiking neurons and are hence of particular interest from the perspective of neuroscience and implementations in spike-based neuromorphic hardware. We leave it as an open question of what performance advantage the new methods that we propose provide for other types of reservoirs.

1 Introduction

One motivation for the introduction of the liquid computing model (Maass et al. 2002) was to understand how complex neural circuits in the brain, or cortical columns, are able to support the diverse computing and learning tasks which the brain has to solve.

A. Subramoney · F. Scherr · W. Maass (✉)
Institute for Theoretical Computer Science, Graz University of Technology, Graz, Austria
e-mail: maass@igi.tugraz.at

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_3

It was shown that recurrent networks of spiking neurons (RSNNs) with randomly chosen weights, including models for cortical columns with given connection probability between laminae and neural populations, could in fact support a large number of different learning tasks, where only the synaptic weights to readout neurons were adapted for a specific task (Maass et al. 2004; Haeusler and Maass 2006). Independently from that, a similar framework (Jaeger 2001) was developed for artificial neural networks, and both methods were subsumed under the umbrella of reservoir computing (Verstraeten et al. 2007). Our methods for training reservoirs that are discussed in this paper have so far only been tested for reservoirs consisting of spiking neurons, as in the liquid computing model.

Considering the learning capabilities of the brain, it is fair to assume that synaptic weights of these neural networks are not just randomly chosen, but shaped through a host of processes—from evolution, over development to preceding learning experiences. These processes are likely to aim at improving the learning and computing capability of the network. Hence we asked whether the performance of reservoirs can also be improved by optimizing the weights of recurrent connections within the recurrent network for a large range of learning tasks. The Learning-to-Learn (L2L) setup offers a suitable framework for examining this question. This framework builds on a long tradition of investigating L2L, also referred to as meta-learning, in cognitive science, neuroscience, and machine learning (Abraham and Bear 1996; Wang et al. 2016, 2018; Hochreiter et al. 2001). The formal model from Hochreiter et al. (2001) and Wang et al. (2016) and related recent work in machine learning assume that learning (or optimization) takes place in two interacting loops (see Fig. 1A). The outer loop aims at capturing the impact of adaptation on a larger time scale (such as evolution, development, and prior learning in the case of brains). It optimizes a set of parameters Θ, for a—in general infinitely large—family $\mathcal{F}$ of learning tasks. Any learning or optimization method can be used for that. For learning a particular task C from $\mathcal{F}$ in the inner loop, the neural network can adapt those of its parameters which do not belong to the hyperparameters Θ that are controlled by the outer loop. These are in our first demo (Sect. 2) the weights of readout neurons. In our second demo in Sect. 3, we assume that—like in Wang et al. (2016, 2018) and Hochreiter et al. (2001)—ALL weights from, to, and within the neural network, in particular

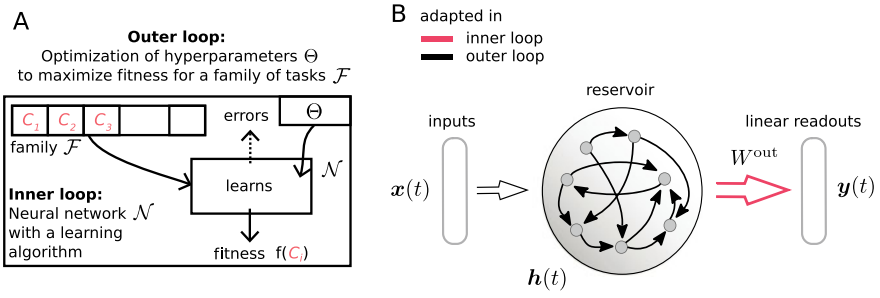


Fig. 1 Learning-to-Learn setup: **A** Schematic of the nested optimization that is carried out in Learning-to-Learn (L2L). **B** Learning architecture that is used to obtain optimized reservoirs

also the weights of readout neurons, are controlled by the outer loop. In this case, just the dynamics of the network can be used to maintain information from preceding examples for the current learning task in order to produce a desirable output for the current network input. One exciting feature of this L2L approach is that all synaptic weights of the network can be used to encode a really efficient network learning algorithm. It was recently shown in Bellec et al. (2018b) that this form of L2L can also be applied to RSNNs. We discuss in Sect. 3 also the interesting fact that L2L induces priors and internal models into reservoirs.

The structure of this article is as follows: We address in Sect. 2 the first form of L2L, where synaptic weights to readout neurons can be trained for each learning task, exactly like in the standard reservoir computing paradigm. We discuss in Sect. 3 the more extreme form of L2L where ALL synaptic weights are determined by the outer loop of L2L, so that no synaptic plasticity is needed for learning in the inner loop. In Sect. 4, we give full technical details for the demos given in Sects. 2 and 3. Finally, in Sect. 5, we will discuss implications of these results and list a number of related open problems.

2 Optimizing Reservoirs to Learn

In the typical workflow of solving a task in reservoir computing, we have to address two main issues: (1) a suitable reservoir has to be generated and (2) a readout function has to be determined that maps the state of the reservoir to a target output. In the following, we address the first issue by a close investigation of how we can improve the process of obtaining suitable reservoirs. For this purpose, we consider here RSNNs as the implementation of the reservoir and its state refers to the activity of all units within the network. In order to generate an instance of such a reservoir, one usually specifies a particular network architecture of the RSNN and then generates the corresponding synaptic weights at random. Those remain fixed throughout learning of a particular task. Clearly, one can tune this random creation process to better suit the needs of the considered task. For example, one can adapt the probability distribution from which weights are drawn. However, it is likely that a reservoir, generated according to a coarse random procedure, is far from perfect at producing reservoir states that are really useful for the readout.

A more principled way of generating a suitable reservoir is to optimize their dynamics for the range of tasks to be expected, such that a readout can easily extract the information it needs.

Description of optimized reservoirs: The main characteristic of our approach is to view the weight of every synaptic connection of the RSNN that implements the reservoir as hyperparameters Θ and to optimize them for the range of tasks. In particular, Θ includes both recurrent and input weights (W^{rec}, W^{in}), but also the initialization of the readout $W^{\text{out,init}}$. This viewpoint allows us to tune the dynamics of the reservoir to give rise to particularly useful reservoir states. Learning of a

particular task can then be carried out as usual, where commonly a linear readout is learned, for example by the method of least squares or even simpler by gradient descent.

As previously described, two interacting loops of optimization are introduced, consisting of an inner loop and an outer loop (Fig. 1A). The inner loop consists here of tasks C that require to map an input time series $\boldsymbol{x}_C(t)$ to a target time series $\boldsymbol{y}_C(t)$ (see Fig. 2A). To solve such tasks, $\boldsymbol{x}_C(t)$ is passed as a stream to the reservoir, which then processes these inputs, and produces reservoir states $\boldsymbol{h}_C(t)$. The emerging features are then used for target prediction by a linear readout:

$$\widehat{\boldsymbol{y}}_C(t) = W_C^{\text{out}}[\boldsymbol{x}_C(t), \boldsymbol{h}_C(t)]^T \ . \tag{1}$$

On this level of the inner loop, only the readout weights W_C^{out} are learned. Specifically, we chose here a particularly simple plasticity rule acting upon these weights, given by gradient descent:

$$\Delta W_C^{\text{out}} = \eta\Big(\boldsymbol{y}_C(t) - \widehat{\boldsymbol{y}}_C(t)\Big) \cdot \boldsymbol{h}_C(t)^T \ , \tag{2}$$

which can be applied continuously, or changes can be accumulated. Note that the initialization of readout weights is provided as a hyperparameter $W^{\text{out,init}}$ and η represents a learning rate.

On the other hand, the outer loop is concerned with improving the learning process in the inner loop for an entire family of tasks $\mathcal{F}$. This goal is formalized using an optimization objective that acts upon the hyperparameters $\Theta = \{W^{\text{in}}, W^{\text{rec}}, W^{\text{out,init}}\}$:

$$\min_{\Theta} \ \mathbb{E}_{C\sim\mathcal{F}}\left[\int_t \left\|\boldsymbol{y}_C(t) - \widehat{\boldsymbol{y}}_C(t)\right\|_2^2 dt\right] \tag{3}$$

$$\text{subject to} \ \ \Delta W_C^{\text{out}} = \eta\Big(\boldsymbol{y}_C(t) - \widehat{\boldsymbol{y}}_C(t)\Big) \cdot \boldsymbol{h}_C(t)^T \quad \text{(readout learning)}. \tag{4}$$

Regressing Volterra filters: Models of reservoir computing typically get applied to tasks that exhibit nontrivial temporal relationships in the mapping from input signal $\boldsymbol{x}_C(t)$ to target $\boldsymbol{y}_C(t)$. Such tasks are suitable because reservoirs have a property of fading memory: Recent events leave a footprint in the reservoir dynamics which can later be extracted by appropriate readouts. Theory guarantees that a large enough reservoir can retain all relevant information. In practice, one is bound to a dynamical system of limited size and hence, it is likely that a reservoir, optimized for the memory requirements and time scales of the specific task family at hand, will perform better than a reservoir that was generated at random.

We consider a task family $\mathcal{F}$ where each task C is determined by a randomly chosen Volterra filter (Volterra 2005). Here, the target $\boldsymbol{y}_C(t)$ arises by application of a randomly chosen second-order Volterra filter (Volterra 2005) to the input $\boldsymbol{x}_C(t)$:

$$\boldsymbol{y}_C(t) = \int_\tau k_C^1(\tau)\boldsymbol{x}_C(t-\tau)\,d\tau + \int_{\tau_1}\int_{\tau_2} k_C^2(\tau_1,\tau_2)\boldsymbol{x}_C(t-\tau_1)\boldsymbol{x}_C(t-\tau_2)\,d\tau_1 d\tau_2 \ , \tag{5}$$

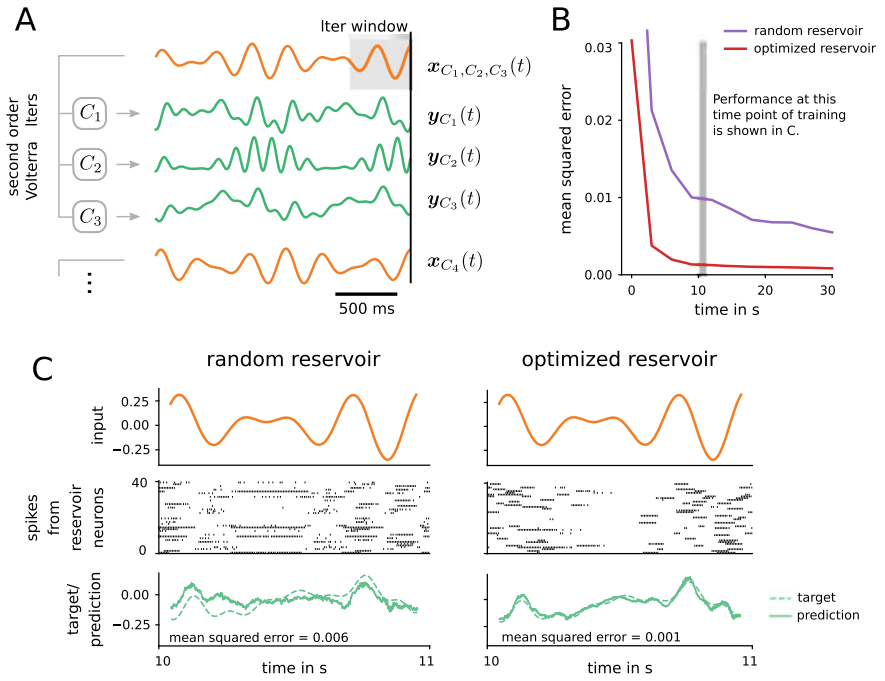


Fig. 2 Learning to learn a nonlinear transformation of a time series: **A** Different tasks C_i arise by sampling second-order Volterra kernels according to a random procedure. Input time series $\boldsymbol{x}_C(t)$ are given as a sum of sines with random properties. To exhibit the variability in the Volterra kernels, we show three examples where different Volterra kernels are applied to the same input. **B** Learning performance in the inner loop using the learning rule (2), both for the case of a reservoir with random weights and for a reservoir that was trained in the outer loop by L2L. Performance at the indicated time window is shown in Panel C. **C** Sample performance of a random reservoir and of an optimized reservoir after readouts have been trained for 10 s. Network activity shows 40 neurons out of 800

see Fig. 2A. The input signal $\boldsymbol{x}_C(t)$ is given as a sum of two sines with different frequencies and with random phase and amplitude. The kernel used in the filter is also sampled randomly according to a predefined procedure for each task C, see Sect. 4.3, and exhibits a typical temporal time scale. Here, the reservoir is responsible to provide suitable features that typically arise for such second-order Volterra filters. In this way, readout weights W_C^{out}, which are adapted according to Eq. (2), can easily extract the required information.

Implementation: The simulations were carried out in discrete time, with steps of 1 ms length. We used a network of 800 recurrently connected neurons with leaky integrate-and-fire (LIF) dynamics. Such neurons are equipped with a membrane potential in which they integrate input current. If this potential crosses a certain threshold, they emit a spike and the membrane voltage is reset, see Sect. 4.1 for details. The reservoir state was implemented as a concatenation of the exponentially filtered

spike trains of all neurons (with a time constant of $\tau_{\text{readout}} = 20\,\text{ms}$). Learning of the linear readout weights in the inner loop was implemented using gradient descent as outlined in Eq. (2). We accumulated weight changes in chunks of 1000 ms and applied them at the end. The objective for the outer loop, as given in Eq. (3), was optimized using backpropagation through time (BPTT), which is an algorithm to perform gradient descent in recurrent neural networks. Observe that this is possible because the dynamics of the plasticity in Eq. (2) is itself differentiable and can therefore be optimized by gradient descent. Because the threshold function that determines the neuron outputs is not differentiable, a heuristic was required to address this problem. Details can be found in Sect. 4.2.

Results: The reservoir that emerged from outer-loop training was compared against a reference baseline, whose weights were not optimized for the task family, but had otherwise exactly the same structure and learning rule for the readout. In Fig. 2B, we report the learning performance on unseen task instances from the family $\mathcal{F}$, averaged over 200 different tasks. We find that the learning performance of the optimized reservoir is substantially improved as compared to the random baseline.

This becomes even more obvious when one compares the quality of the fit on a concrete example as shown in Fig. 2C. Whereas the random reservoir fails to make consistent predictions about the desired output signal based on the reservoir state, the optimized reservoir is able to capture all important aspects of the target signal. This occurred just 10 s within learning the specific task, because the optimized reservoir was already confronted before with tasks of a similar structure, and could capture through the outer-loop optimization the smoothness of the Volterra kernels and the relevant time dependencies in its recurrent weights.

3 Reservoirs Can Also Learn Without Changing Synaptic Weights to Readout Neurons

We next asked whether reservoirs could also learn a specific task without changing any synaptic weight, not even weights to readout neurons. It was shown in Hochreiter et al. (2001) that LSTM networks can learn nonlinear functions from a teacher without modifying their recurrent or readout weights. It has recently been argued in Wang et al. (2018) that the pre-frontal cortex (PFC) accumulates knowledge during fast reward-based learning in its short-term memory, without using synaptic plasticity, see the text to supplementary Fig. 3 in Wang et al. (2018). The experimental results of Perich et al. (2018) also suggest a prominent role of network dynamics and short-term memory for fast learning in the motor cortex. Inspired by these results from biology and machine learning, we explored the extent to which recurrent networks of spiking neurons can learn using just their internal dynamics, without synaptic plasticity.

In this section, we show that one can generate reservoirs through L2L that are able to learn with fixed weights, provided that the reservoir receives feedback about

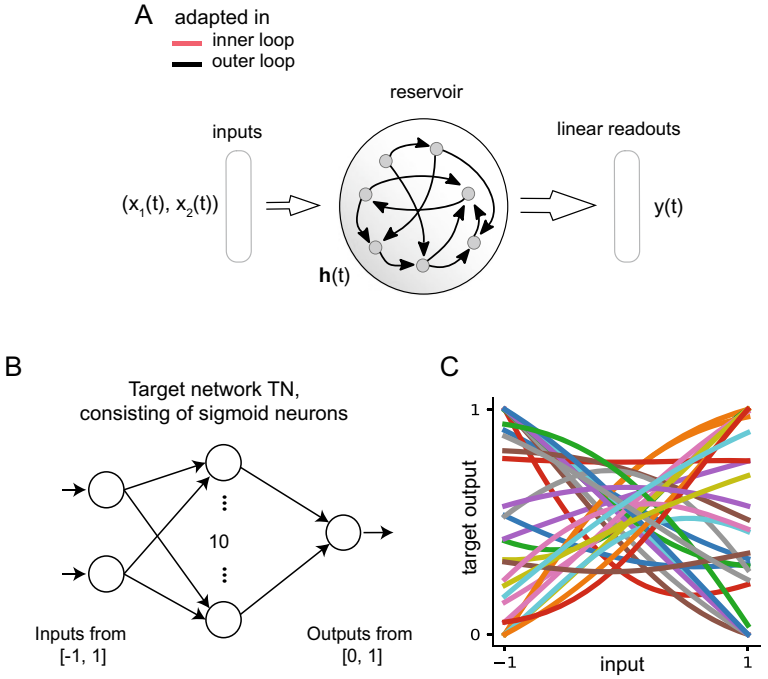


Fig. 3 L2L setup with reservoirs that learn using their internal dynamics A Learning architecture for RSNN reservoirs. All the weights are only updated in the outer-loop training using BPTT. **B** Supervised regression tasks are implemented as neural networks with randomly sampled weights: target networks (TN). **C** Sample input/output curves of TNs on a 1D subset of the 2D input space, for different weight and bias values

the prediction target as input. In addition, relying on the internal dynamics of the reservoir to learn allows the reservoir to learn as fast as possible for a given task, i.e., the learning speed is not determined by any predetermined learning rate.

Target networks as the task family $\mathcal{F}$: We chose the task family to demonstrate that reservoirs can use their internal dynamics to regress complex nonlinear functions and are not limited to generating or predicting temporal patterns. This task family also allows us to illustrate and analyze the learning process in the inner loop more explicitly. We defined the family of tasks $\mathcal{F}$ using a family of nonlinear functions that are each defined by a target feed-forward network (TN) as illustrated in Fig. 3B. Specifically, we chose a class of continuous functions of two real-valued variables (x_1, x_2) as the family $\mathcal{F}$ of tasks. This class was defined as the family of all functions that can be computed by a two-layer artificial neural network of sigmoidal neurons with 10 neurons in the hidden layer and weights and biases in the range $[-1, 1]$. Thus overall, each such target network (TN) from $\mathcal{F}$ was defined through 40 parameters in the range $[-1, 1]$: 30 weights and 10 biases. Random instances of target networks were generated for each episode by randomly sampling the 40 parameters in the

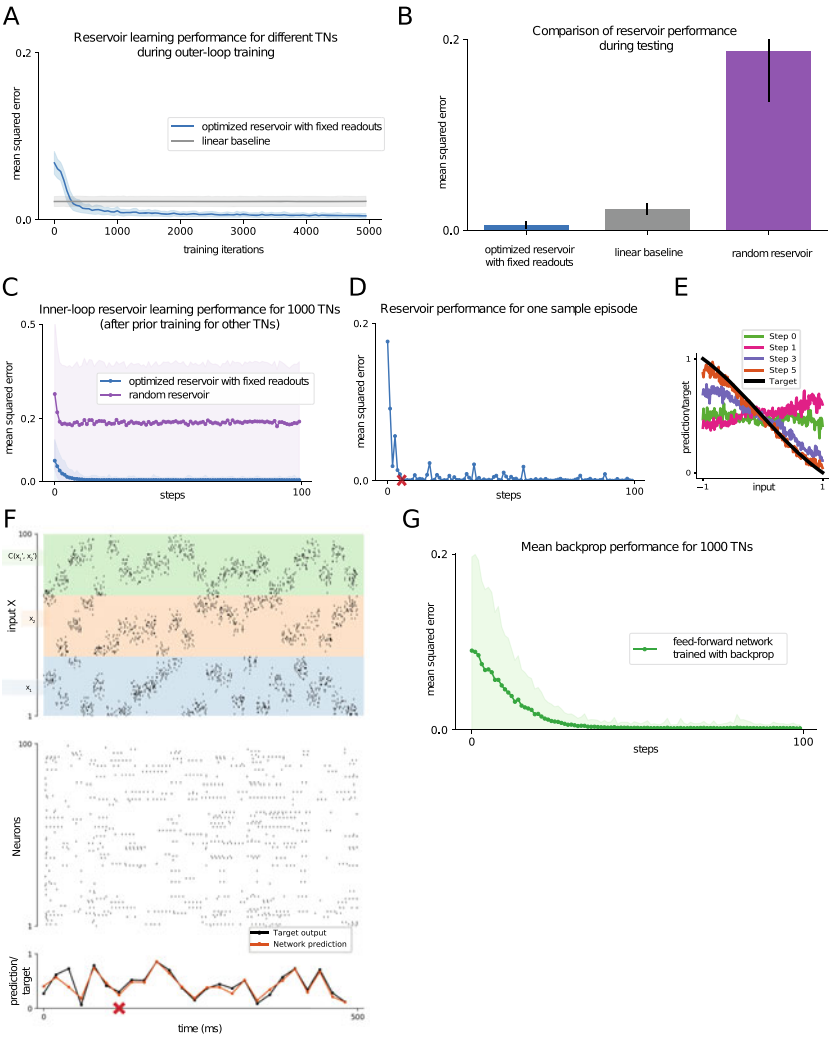


Fig. 4 Learning to learn a nonlinear function that is defined by an unknown target network (TN): **A** Performance of the reservoir in learning a new TN during training in the outer loop of L2L. **B** Performance of the optimized reservoir during testing compared to a random reservoir and the linear baseline. **C** Learning performance within a single inner-loop episode of the reservoir for 1000 new TNs (mean and one standard deviation). Performance is compared to that of a random reservoir. **D** Performance for a single sample TN, a red cross marks the step after which output predictions became very good for this TN. The spike raster for this learning process is the one depicted in (F). **E** The internal model of the reservoir (as described in the text) is shown for the first few steps of inner-loop learning. The reservoir starts by predicting a smooth function and updates its internal model in just 5 steps to correctly predict the target function. **F** Network input (top row, only 100 of 300 neurons shown), internal spike-based processing with low firing rates in the neuron populations (middle row), and network output (bottom row) for 25 steps of 20 ms each. **G** Learning performance of backpropagation for the same 1000 TNs as in C, working directly on the ANN from Fig. 3B, with a prior for small weights, with the best hyperparameters from a grid search

above range. Most of the functions that are computed by TNs from the class $\mathcal{F}$ are nonlinear, as illustrated in Fig. 3C for the case of inputs (x_1, x_2) with $x_1 = x_2$.

Learning setup: In an inner-loop learning episode, the reservoir was shown as a sequence of pairs of inputs (x_1, x_2) and delayed targets $C(x_1', x_2')$ sampled from the nonlinear function generated by one random instance of the TN. After each such pair was presented, the reservoir was trained to produce a prediction $\hat{C}(x_1, x_2)$ of $C(x_1, x_2)$. The task of the reservoir was to produce predictions with a low error. In other words, the task of the reservoir was to perform nonlinear regression on the presented pairs of inputs and targets and produce predictions of low error on new inputs. The reservoir was optimized in the outer loop to learn this fast and well.

When giving an input x_1, x_2 for which the reservoir had to produce prediction $\hat{C}(x_1, x_2)$, we could not also give the target $C(x_1, x_2)$ for that same input at the same time. This is because, the reservoir could then "cheat" by simply producing this value $C(x_1, x_2)$ as its prediction $\hat{C}(x_1, x_2)$. Therefore, we gave the target value to the reservoir with a delay, after it had generated the prediction $\hat{C}(x_1, x_2)$. Giving the target value as input to the reservoir is necessary, as otherwise, the reservoir has no way of figuring out the specific underlying nonlinear function for which it needs to make predictions.

Learning is carried out simultaneously in two loops as before (see Fig. 1A). Like in Hochreiter et al. (2001), Wang et al. (2016), and Duan et al. (2016), we let all synaptic weights of $\mathcal{N}$, including the recurrent, input, and readout weights, to belong to the set of hyperparameters that are optimized in the outer loop. Hence, the network is forced to encode all results from learning the current task C in its internal state, in particular in its firing activity. Thus, the synaptic weights of the neural network $\mathcal{N}$ are free to encode an efficient *algorithm* for learning arbitrary tasks C from $\mathcal{F}$.

Implementation: We considered a reservoir $\mathcal{N}$ consisting of 300 LIF neurons with full connectivity. The neuron model is described in Methods Sect. 4.1. All neurons in the reservoir received input from a population X of 300 external input neurons. A linear readout receiving inputs from all neurons in the reservoir was used for the output predictions. The reservoir received a stream of three types of external inputs (see top row of Fig. 4F): the values of x_1, x_2, and of the output $C(x_1', x_2')$ of the TN for the preceding input pair x_1', x_2' (set to 0 at the first trial), each represented through population coding in an external population of 100 spiking neurons. It produced outputs in the form of weighted spike counts during 20 ms windows from all neurons in the network (see bottom row of Fig. 4F). The weights for this linear readout were trained, like all weights inside the reservoir, in the outer loop, and remained fixed during learning of a particular TN.

The training procedure in the outer loop of L2L was as follows: Network training was divided into training episodes. At the start of each training episode, a new TN was randomly chosen and used to generate target values $C(x_1, x_2) \in [0, 1]$ for randomly chosen input pairs (x_1, x_2). 400 of these input pairs and targets were used as training data and presented one per step to the reservoir during the episode, where each step lasted 20 ms. The reservoir parameters were updated using BPTT to minimize the mean-squared error between the reservoir output and the target in the training

set, using gradients computed over batches of 10 such episodes, which formed one iteration of the outer loop. In other words, each weight update included gradients calculated on the input/target pairs from 10 different TNs. This training procedure forced the reservoir to adapt its parameters in a way that supported learning of many different TNs, rather than specializing in predicting the output of a single TN. After training, the weights of the reservoir remained fixed, and it was required to learn the input/output behavior of TNs from $\mathcal{F}$ that it had never seen before in an online manner by just using its fading memory and dynamics. See Methods (Sect. 4.4) for further details of the implementation.

Results: The reservoir achieves low mean-squared error (MSE) for learning new TNs from the family $\mathcal{F}$, significantly surpassing the performance of an optimal linear approximator (linear regression) that was trained on all 400 pairs of inputs and target outputs, see gray bar in Fig. 4B. One sample of a generic learning process is shown in Fig. 4D.

Each sequence of examples evokes an "internal model" of the current target function in the internal dynamics of the reservoir. We make the current internal model of the reservoir visible by probing its prediction $C(x_1, x_2)$ for hypothetical new inputs for evenly spaced points (x_1, x_2) in the entire domain, without allowing it to modify its internal state (otherwise, inputs usually advance the network state according to the dynamics of the network). Figure 4E shows the fast evolution of internal models of the reservoir for the TN during the first trials (visualized for a 1D subset of the 2D input space). One sees that the internal model of the reservoir is from the beginning a smooth function, of the same type as the ones defined by the TNs in $\mathcal{F}$. Within a few trials, this smooth function approximated the TN quite well. Hence ,the reservoir had acquired during the training in the outer loop of L2L a prior for the types of functions that are to be learned, that was encoded in its synaptic weights. This prior was in fact quite efficient, as Fig. 4C, D, E shows, compared to that of a random reservoir. The reservoir was able to learn a TN with substantially fewer trials than a generic learning algorithm for learning the TN directly in an artificial neural network as shown in Fig. 4G: backpropagation with a prior that favored small weights and biases. In this case, the target input was given as feedback to the reservoir throughout the episode, and we compare the training error achieved by the reservoir with that of a FF network trained using backpropagation. A reservoir with a long short-term memory mechanism where we could freeze the memory after low error was achieved allowed us to stop giving the target input after the memory was frozen (results not shown). This long short-term memory mechanism was in the form of neurons with adapting thresholds as described in Bellec et al. (2018a, b). These results suggest that L2L is able to install some form of prior knowledge about the task in the reservoir. We conjectured that the reservoirs fit internal models for smooth functions to the examples it received.

We tested this conjecture in a second, much simpler, L2L scenario. Here, the family $\mathcal{F}$ consisted of all sine functions with arbitrary phase and amplitudes between 0.1 and 5. The reservoir also acquired an internal model for sine functions in this setup from training in the outer loop, as shown in Bellec et al. (2018b). Even when we

selected examples in an adversarial manner, which happened to be in a straight line, this did not disturb the prior knowledge of the reservoir.

Altogether the network learning that was induced through L2L in the reservoir is of particular interest from the perspective of the design of learning algorithms, since we are not aware of previously documented methods for installing structural priors for online learning of a RSNN.

4 Methods

4.1 Leaky Integrate-and-Fire Neurons

We used leaky integrate-and-fire (LIF) models of spiking neurons, where the membrane potential $V_j(t)$ of neuron j evolves according to

$$V_j(t+1) = \rho_j\, V_j(t) + (1-\rho_j)\, R_m\, I_j(t) - B_j(t)\, z_j(t), \tag{6}$$

where R_m is the membrane resistance, ρ_j is the decay constant defined using the membrane time constant τ_j as $\rho_j = e^{\frac{-\Delta t}{\tau_j}}$, and Δt is the time step of simulation. A neuron j spikes as soon at its normalized membrane potential $v_j(t) = \frac{V_j(t)-B_j(t)}{B_j(t)}$ is above its firing threshold v_{th}. At each spike time t, the membrane potential $V_j(t)$ is reset by subtracting the current threshold value $B_j(t)$. After each spike, the neuron enters a strict refractory period during which it cannot spike.

4.2 Backpropagation Through Time

We introduced a version of backpropagation through time (BPTT) in Bellec et al. (2018b) which allows us to back-propagate the gradient through the discontinuous firing event of spiking neurons. The firing is formalized through a binary step function H applied to the scaled membrane voltage $v(t)$. The gradient is propagated through this step function with a pseudo-derivative as in Courbariaux et al. (2016) and Esser et al. (2016), but with a dampened amplitude at each spike.

Specifically, the derivative of the spiking $z_j(t)$ with respect to the normalized membrane potential $v_j(t) = \frac{V_j(t)-B_j(t)}{B_j(t)}$ is defined as

$$\frac{dz_j(t)}{dv_j(t)} := \gamma \max\{0,\, 1 - |v_j(t)|\}. \tag{7}$$

In this way, the architecture and parameters of an RSNN can be optimized for a given computational task.

4.3 Optimizing Reservoirs to Learn

Reservoir model: Our reservoir consisted of 800 recurrently connected leaky integrate-and-fire (LIF) neurons according to the dynamics defined above. The network simulation is carried out in discrete timesteps of $\Delta t = 1$ ms. The membrane voltage decay was uniform across all neurons and was computed to correspond to a time constant of 20 ms ($\rho_j = 0.368$). The normalized spike threshold was set to 0.02 and a refractory period of 5 ms was introduced. Synapses had delays of 5 ms. In the beginning of the experiment, input W^{in} and recurrent weights W^{rec} were initialized according to the Gaussian distributions with zero mean and standard deviations of $\frac{1}{\sqrt{3}}$ and $\frac{1}{\sqrt{800}}$, respectively. Similarly, the initial values of the readout $W^{\text{out,init}}$ were also optimized in the outer loop and were randomly initialized at the beginning of the experiment according to a uniform distribution, as proposed in Glorot and Bengio (2010).

Readout learning: The readout was iteratively adapted according to Eq. (2). It received as input the input $\boldsymbol{x}_C(t)$ itself and the features $\boldsymbol{h}_C(t)$ from the reservoir, which were given as exponentially filtered spike trains: $h_{C,j}(t) = \sum_{t' \leq t} \kappa^{t-t'} z_{C,j}(t')$. Here, $\kappa = e^{\frac{-\Delta t}{\tau_{\text{readout}}}}$ is the decay of leaky readout neurons. Weight changes were computed at each timestep and accumulated. After every second, these changes were used to actually modify the readout weights. Thus, formulated in discrete time, the plasticity of the readout weights in a task C took the following form:

$$\Delta W_C^{\text{out}} = \eta \sum_{t'=t-1000\,\text{ms}}^{t} \left(\boldsymbol{y}_C(t') - \widehat{\boldsymbol{y}}_C(t') \right) \cdot \boldsymbol{h}_C(t')^T, \tag{8}$$

where η is a learning rate.

Outer-loop optimization: To optimize input and recurrent weights of the reservoir in the outer loop, we simulated the learning procedure described above for $m = 40$ different tasks in parallel. After each 3 s, the simulation was paused and the outer-loop objective was evaluated. Note that the readout weights were updated 3 times within these 3 s according to our scheme. The outer-loop objective, as given in Eq. 3, is approximated by

$$\mathcal{L} = \frac{1}{m} \sum_{n=1}^{m} \sum_{t'=t-2000\,\text{ms}}^{t} \left\| \boldsymbol{y}_n(t') - \widehat{\boldsymbol{y}}_n(t') \right\|_2^2 + \mathcal{L}_{\text{reg}} \,. \tag{9}$$

We found that learning is improved if one includes only the last two seconds of simulation. This is because the readout weights seem fixed and unaffected by the plasticity of Eq. 2 in the first second, as BPTT cannot see beyond the truncation of 3 s. The cost function $\mathcal{L}$ was then minimized using a variant of gradient descent (Adam (Kingma and Ba 2014)), where a learning rate of 0.001 was used. The required

gradient $\nabla\mathcal{L}$ was computed with BPTT using the 3 s chunks of simulation and was clipped if the $\mathcal{L}_2$-norm exceeded a value of 1000.

Regularization: In order to encourage the model to settle into a regime of plausible firing rates, we add to the outer-loop cost function a term that penalizes excessive firing rates:

$$\mathcal{L}_{\text{reg}} = \alpha \sum_{j=1}^{800} (f_j - 20\,\text{Hz})^2, \tag{10}$$

with the hyperparameter $\alpha = 1200$. We compute the firing rate of a neuron f_j based on the number of spikes in the past 3 s.

Task details: We describe here the procedure according to which the input time series $\boldsymbol{x}_C(t)$ and target time series $\boldsymbol{y}_C(t)$ were generated. The input signal was composed of a sum of two sines with random phase $\phi_n \in [0, \frac{\pi}{2}]$ and amplitude $A_n \in [0.5, 1]$, both sampled uniformly in the given interval.

$$\boldsymbol{x}_C(t) = \sum_{n=1}^{2} A_n \sin\left(2\pi \frac{t}{T_n} + \phi_n\right), \tag{11}$$

with periods of $T_1 = 0.323$ s and $T_2 = 0.5$ s.

The corresponding target function $\boldsymbol{y}_C(t)$ was then computed by an application of a random second-order Volterra filter to $\boldsymbol{x}_C(t)$ according to Eq. (5). Each task uses a different kernel in the Volterra filter and we explain here the process by which we generate the kernels k^1 and k^2. Recall that we truncate the kernels after a time lag of 500 ms. Together with the fact that we simulate in discrete time steps of 1 ms, we can represent k^1 as a vector with 500 entries and k^2 as a matrix of dimension 500×500.

Sampling k^1: We parametrize k^1 as a normalized sum of two different exponential filters with random properties:

$$\tilde{k}^1(t) = \sum_{n=1}^{2} a_n \exp\left(-\frac{t}{b_n}\right), \tag{12}$$

$$k^1(t) = \frac{\tilde{k}^1(t)}{\|\tilde{k}^1\|_1}, \tag{13}$$

with a_n being sampled uniformly in $[-1, 1]$, and b_n drawn randomly in [0.1 s, 0.3 s]. For normalization, we use the sum of all entries of the filter in the discrete representation ($t \in \{0, 0.001, 0.002, \ldots, 0.499\}$).

Sampling k^2: We construct k^2 to resemble a Gaussian bell shape centered at $t = 0$, with a randomized "covariance" matrix Σ, which we parametrize such that we always obtain a positive definite matrix:

$$\Sigma = \begin{bmatrix} \sqrt{1+u^2+v^2}+u & v \\ v & \sqrt{1+u^2+v^2}-u \end{bmatrix}, \tag{14}$$

where u, v are sampled uniformly in $[-12, 12]$. With this, we defined the kernel k^2 according to

$$\tilde{k}^2(t_1, t_2) = \exp\left(-\frac{1}{24}\begin{bmatrix} t_1, t_2 \end{bmatrix} \Sigma^{-1} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}\right), \tag{15}$$

$$k^2(t_1, t_2) = \frac{\tilde{k}^2(t_1, t_2)}{\|\tilde{k}^2\|_1} \cdot 14. \tag{16}$$

The normalization term here is again given by the sum of all entries of the matrix in the discrete time representation ($[t_1, t_2] \in \{0, 0.001, 0.002, \ldots, 0.499\}^2$).

4.4 Reservoirs Can Also Learn Without Changing Synaptic Weights to Readout Neurons

Reservoir model: The reservoir model used here was the same as that in Sect. 4.3, but with 300 neurons.

Input encoding: Analog values were transformed into spiking trains to serve as inputs to the reservoir as follows: For each input component, 100 input neurons are assigned values $m_1, \ldots m_{100}$ evenly distributed between the minimum and maximum possible value of the input. Each input neuron has a Gaussian response field with a particular mean and standard deviation, where the means are uniformly distributed between the minimum and maximum values to be encoded, and with a constant standard deviation. More precisely, the firing rate r_i (in Hz) of each input neuron i is given by $r_i = r_{\max} \exp\left(-\frac{(m_i - z_i)^2}{2\sigma^2}\right)$, where $r_{\max} = 200$ Hz, m_i is the value assigned to that neuron, z_i is the analog value to be encoded, and $\sigma = \frac{(m_{\max} - m_{\min})}{1000}$, $m_{\min}$ with $m_{\max}$ being the minimum and maximum values to be encoded.

Setup and training schedule: The output of the reservoir was a linear readout that received as input the mean firing rate of each of the neurons per step, i.e., the number of spikes divided by 20 for the 20 ms time window that constitutes a step.

The network training proceeded as follows: A new target function was randomly chosen for each *episode* of training, i.e., the parameters of the target function are chosen uniformly randomly from within the ranges above.

Each *episode* consisted of a sequence of 400 *steps*, each lasting for 20 ms. In each step, one training example from the current function to be learned was presented to the reservoir. In such a step, the inputs to the reservoir consisted of a randomly chosen vector $\mathbf{x} = (x_1, x_2)$ as described earlier. In addition, at each step, the reservoir also got the target value $C(x_1', x_2')$ from the previous step, i.e., the value of the target

calculated using the target function for the inputs given at the previous step (in the first step, $C(x_1', x_2')$ is set to 0). The previous target input was provided to the reservoir during all steps of the episode.

All the weights of the reservoir were updated using our variant of BPTT, once per *iteration*, where an *iteration* consisted of a batch of 10 *episodes*, and the weight updates were accumulated across episodes in an iteration. The ADAM (Kingma and Ba 2014) variant of gradient descent was used with standard parameters and a learning rate of 0.001. The loss function for training was the mean-squared error (MSE) of the predictions over an iteration (i.e., over all the steps in an episode and over the entire batch of episodes in an iteration), with the optimization problem written as

$$\min_{\Theta} \ \mathbb{E}_{C\sim\mathcal{F}}\left[\sum_{t}\left(C(x_1^t, x_2^t; \Theta) - \widehat{C}(x_1^t, x_2^t; \Theta)\right)^2\right]. \tag{17}$$

In addition, a regularization term was used to maintain a firing rate of 20 Hz as in Eq. 10, with $\alpha = 30$. In this way, we induce the reservoir to use sparse firing. We trained the reservoir for 5000 iterations.

Parameter values: The parameters of the leaky integrate-and-fire neurons were as follows: 5 ms neuronal refractory period, delays spread uniformly between 0 and 5 ms, membrane time constant $\tau_j = \tau = 20$ms ($\rho_j = \rho = 0.368$) for all neurons j, and $v_{\text{th}} = 0.03$ V baseline threshold voltage. The dampening factor for training was $\gamma = 0.4$ in Eq. 7.

Comparison with Linear baseline: The linear baseline was calculated using linear regression with L2 regularization with a regularization factor of 100 (determined using grid search), using the mean spiking trace of all the neurons. The mean spiking trace was calculated as follows: First, the neuron traces were calculated using an exponential kernel with 20 ms width and a time constant of 20 ms. Then, for every step, the mean value of this trace was calculated to obtain the mean spiking trace. In Fig. 4B, for each episode consisting of 400 steps, the mean spiking trace from a subset of 320 steps was used to train the linear regressor, and the mean spiking trace from the remaining 80 steps was used to calculate the test error. The reported baseline is the mean of the test error over one batch of 1000 episodes with error bars of one standard deviation.

The total test MSE was 0.0056 ± 0.0039 (linear baseline MSE was 0.0217 ± 0.0046) for the TN task.

Comparison with random reservoir: In Fig. 4B, C, a reservoir with randomly initialized input, recurrent, and readout weights was tested in the same way as the optimized reservoir—with the same sets of inputs, and without any synaptic plasticity in the inner loop. The plotted curves are the average of over 8000 different TNs.

Comparison with backprop: The comparison was done for the case where the reservoir was trained on the function family defined by target networks. A feed-forward (FF) network with 10 hidden neurons and 1 output was constructed. The

input to this FF network were the analog values that were used to generate the spiking input and targets for the reservoir. Therefore, the FF had 2 inputs, one for each of x_1 and x_2. The error reported in Fig. 4G is the mean training error over 1000 TNs with error bars of one standard deviation.

The FF network was initialized with the Xavier normal initialization (Glorot and Bengio 2010) (which had the best performance, compared to the Xavier uniform and plain uniform between $[-1, 1]$). Adam (Kingma and Ba 2014) with AMSGrad (Reddi et al. 2018) was used with parameters $\eta = 10^{-1}$, $\beta_1 = 0.7$, $\beta_2 = 0.9$, $and C = 10^{-5}$. These were the optimal parameters as determined by a grid search. Together with the Xavier normal initialization and the weight regularization parameter C, the training of the FF favored small weights and biases.

5 Discussion

We have presented a new form of reservoir computing, where the reservoir is optimized for subsequent fast learning of any particular task from a large—in general even infinitely large—family of possible tasks. We adapted for that purpose the well-known L2L method from machine learning. We found that for the case of reservoirs consisting of spiking neurons, this two-tier process does in fact enhance subsequent reservoir learning performance substantially in terms of precision and speed of learning. We propose that similar advantages can be gained for other types of reservoirs, e.g., recurrent networks of artificial neurons or physical embodiments of reservoirs (see Tanaka et al. 2019 for a recent review) for which some of their parameters can be set to specific values. If one does not have a differentiable computer model for such a physically implemented reservoir, one would have to use a gradient-free optimization method for the outer loop, such as simulated annealing or stochastic search, see Bohnstingl et al. (2019) for the first step in that direction.

We have explored in Sect. 3 a variant of this method, where not even the weights to readout neurons need to be adapted for learning specific tasks. Instead, the weights of recurrent connections within the reservoir can be optimized so that the reservoir can learn a task from a given family $\mathcal{F}$ of tasks by maintaining learned information for the current task in its working memory, i.e., in its network state. This state may include values of hidden variables such as current values of adaptive thresholds, as in the case of LSNNs (Bellec et al. 2018b). It turns out that L2L without any synaptic plasticity in the inner loop enables the reservoir to learn faster than the optimal learning method from machine learning for the same task: Backpropagation applied directly to the target network architecture which generated the nonlinear transformation, compare panels C and G of Fig. 4. We also have demonstrated in Fig. 4E (and in Bellec et al. 2018b) that the L2L method can be viewed as installing a prior in the reservoir. This observation raises the question of what types of priors or rules can be installed in reservoirs with this approach. For neurorobotics applications, it would be especially important to be able to install safety rules in a neural network controller that cannot

be overridden by subsequent learning. We believe that L2L methods could provide valuable tools for that.

Another open question is whether biologically more plausible and computationally more efficient approximations to BPTT, such as e-prop (Bellec et al. 2019b), can be used instead of BPTT for optimizing a reservoir in the outer loop of L2L. In addition, it was shown in Bellec et al. (2019a) that if one allows that the reservoir adapts weights of synaptic connections within a recurrent neural network via e-prop, even one-shot learning of new arm movements becomes feasible.

Acknowledgements This research was partially supported by the Human Brain Project, funded from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 720270 (Human Brain Project SGA1) and under the Specific Grant Agreement No. 785907 (Human Brain Project SGA2). The research leading to these results has in parts been carried out on the Human Brain Project PCP Pilot Systems at the Jülich Supercomputing Centre, which received co-funding from the European Union (Grant Agreement No. 604102). We gratefully acknowledge Sandra Diaz, Alexander Peyser, and Wouter Klijn from the Simulation Laboratory Neuroscience of the Jülich Supercomputing Centre for their support.

References

W.C. Abraham, M.F. Bear, Metaplasticity: the plasticity of synaptic plasticity. Trends Neurosci. **19**(4), 126–130 (1996)

G. Bellec, D. Salaj, A. Subramoney, C. Kraisnikovic, R. Legenstein, W. Maass, Slow dynamic processes in spiking neurons substantially enhance their computing capability (2018a). In preparation

G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, W. Maass, Long short-term memory and learning-to-learn in networks of spiking neurons, in *Advances in Neural Information Processing Systems*, vol. 31, ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Curran Associates, Inc., 2018b), pp. 795–805

G. Bellec, F. Scherr, E. Hajek, D. Salaj, R. Legenstein, W. Maass, Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets (2019a), arXiv:1901.09049 [cs]

G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, W. Maass, A solution to the learning dilemma for recurrent networks of spiking neurons (2019b). *bioRxiv*, p. 738385. 00000

T. Bohnstingl, F. Scherr, C. Pehle, K. Meier, W. Maass, Neuromorphic hardware learns to learn. Front. Neurosci. **13**, 483 (2019)

M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, Y. Bengio, Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or -1 (2016), arXiv:1602.02830

Y. Duan, J. Schulman, X. Chen, P.L. Bartlett, I. Sutskever, P. Abbeel, RL^2: Fast reinforcement learning via slow reinforcement learning (2016), arXiv:1611.02779

S.K. Esser, P.A. Merolla, J.V. Arthur, A.S. Cassidy, R. Appuswamy, A. Andreopoulos, D.J. Berg, J.L. McKinstry, T. Melano, D.R. Barch, C. di Nolfo, P. Datta, A. Amir, B. Taba, M.D. Flickner, D.S. Modha, Convolutional networks for fast, energy-efficient neuromorphic computing. Proc. Natl. Acad. Sci. **113**(41), 11441–11446 (2016)

X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (2010), pp. 249–256

S. Haeusler, W. Maass, A statistical analysis of information-processing properties of lamina-specific cortical microcircuit models. Cereb. Cortex **17**(1), 149–162 (2006)
S. Hochreiter, A.S. Younger, P.R. Conwell, Learning to learn using gradient descent, in *International Conference on Artificial Neural Networks* (Springer, 2001), pp. 87–94
H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. Technical Report 148:34, German National Research Center for Information Technology GMD, Bonn, Germany (2001)
D.P. Kingma, J. Ba, Adam: a method for stochastic optimization (2014), arXiv:1412.6980
W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. **14**(11), 2531–2560 (2002)
W. Maass, T. Natschläger, H. Markram, Fading memory and kernel properties of generic cortical microcircuit models. J. Physiol.-Paris **98**(4–6), 315–330 (2004)
M.G. Perich, J.A. Gallego, L. Miller, A neural population mechanism for rapid learning. Neuron 964–976.e7 (2018)
S.J. Reddi, S. Kale, S. Kumar, On the convergence of Adam and beyond, in *International Conference on Learning* (2018) (Representations)
G. Tanaka, T. Yamane, J.B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, A. Hirose, Recent advances in physical reservoir computing: a review. Neural Netw. **115**, 100–123 (2019)
D. Verstraeten, B. Schrauwen, M. D'Haene, D. Stroobandt, An experimental unification of reservoir computing methods. Neural Netw. **20**(3), 391–403 (2007). Echo State Networks and Liquid State Machines
V. Volterra, *Theory of functionals and of integral and integro-differential equations* (2005). Courier Corporation
J.X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J.Z. Leibo, R. Munos, C. Blundell, D. Kumaran, M. Botvinick, Learning to reinforcement learn (2016), arXiv:1611.05763
J.X. Wang, Z. Kurth-Nelson, D. Kumaran, D. Tirumala, H. Soyer, J.Z. Leibo, D. Hassabis, M. Botvinick, Prefrontal cortex as a meta-reinforcement learning system. Nat. Neurosci. **21**(6), 860–868 (2018)

Deep Reservoir Computing

Claudio Gallicchio and Alessio Micheli

Abstract This chapter surveys the recent advancements on the extension of Reservoir Computing toward deep architectures, which is gaining increasing research attention in the neural networks community. Within this context, we focus on describing the major features of Deep Echo State Networks based on the hierarchical composition of multiple reservoirs. The intent is to provide a useful reference to guide applications and further developments of this efficient and effective class of approaches to deal with times-series and more complex data within a unified description and analysis.

Keywords Deep reservoir computing · Deep Echo State Networks · Deep Recurrent Neural Networks

1 Introduction

In recent years, the study of deep neural network architectures for temporal data has been an attractive area of research in the neural networks community (Angelov and Sperduti 2016; Goodfellow et al. 2016; Schmidhuber 2015). Investigations in the field of hierarchically organized Recurrent Neural Networks (RNNs) showed that deep RNNs are able to develop internal states that are multiple time-scale representations of the temporal information. This is a much desired feature, e.g., when approaching complex tasks especially in domains related to human cognition, like speech and text processing (Graves et al. 2013; Hermans and Schrauwen 2013). In addition, the interest in studying hierarchical RNN models finds strong motivation also from the different, but related, perspective of computational neuroscience, from which we know that a "deep" hierarchical organization of recurrent neural units is a major

C. Gallicchio (✉) · A. Micheli
Department of Computer Science, University of Pisa, Largo Bruno Pontecorvo 3, 56127 Pisa, Italy
e-mail: gallicch@di.unipi.it

A. Micheli
e-mail: micheli@di.unipi.it

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_4

pattern in the neocortex (e.g., Churchland and Sejnowski 1992; Gerstner and Kistler 2002). In this sense, information processing in deep RNN architectures has a strong biological motivation.

Recently, within the umbrella of randomized neural network approaches (Gallicchio and Scardapane 2020; Gallicchio et al. 2017a, 2018d; Scardapane and Wang 2017), the Reservoir Computing (RC) (Lukoševičius and Jaeger 2009; Verstraeten et al. 2007) paradigm offered a novel perspective to the analysis and design of deep RNNs. In particular, in the context of discrete-time reservoirs, the introduction of the Deep Echo State Network (DeepESN) model (Gallicchio and Micheli 2016; Gallicchio et al. 2017b) has allowed the study of the properties of layered RNN architectures separately from the learning aspects. Remarkably, such studies pointed out that the structured state space organization with multiple time-scale dynamics in deep RNNs is *intrinsic* to the nature of compositionality of recurrent neural models. The interest in the study of the DeepESN model is hence twofold. On the one hand, sheds light on the intrinsic properties of state dynamics of layered RNNs (Gallicchio and Micheli 2017a, 2018c; Gallicchio et al. 2018c). On the other hand, it enables the design of efficiently trained deep neural networks for temporal data, capable of improving on previous state-of-the-art results in complex tasks (Gallicchio et al. 2018b).

From a historical perspective, before the explicit introduction of the DeepESN model in Gallicchio et al. (2017b), preliminary studies on hierarchical RC models targeted ad-hoc constructed architectures, where different modules were trained for the discovery of temporal features at different scales on synthetic data (Jaeger 2007). Moreover, ad-hoc constructed modular networks made up of multiple ESN modules have also been investigated in the speech processing area (Triefenbach et al. 2010, 2013). More recently, the advantages of multilayered RC networks have been experimentally studied on time-series benchmarks in the RC area (Malik et al. 2017). Different from the above mentioned works, the studies on DeepESN considered in the following aim to address some fundamental questions pertaining to the true nature of layering as a factor of architectural RNN design (Gallicchio and Micheli 2018c). Such basic questions can be essentially summarized as follows:

(i) Why stacking layers of recurrent units?
(ii) What is the inherent architectural effect of layering in RNNs (independently from learning)?
(iii) Can we extend the advantages of depth in RNN design using efficiently trained RC approaches?
(iv) Can we exploit the insights from such analysis to address the automatic design of deep recurrent models (including fundamental parameters such as the architectural form, the number of layers, the number of units in each layer, etc.)?

This chapter is intended both to draw a line of recent developments in response to the above mentioned key research questions and to provide an up-to-date overview on the progress and on the perspectives in the studies of DeepESNs. The rest of this contribution is organized as follows. The DeepESN model is introduced and

discussed in Sect. 2, both from the architectural and the dynamical system viewpoints. The progressive advances in the study of DeepESN field are summarized in Sect. 3, while further developments related to other hierarchical reservoir models are recalled in Sect. 4. Finally, conclusions are drawn in Sect. 5.

2 Deep Echo State Network

This section is intended to provide an introduction to the major characteristics of deep RC models. In particular, we focus on discrete-time reservoir systems, i.e., we frame our analysis adopting the formalism of Echo State Networks (ESNs) (Jaeger 2001; Jaeger and Haas 2004). In this context, we illustrate the main properties of deep reservoir architectures in Sect. 2.1, while in Sect. 2.2 we analyze the behavior of deep reservoirs from the point of view of dynamical systems.

2.1 Architecture

As for the standard shallow ESN model (Jaeger 2001; Jaeger and Haas 2004), a DeepESN (Gallicchio et al. 2017b) is composed by a dynamical *reservoir* system, which embeds the input history into a rich state representation, and by a feed-forward *readout* part, which exploits the state encoding provided by the reservoir to compute the output. Crucially, the reservoir of a DeepESN is organized into a *hierarchy of stacked recurrent layers*, where the output of each layer acts as input for the next one. At each time step t, the state computation proceeds by following the pipeline of recurrent layers, from the first one, which is directly fed by the external input, up to the highest one in the reservoir architecture (i.e., the farthest one from the external input). The layered reservoir architecture of a DeepESN is illustrated in Fig. 1. In our notation, we use N_U to denote the external input dimension, N_L to indicate the number of reservoir layers, and we assume, for the only sake of simplicity, that each reservoir layer has N_R recurrent units. Moreover, we use $\mathbf{u}(t) \in \mathbb{R}^{N_U}$ to denote the external input at time step t, while $\mathbf{x}^{(i)}(t) \in \mathbb{R}^{N_R}$ is the state of the reservoir layer i at time step t. In general, we use the superscript (i) to indicate that an item is related to the ith reservoir in the stack. At each time step t, the composition of the states in all the reservoir layers, i.e. $\mathbf{x}(t) = \left(\mathbf{x}^{(1)}(t), \ldots, \mathbf{x}^{(N_L)}(t)\right) \in \mathbb{R}^{N_R N_L}$, gives the global state of the network.

Assuming leaky integrator reservoir units (Jaeger et al. 2007) in each layer and omitting the bias terms for the ease of notation, the state transition functions in the reservoir layers can be described as follows: For the first layer, we have

$$\mathbf{x}^{(1)}(t) = (1 - a^{(1)})\mathbf{x}^{(1)}(t-1) + a^{(1)}\mathbf{f}(\mathbf{W}^{(1)}\mathbf{u}(t) + \hat{\mathbf{W}}^{(1)}\mathbf{x}^{(1)}(t-1)), \qquad (1)$$

while for successive layers $i > 1$ the state update is given by

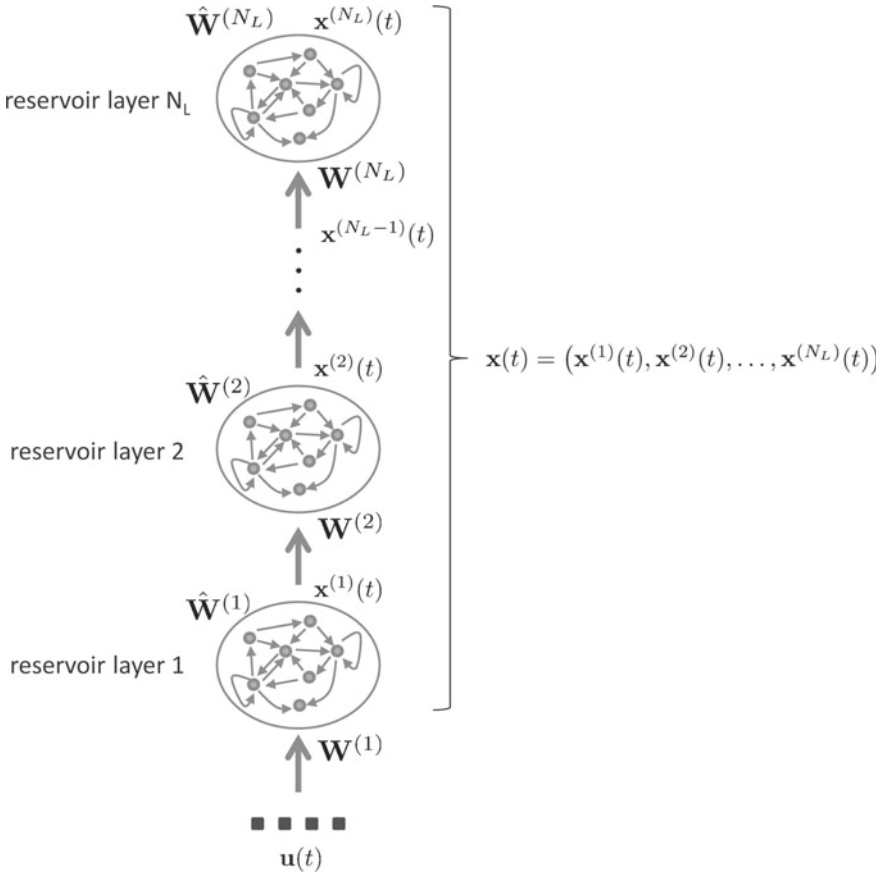


Fig. 1 Reservoir architecture of a Deep Echo State Network

$$\mathbf{x}^{(i)}(t) = (1 - a^{(i)})\mathbf{x}^{(i)}(t-1) + a^{(i)}\mathbf{f}(\mathbf{W}^{(i)}\mathbf{x}^{(i-1)}(t) + \hat{\mathbf{W}}^{(i)}\mathbf{x}^{(i)}(t-1)). \quad (2)$$

In the above Eqs. 1 and 2, $\mathbf{W}^{(1)} \in \mathbb{R}^{N_R \times N_U}$ denotes the input weight matrix, $\mathbf{W}^{(i)} \in \mathbb{R}^{N_R \times N_R}$ (for $i > 1$) is the weight matrix for inter-layer connections from layer $(i-1)$ to layer i, $\hat{\mathbf{W}}^{(i)} \in \mathbb{R}^{N_R \times N_R}$ is the recurrent weight matrix for layer i, $a^{(i)} \in [0, 1]$ is the leaking rate for layer i and $\mathbf{f}$ denotes the element-wise applied activation function for the recurrent reservoir units (typically, the $tanh$ non-linearity is used).

Remark 1 In light of the mathematical description introduced in Eqs. 1 and 2, we can see that the standard (shallow) ESN model can be seen as a special case of DeepESN, obtained whenever a single reservoir layer is considered, i.e., for $N_L = 1$.

Interestingly, as graphically illustrated in Fig. 2, we can observe that the reservoir architecture of a DeepESN can be characterized, with respect to the shallow counterpart, by interpreting it as a constrained version of standard shallow ESN/RNN with

the same total number of recurrent units. In particular, the following constraints are applied in order to obtain a layered architecture:

- all the connections from the input layer to reservoir layers at a level higher than 1 are removed (influencing the way in which the external input information is seen by recurrent units progressively more distant from the input layer);
- all the connections from higher layers to lower ones are removed (which affects the flow of information and the dynamics of sub-parts of the network's state);
- all the connections from each layer to higher layers different from the next one in the pipeline are removed (which affects the flow of information and the dynamics of sub-parts of the network's state).

The above mentioned constraints, that graphically correspond to layering, have been explicitly and extensively discussed in our previous work in Gallicchio et al. (2017b). Under this point of view, the DeepESN architecture can be seen as a simplification of the corresponding single-layer ESN, leading to a reduction in the absolute number of recurrent weights which, assuming full-connected reservoirs at each layer, is quadratic in both the number of recurrent units per layer and the total number of layers (Gallicchio et al. 2018c). As detailed in the above points, however, note that this peculiar architectural organization influences the way in which the temporal information is processed by the different sub-parts of the hierarchical reservoir, composed of recurrent units that are progressively more distant from the external input.

Furthermore, different from the case of a standard ESN/RNN, the state information transmission between consecutive layers in a DeepESN presents no temporal delays. In this respect, we can make the following considerations:

- the aspect of sequentiality between layers operation is already present and discussed in previous works in the literature on deep RNN (see, e.g., El Hihi et al. 1996; Graves et al. 2013; Hermans and Schrauwen 2013; Schmidhuber 1992), which actually stimulated the investigation on the intrinsic role of layering in such hierarchically organized recurrent network architectures;
- this choice allows the model to process the temporal information at each time step in a "deep" temporal fashion, i.e., through a hierarchical composition of multiple levels of recurrent units;
- in particular, notice that the use of (hyperbolic tangent) non-linearities applied individually to each layer during the state computation does not allow to describe the DeepESN dynamics by means of an equivalent shallow system.

Based on the above observations, a major research question naturally arises and drives the motivation to the studies reported in Sect. 3, i.e., how and to what extent, do the described constraints that rule the layered construction and the hierarchical representation in deep recurrent models have an influence on their dynamics?

As regards the output computation, although different choices are possible for the pattern of connectivity between the reservoir layers and the output module (see, e.g., Hermans and Schrauwen 2013; Pascanu et al. 2014), a typical setting consists of feeding at each time step t the state of all reservoir layers (i.e., the global state of the

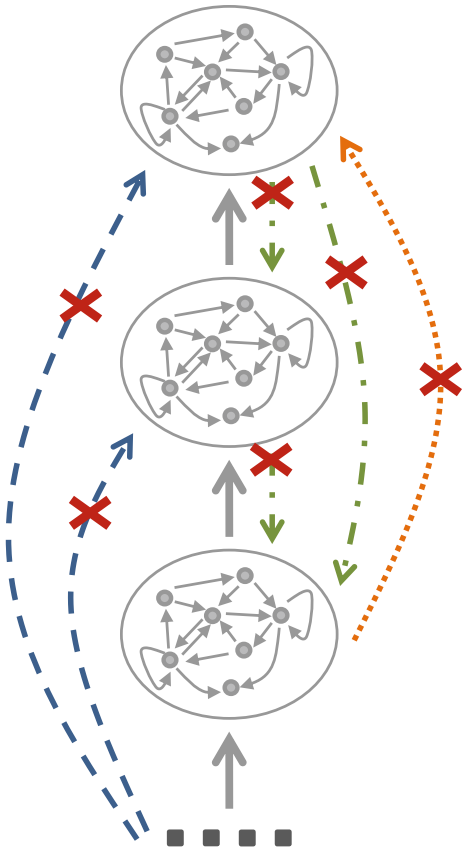


Fig. 2 The layered reservoir architecture of DeepESN as a constrained version of a shallow reservoir. Compared to the shallow case with the same total number of recurrent units, in a stacked DeepESN architecture the following connections are removed: from the input to reservoir levels at height > 1 (blue dashed arrows), from higher to lower reservoir levels (green dash dotted arrows), from each reservoir at level i to all reservoirs at levels higher than $i + 1$ (orange dotted arrows)

DeepESN) to the output layer, as illustrated in Fig. 3. Note that this choice enables the readout component to give different weights to the dynamics developed at different layers, thereby allowing to exploit the potential variety of state representations in the stack of reservoirs. Under this setting, denoting by N_Y the size of the output space, in the typical case of linear readout, the output at time step t is computed as

$$\mathbf{y}(t) = \mathbf{W}_{\text{out}}\mathbf{x}(t) = \mathbf{W}_{\text{out}}\left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N_L)}\right), \tag{3}$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{N_Y \times N_R N_L}$ is the readout weight matrix that is adapted on a training set, typically in closed form through direct methods such as pseudo-inversion or ridge-regression.

As in the standard RC framework, all the reservoir parameters, i.e., the weights in matrices $\mathbf{W}^{(i)}$ and $\hat{\mathbf{W}}^{(i)}$, are left untrained after initialization under stability constraints given by the Echo State Property (Gallicchio and Micheli 2017a; Jaeger 2001; Yildiz et al. 2012). This aspect is related to the analysis of dynamical regimes of stacked reservoir systems, and it is detailed in Sect. 2.2.

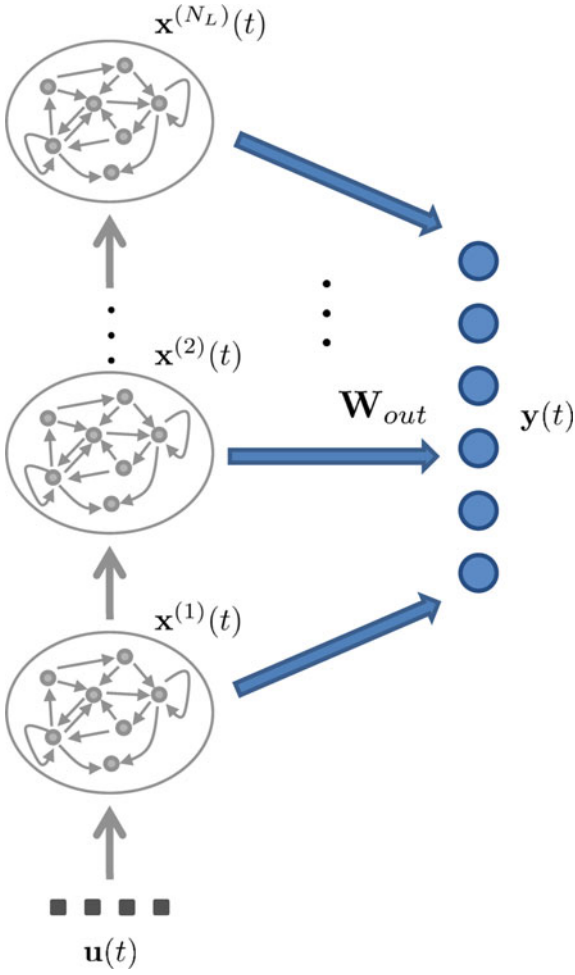


Fig. 3 Readout organization for DeepESN in which at each time step the reservoir states of all layers are used as input for the output layer

2.2 Dynamics of Deep Reservoirs and Echo State Property

The computation carried out by the stack of reservoirs of a DeepESN can be analyzed from a dynamical system viewpoint in terms of input-driven discrete-time non-linear dynamical systems. In particular, we can see that the dynamics of the first layer, driven by the external input, are ruled by a function $F^{(1)}$:

$$\begin{aligned} &F^{(1)} : \mathbb{R}^{N_R} \times \mathbb{R}^{N_U} \rightarrow \mathbb{R}^{N_R} \\ &\mathbf{x}^{(1)}(t) = F^{(1)}(\mathbf{x}^{(1)}(t-1), \mathbf{u}(t)). \end{aligned} \tag{4}$$

The dynamical behavior of each successive layer $i > 1$ is driven by the state of the previous layer in the pipeline, which determines a dependence (through multiple non-linearities) of $\mathbf{x}^{(i)}(t)$ from the states of the hierarchy computed at the previous time step from the first layer up to level i, i.e., $\mathbf{x}^{(1)}(t-1), \ldots, \mathbf{x}^{(i)}(t-1)$, as well as from the input. This is expressed by a function $F^{(i)}$, as follows:

$$\begin{aligned} F^{(i)} &: \underbrace{\mathbb{R}^{N_R} \times \cdots \times \mathbb{R}^{N_R}}_{i \ \ times} \times \mathbb{R}^{N_U} \rightarrow \mathbb{R}^{N_R} \\ \mathbf{x}^{(i)}(t) &= F^{(i)}(\mathbf{x}^{(1)}(t-1), \ldots, \mathbf{x}^{(i)}(t-1), \mathbf{u}(t)). \end{aligned} \tag{5}$$

Note that for both Eqs. 4 and 5, the specific shape of the state transition functions has been described in terms of leaky integrator reservoir units respectively in Eqs. 1 and 2, and are parametrized by the weight values in matrices $\mathbf{W}^{(i)}$ and $\hat{\mathbf{W}}^{(i)}$, for $i = 1, \ldots, N_L$.

When we turn into considering the global state of the DeepESN as the composition of the reservoir states in all the levels of the hierarchy, i.e., $\mathbf{x}(t) = \left(\mathbf{x}^{(1)}(t), \ldots, \mathbf{x}^{(N_L)}(t)\right) \in \mathbb{R}^{N_R N_L}$, we can see that the state dynamics are ruled by a global state transition function F. This function can be defined as a composition of the layer-wise applied functions $F^{(i)}$, i.e., $F = \left(F^{(1)}, \ldots, F^{(N_L)}\right)$. At each time step t, function F computes the next state of the entire deep reservoir system based on the external input information and on the previous state of the deep reservoir, as follows:

$$\begin{aligned} F &: \underbrace{\mathbb{R}^{N_R} \times \cdots \times \mathbb{R}^{N_R}}_{N_L \ \ times} \times \mathbb{R}^{N_U} \rightarrow \underbrace{\mathbb{R}^{N_R} \times \cdots \times \mathbb{R}^{N_R}}_{N_L \ \ times} \\ \mathbf{x}(t) &= F(\mathbf{x}(t), \mathbf{u}(t)) \\ &= \left(F^{(1)}(\mathbf{x}^{(1)}(t-1), \mathbf{u}(t)), \ldots, F^{(N_L)}(\mathbf{x}^{(1)}(t-1), \ldots, \mathbf{x}^{(N_L)}(t-1), \mathbf{u}(t))\right). \end{aligned} \tag{6}$$

As in the case of standard shallow RC architectures, in order to avoid training of the reservoir connections, the state dynamics of a deep reservoir system described by Eq. 6 should exhibit global asymptotic (Lyapunov) stability, as prescribed by the Echo State Property (ESP) (Jaeger 2001; Yildiz et al. 2012). This aspect has been analyzed in detail in Gallicchio and Micheli (2017a), where the well known algebraic conditions for the ESP have been extended to cope with the case of deep reservoirs. Here we recall the statements of Theorems 1 and 2 in Gallicchio and Micheli (2017a), which provide practical means for initialization of DeepESNs. Note that, as in the shallow case, when analyzing the behavior of deep reservoirs we shall assume that both the input space and the reservoir state spaces in all the layers are compact sets.[1]

Theorem 1 (Necessary Condition for the ESP of DeepESN) *Consider a DeepESN whose dynamics are ruled by Eq. 6, implemented in terms of leaky integrator reservoir*

[1] The latter set of conditions is ensured when the reservoir state transition functions in Eqs. 1 and 2 are squashing non-linearities, such is the case, e.g., of tanh.

units as in Eqs. 1 and 2, and assume that the null sequence is an admissible input for the system. Then a necessary condition for the ESP to hold is provided by the following equation:

$$\max_{i=1,\ldots,N_L} \rho^{(i)} = \max_{i=1,\ldots,N_L} \rho\big((1-a^{(i)})\mathbf{I} + a^{(i)}\hat{\mathbf{W}}^{(i)}\big) < 1, \tag{7}$$

where $\rho(\cdot)$ denotes the spectral radius operator (i.e., the maximum absolute eigenvalue of its matrix argument), and $\mathbf{I}$ is the identity matrix of size N_R.

Theorem 2 (Sufficient Condition for the ESP of DeepESN) *Consider a DeepESN whose dynamics are ruled by Eq. 6, implemented in terms of leaky integrator reservoir units as in Eqs. 1 and 2, with* tanh *non-linearity as activation function. If the DeepESN is featured by globally contractive dynamics then it satisfies the ESP. Accordingly, a sufficient condition for the ESP to hold is given by the following equation:*

$$\max_{i=1,\ldots,N_L} C^{(i)} < 1, \tag{8}$$

where $C^{(i)}$ denotes the Lipschitz constant of the state transition function $F^{(i)}$ of the ith reservoir level, and it is computed as follows:

$$C^{(i)} = \begin{cases} (1-a^{(1)}) + a^{(1)}\|\hat{\mathbf{W}}^{(1)}\| & if \;\; i = 1 \\ (1-a^{(i)}) + a^{(i)}\big(C^{(i-1)}\|\mathbf{W}^{(i)}\| + \|\hat{\mathbf{W}}^{(i)}\|\big) & if \;\; i > 1, \end{cases} \tag{9}$$

where $\|\cdot\|$ is the matrix norm induced by the L_2-norm defined on the corresponding state spaces.

The proofs of both Theorems 1 and 2 are given in Gallicchio and Micheli (2017a).

A simple approach to initialize the reservoir weights in DeepESN is then to randomly draw the elements in $\mathbf{W}^{(i)}$ and $\hat{\mathbf{W}}^{(i)}$, e.g., from a uniform distribution in $[-1, 1]$, and then rescale them in order to meet one of the conditions expressed by Theorems 1 or 2. As for standard shallow reservoirs, the sufficient condition is often too restrictive in practice, and the necessary one is commonly adopted in DeepESN applications.

Remark 2 The necessary and the sufficient conditions for the ESP of DeepESN expressed by Theorems 1 and 2 generalize the corresponding conditions for shallow reservoirs given in standard RC literature (Jaeger 2001; Yildiz et al. 2012), respectively obtained from Eqs. 7 and 9 by considering reservoir architectures with just one layer, i.e., for $N_L = 1$.

An interesting insight that we can get from the formulations of the conditions in Theorems 1 and 2, is that adding progressively more reservoir layers to the architecture of a DeepESN can never lower either the degree of stability (max operator in Eq. 7) or the Lipschitz constant (max operator in Eq. 8) of the global deep reservoir system.

This essentially translates into a propensity of deeper recurrent neural systems to show longer memory spans even in the absence of training of the recurrent connections (as observed by several numerical simulations in Gallicchio and Micheli 2017a; Gallicchio et al. 2017b). This insight is also confirmed by more in-depth studies on local Lyapunov exponents of DeepESN states, reported in Gallicchio et al. (2018c). In particular, the results of the analysis given in Gallicchio et al. (2018c) indicate that in conditions of an equal number of recurrent units, deeper reservoir architectures are more easily shifted nearby the edge of stability (or criticality), a dynamical regime close to a stable-unstable transition, where recurrent neural systems are known to develop richer temporal representations of their driving input signals (Legenstein and Maass 2007a, b).

3 Advances

In this section, we briefly survey the recent advances in the study of the DeepESN model. The works described in the following, by addressing the key questions summarized in the Introduction, provide general support to the significance of the DeepESN, and also critically discuss its advantages and drawbacks. An updated overview on the advancements in DeepESN research is also available in Gallicchio and Micheli (2018a).

Multiple time-scale representation. The DeepESN model has been introduced in Gallicchio et al. (2017b), which extends the preliminary work in Gallicchio and Micheli (2016). The analysis provided in these papers revealed, through empirical investigations, the hierarchical structure of temporal data representations developed by the layered reservoir architecture of a DeepESN. Specifically, the stacked composition of recurrent reservoir layers was shown to enable a *multiple time-scale representation* of the temporal information, naturally ordered along the network's hierarchy. Besides, in Gallicchio et al. (2017b), layering proved effective also as a way to enhance the effect of known RC factors of network design, including unsupervised reservoir adaptation by means of Intrinsic Plasticity (Schrauwen et al. 2008). The resulting effects have been analyzed also in terms of state entropy and memory.

Multiple frequency representation. The hierarchically structured state representation in DeepESNs has been investigated by means of *frequency analysis* in Gallicchio et al. (2019b), which specifically considered the case of recurrent units with *linear* activation functions. Results pointed out the intrinsic multiple frequency representation in DeepESN states, where, even in the simplified linear setting, progressively higher layers focus on progressively lower frequencies. In Gallicchio et al. (2019b), the potentiality of the deep RC approach has also been exploited in predictive experiments, showing that DeepESNs outperform state-of-the-art results on the class of Multiple Superimposed Oscillator (MSO) tasks by several orders of magnitude.

Echo State Property. The fundamental RC conditions related to the *Echo State Property* (ESP) have been generalized to the case of deep RC networks in Gallicchio and Micheli (2017a). Specifically, through the study of stability and contractivity of nested dynamical systems, the theoretical analysis in Gallicchio and Micheli (2017a) gives a sufficient condition and a necessary condition for the Echo State Property to hold in case of deep RNN architectures. Remarkably, the work in Gallicchio and Micheli (2017a) provides a relevant conceptual and practical tool for the definition, validity and usage of DeepESN in an "autonomous" (i.e., self-sufficient) way with respect to the standard ESN model.

Local Lyapunov Exponents. The study of DeepESN dynamics under a dynamical system perspective has been pursued in Gallicchio et al. (2017c, 2018c), which provide a theoretical and practical framework for the study of stability of layered recurrent dynamics in terms of *local Lyapunov exponents*. The analysis along this research direction provided interesting insights in terms of the quality of the developed system dynamics, showing the natural beneficial effects due to layering. This aspect is graphically illustrated in Fig. 4, which shows the maximum local Lyapunov exponent (MLLE) of reservoir systems at the increase of the total number of recurrent units, for the case of deep architectures (where the available recurrent units are arranged in layers of 10 units each) and shallow ones (where all the available recurrent units form a single layer). The MLLE is important to characterize the regime of dynamical stability of reservoir systems: values smaller than 0 denote a stable behavior, values greater than 0 indicate instability, and 0 identifies the *edge of stability* (or criticality) condition where dynamical recurrent systems are known to show richer representations of temporal data (Legenstein and Maass 2007a, b). As Fig. 4 indicates, compared to shallow ESN settings in the condition of an equal number of recurrent units, DeepESNs consistently show higher values of the MLLE, closer to the edge of stability, i.e., richer dynamics. The natural enrichment of reservoir quality in deep reservoir settings has an interesting aftereffect in terms of short-term memory ability of the networks. Figure 5 shows the Memory Capacity (MC) score (as defined in Jaeger (2001)) achieved by DeepESN and shallow ESN under the same conditions considered for Fig. 4, indicating the consistent improvement brought by the layered setting. The interested reader can find a more in-depth discussion on the MLLE, MC, and their relation in the context of DeepESNs in Gallicchio et al. (2018c).

Design of Deep Recurrent Neural Networks. The study of the frequency spectrum of deep reservoirs enabled us to address one of the fundamental open issues in deep learning, namely *how to choose the number of layers in a deep RNN architecture*. Starting from the analysis of the intrinsic differentiation of the filtering effects of successive levels in a stacked RNN architecture, the work in Gallicchio et al. (2018b) proposed an automatic method for the *design* of DeepESNs. Noticeably, the proposed approach allows to tailor the DeepESN architecture to the characteristics of the input signals, consistently relieving the cost of the model selection process, and leading to new state-of-the-art results in speech and music processing tasks.

Deep Reservoirs for Structured Domains. A first extension of the deep RC framework for *learning in structured domains* has been presented in Gallicchio and Micheli (2018b, 2019), which introduced the Deep Tree Echo State Network (DeepTESN) model. The new model points out that it is possible to combine the concepts of deep learning, learning for trees and RC training efficiency, taking advantages of the layered architectural organization and from the compositionality of the structured representations both in terms of efficiency and in terms of effectiveness.

In particular, experimental results in Gallicchio and Micheli (2018b, 2019) concretely demonstrate that deep RC models for trees can outperform the accuracy achieved by state-of-the-art approaches for learning in structured domains in challenging problems in the areas of document processing and computational biology, at the same time being extremely advantageous in terms of required training times. Overall, DeepESN provides the first instance of an extremely efficient approach for the design of deep neural networks for learning in cases where the input data is represented in the form of tree structures. Besides, from a theoretical perspective, the work in Gallicchio and Micheli (2019) also provides an in-depth analysis of asymptotic stability of untrained (non-linear) state transition systems operating on discrete tree structures. This results in a generalization of the ESP of conventional reservoirs, proposed under the name of Tree Echo State Property (Gallicchio and Micheli 2019).

The Deep RC approach has been proved extremely advantageous also in the case of learning in domains of graph data, enabling the development of Fast and Deep Graph Neural Networks (FDGNNs) in Gallicchio and Micheli (2020). The concept of reservoirs operating on discrete graph structures has been first introduced in Gallicchio (2010) and revolves around the computation of a state embedding for each vertex in an input graph. In particular, the state for a vertex v is computed as a function of the input information attached to the vertex v itself (i.e., a vector of features that takes the role of external input in the system), and of the state computed for the neighbors of v (a concept that takes the role of "previous time-step" in the case of conventional RC systems for time-series). The stability of the resulting dynamics can be studied by generalizing the mathematical means described in Sect. 2.2, leading to the definition of Graph Embedding Stability (GES), a stability property for neural embedding systems on graphs introduced in Gallicchio and Micheli (2020), to which the interested reader is referred for further information. Besides the introduction of GES, the work in Gallicchio and Micheli (2020) illustrates how to design a deep RC system for graphs, where each layer builds its embedding on the basis of the state information produced in the previous layer. The FDGNN approach was shown to reach (and even outperform) state-of-the-art accuracy on known benchmarks for graph classification, comparing well with many literature approaches, especially based on convolutional neural networks and kernel for graphs. Inheriting the ease of training algorithms from the RC paradigm, the approach is also extremely faster than literature models, enabling a sensible speed-up in the required training times (up to ≈ 3 orders of magnitude in the experiments reported in Gallicchio and Micheli (2020)).

Applications. For what regards the experimental analysis in *applications*, DeepESNs were shown to bring several advantages in both cases of synthetic and real-world tasks. Specifically, DeepESNs outperformed shallow reservoir architectures (under fair conditions on the number of total recurrent units and, as such, on the number of trainable readout parameters) on the Mackey-Glass next-step prediction task (Gallicchio and Micheli 2018c), on the short-term MC task (Gallicchio 2018; Gallicchio et al. 2017b), on MSO tasks (Gallicchio et al. 2019b), as well as on a Frequency Based Classification task (Gallicchio et al. 2018b), purposely designed to assess multiple frequency representation abilities. As pertains to *real-world problems*, the DeepESN approach recently proved effective in a variety of domains, including Ambient Assisted Living (AAL) (Gallicchio and Micheli 2017b), medical diagnosis (Gallicchio et al. 2018a), speech and polyphonic music processing (Gallicchio et al. 2018b, 2019a), metereological forecasting (Alizamir et al. 2020; Kim and King 2020), solar irradiance prediction (Li et al. 2020), energy consumption and wind power generation prediction (Hu et al. 2020), short-term traffic forecasting (Del Ser et al. 2020), destination prediction (Song et al. 2020) car parking and bike-sharing in urban computing (Kim and King 2020), financial market predictions (Kim and King 2020), and industrial applications (for blast furnace off-gas) (Colla et al. 2019; Dettori et al. 2020).

Software. *Software implementations* for DeepESNs applications have recently been made *publicly available*, with references (Gallicchio et al. 2017b, 2018b) representing citation requests for the use of the developed libraries. The DeepESN software is provided in the following forms:

- `Deep Echo State Network (DeepESN) MATLAB Toolbox` available through the MATLAB Add-On Explorer, and directly at the link https://it.mathworks.com/matlabcentral/fileexchange/69402-deepesn;
- `DeepESN Numpy Library (DeepESNpy)` available at https://github.com/lucapedrelli/DeepESN.
- `DeepRC TensorFlow Library (DeepRC-TF)` available at https://github.com/gallicch/DeepRC-TF.

4 Other Hierarchical Reservoir Computing Models

In this section, we briefly summarize further developments in the field of hierarchical RC architectures, successive to the introduction of the DeepESN model. The works described below further contribute to provide an integrated view on the research lines attempting at bridging the gap between the areas of RC and deep learning.

Outside the ESN formalism, hierarchical modularity in reservoir models construction has been recently explored (Zajzon et al. 2018) in the strictly related field of Liquid State Machines (Maass et al. 2002), where reservoirs are implemented using spiking neural networks. In particular, results in Zajzon et al. (2018) indicate that

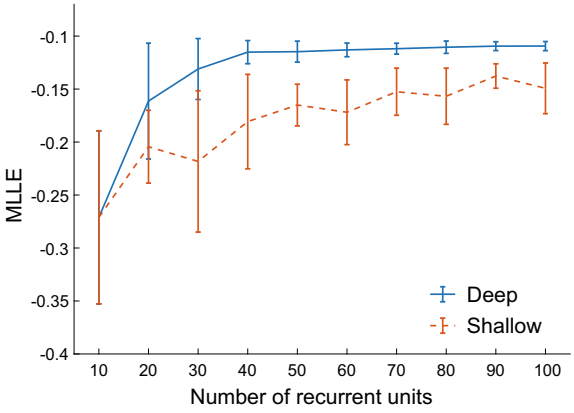


Fig. 4 MLLE of deep and shallow reservoir systems for increasing number of total recurrent units. In the deep case, the available reservoir units are organized into a layered architecture, with 10 units in each layer. In the shallow case, the available reservoir units are arranged into a shallow architecture with a single layer. The values of MLLE are computed as described in Gallicchio et al. (2018c), using the same input time-series, where individual elements were drawn from a uniform distribution in $[-0.8, 0.8]$. Results correspond to a simple network setting, with input scaling $\|\mathbf{W}^{(1)}\| = 1$, and in which all the reservoir layers share the same hyper-parametrization: spectral radius $\rho^{(i)} = 0.9$ (for all layers i) and inter-layer scaling $\|\mathbf{W}^{(i)}\| = 0.5$ (for layers $i > 1$). For each configuration, results are averaged (and the standard deviation is computed) over 10 network guesses (with the same values of hyper-parameters, but different seed for random weights initialization)

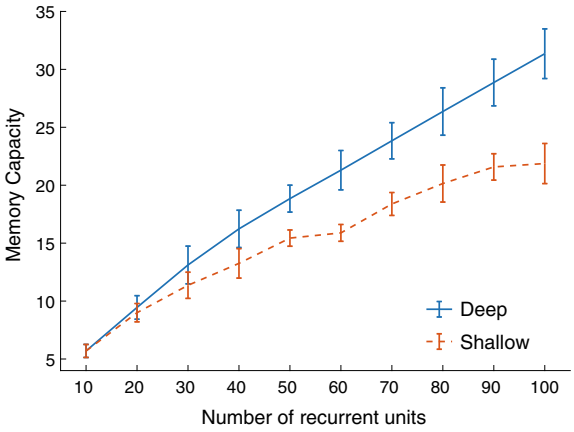


Fig. 5 Average test MC achieved by deep and shallow reservoir systems for increasing number of total recurrent units (the higher the better). In the deep case, the available reservoir units are organized into a layered architecture, with 10 units in each layer. In the shallow case, the available reservoir units are arranged into a shallow architecture with a single layer. The values of MC are computed using the same task settings reported in Gallicchio et al. (2018c), and the same experimental settings considered for Fig. 4

information propagation across the layers might be difficult in deep organizations of spiking neural networks, and a way to overcome this difficulty might consist of using specific patterns of inter-reservoir connectivity in the form of structural mappings with topographic projections between successive layers.

Hierarchical multilayered architectures have been explored also in the context of reservoir systems implemented as cellular automata (CA) (Von Neumann 1996; Wolfram 1984), which can offer some potential advantages *per se*, e.g., in terms of further reductions of computational complexity (see, e.g., Yilmaz 2014). In this context, the preliminary work in Nichele and Molund (2017) shows promising results of 2-layered architectures with CA reservoirs applied to a 5-bit memorization task.

It is known that one of the potentially more costly aspects in the design of RC models is given by the optimization of reservoir hyper-parameters to the specificity of the task at hand. This kind of difficulty, already challenging in the case of shallow reservoirs (Lukoševičius and Jaeger 2009), can be amplified when more reservoir layers are considered. An interesting research line in this concern is thus given by the application of evolutionary algorithms for the optimization of hyper-parameter values of hierarchical reservoirs. A first attempt in this direction is described in Dale (2018), where a steady-state genetic algorithm called Microbial GA (Harvey 2009) is adopted.

A further line of research involves the study of hybrid neural networks architectures that exploit the composition of deep models with shallow RC networks. An instance is represented by the recently introduced Deep Belief ESN (Sun et al. 2017), in which an ESN module is stacked on top of a Deep Belief Network (DBN). The DBN essentially operates a hierarchical non-linear transformation on the input data, while the final output is given by the (shallow) ESN layer. The components of the architecture are trained individually, employing a mixture of unsupervised and supervised learning in which the DBN is trained by contrastive divergence (following a greedy layer-wise approach), and the ESN is trained by pseudo-inversion. Finally, as generally analyzed in the context of deep RNN construction (Pascanu et al. 2014), depth can enter the design of recurrent neural models in several disguises. In the field of RC, we foresee that the synergy between the benefits of modular compositionality both in the recurrent part (deep reservoir) and in the output part (deep readout) could result in breakthrough application results in challenging real-world tasks. The first step along this research direction has been pursued in Bianchi et al. (2018), in which a bidirectional (shallow) reservoir network is coupled with a deep readout component, showing promising results both on benchmark datasets and on a real-world task in the area of medical diagnosis.

Finally, it is worth mentioning recent works that attempt at implementing the DeepESN concept in neuromorphic hardware, especially in photonics, see, e.g., Lugnan et al. (2020), Freiberger et al. (2019).

5 Conclusions

In this chapter, we have provided a brief overview of the extension of the RC approach towards the deep learning framework. Focusing the analysis in the context of discrete-time reservoir models, we have described the salient features of the DeepESN model. Noticeably, DeepESNs enable the analysis of the intrinsic properties of state dynamics in deep RNN architectures, i.e., the study of the bias due to layering in the design of RNNs. At the same time, DeepESNs allow to transfer the striking advantages of the RC methodology to the case of deep recurrent architectures, leading to an efficient approach for designing deep neural networks for temporal data.

The analysis of the distinctive characteristics and dynamical properties of the DeepESN model has been carried out, first empirically, in terms of entropy of state dynamics and system memory. Then, it has been conducted through more abstract theoretical investigations that allowed the derivation of the fundamental conditions for the ESP of deep networks, as well as the characterization of the developed dynamical regimes in terms of local Lyapunov exponents. Besides, studies on the frequency analysis of DeepESN dynamics allowed the development of a grounded algorithm for the automatic setup of (the number of layers of) a DeepESN. Current developments already include model variants and applications to both synthetic and real-world tasks. Besides, extensions of the deep RC approach to learning in structured domains have been introduced with DeepTESN (for trees) and FDGNN (for graphs). Finally, a glimpse of the developments in the study of hierarchical RC-based models successive to the introduction of DeepESN has been provided.

The authors hope that the development of these kinds of architectures and their extensions, in particular towards learning in structured domains, can foster the spreading of RC models both in research and applied fields.

References

M. Alizamir, S. Kim, O. Kisi, M. Zounemat-Kermani, Deep echo state network: a novel machine learning approach to model dew point temperature using meteorological variables. Hydrol. Sci. J. **65**(7), 1173–1190 (2020)

P. Angelov, A. Sperduti, Challenges in deep learning, in *Proceedings of the 24th European Symposium on Artificial Neural Networks (ESANN)* (2016), pp. 489–495, i6doc.com

F.M. Bianchi, S. Scardapane, S. Lokse, R. Jenssen, Bidirectional deep-readout echo state networks, in *Proceedings of the 26th European Symposium on Artificial Neural Networks (ESANN)* (2018), pp. 425–430

P.S. Churchland, T.J. Sejnowski, *The Computational Brain* (The MIT Press, 1992)

V. Colla, I. Matino, S. Dettori, S. Cateni, R. Matino, Reservoir computing approaches applied to energy management in industry, in *International Conference on Engineering Applications of Neural Networks* (Springer, 2019), pp. 66–79

M. Dale, Neuroevolution of hierarchical reservoir computers, in *Proceedings of the Genetic and Evolutionary Computation Conference*. (ACM, 2018), pp. 410–417

J. Del Ser, I. Lana, E.L. Manibardo, I. Oregi, E. Osaba, J.L. Lobo, M.N. Bilbao, E.I. Vlahogianni, Deep echo state networks for short-term traffic forecasting: performance comparison and statistical assessment (2020), arXiv:2004.08170

S. Dettori, I. Matino, V. Colla, R. Speets, Deep echo state networks in industrial applications, in *IFIP International Conference on Artificial Intelligence Applications and Innovations* (Springer, 2020), pp. 53–63

S. El Hihi, Y. Bengio, Hierarchical recurrent neural networks for long-term dependencies, in *Advances in Neural Information Processing Systems (NIPS)* (1996), pp. 493–499

M. Freiberger, S. Sackesyn, C. Ma, A. Katumba, P. Bienstman, J. Dambre, Improving time series recognition and prediction with networks and ensembles of passive photonic reservoirs. IEEE J. Sel. Top. Quantum Electron. **26**(1), 1–11 (2019)

C. Gallicchio, A. Micheli, Graph echo state networks, in *The 2010 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2010), pp. 1–8

C. Gallicchio, Short-term memory of deep RNN, in *Proceedings of the 26th European Symposium on Artificial Neural Networks (ESANN)* (2018), pp. 633–638

C. Gallicchio, A. Micheli, Deep reservoir computing: a critical analysis, in *Proceedings of the 24th European Symposium on Artificial Neural Networks (ESANN)* (2016), pp. 497–502, i6doc.com

C. Gallicchio, A. Micheli, Echo state property of deep reservoir computing networks. Cogn. Comput. **9**(3), 337–350 (2017a)

C. Gallicchio, A. Micheli, Experimental analysis of deep echo state networks for ambient assisted living, in *Proceedings of the 3rd Workshop on Artificial Intelligence for Ambient Assisted Living (AI*AAL 2017), Co-located with the 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017)* (2017b)

C. Gallicchio, A. Micheli, Deep Echo State Network (DeepESN): a brief survey (2018a), arXiv:1712.04323

C. Gallicchio, A. Micheli, Deep tree echo state networks, in *Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2018b), pp. 499–506

C. Gallicchio, A. Micheli, Why layering in recurrent neural networks? a DeepESN survey, in *Proceedings of the 2018 IEEE International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2018c), pp. 1800–1807

C. Gallicchio, A. Micheli, Deep reservoir neural networks for trees. Inf. Sci. **174–193** (2019)

C. Gallicchio, A. Micheli, Fast and deep graph neural networks, in *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)* (2020), pp. 3898–3905

C. Gallicchio, S. Scardapane, Deep randomized neural networks, in *Recent Trends in Learning From Data*. (Springer, 2020), pp. 43–68

C. Gallicchio, J.D. Martin-Guerrero, A. Micheli, E. Soria-Olivas, Randomized machine learning approaches: Recent developments and challenges, in *Proceedings of the 25th European Symposium on Artificial Neural Networks (ESANN)* (2017a), pp. 77–86, i6doc.com

C. Gallicchio, A. Micheli, L. Pedrelli, Deep reservoir computing: a critical experimental analysis. Neurocomputing **268**, 87–99 (2017bb). https://doi.org/10.1016/j.neucom.2016.12.089

C. Gallicchio, A. Micheli, L. Silvestri, Local lyapunov exponents of deep RNN, in *Proceedings of the 25th European Symposium on Artificial Neural Networks (ESANN)* (2017c), pp. 559–564, i6doc.com

C. Gallicchio, A. Micheli, L. Pedrelli, Deep echo state networks for diagnosis of Parkinson's disease, in *Proceedings of the 26th European Symposium on Artificial Neural Networks (ESANN)* (2018a), pp. 397–402

C. Gallicchio, A. Micheli, L. Pedrelli, Design of deep echo state networks. Neural Netw. **108**, 33–47 (2018b)

C. Gallicchio, A. Micheli, L. Silvestri, Local lyapunov exponents of deep echo state networks. Neurocomputing **298**, 34–45 (2018c)

C. Gallicchio, A. Micheli, P. Tiňo, Randomized recurrent neural networks, in *Proceedings of the 26th European Symposium on Artificial Neural Networks (ESANN)* (2018d), pp. 415–424, i6doc.com

C. Gallicchio, A. Micheli, L. Pedrelli, Comparison between DeepESNs and gated RNNs on multivariate time-series prediction, in *Proceedings of the 27th European Symposium on Artificial Neural Networks (ESANN)* (2019a), pp. 619–624

C. Gallicchio, A. Micheli, L. Pedrelli, Hierarchical temporal representation in linear reservoir computing, in *Neural Advances in Processing Nonlinear Dynamic Signals*, ed. by A. Esposito, M. Faundez-Zanuy, F.C. Morabito, E. Pasero (Springer International Publishing, Cham, 2019b), pp. 119–129. https://doi.org/10.1007/978-3-319-95098-3_11, arXiv:1705.05782

W. Gerstner, W.M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity* (Cambridge University Press, Cambridge, 2002)

I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning* (MIT Press, 2016)

A. Graves, A.R. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2013), pp. 6645–6649

I. Harvey, The microbial genetic algorithm, in *European Conference on Artificial Life* (Springer, 2009), pp. 126–133

M. Hermans, B. Schrauwen, Training and analysing deep recurrent neural networks, in *NIPS* (2013), pp. 190–198

H. Hu, L. Wang, S.X. Lv, Forecasting energy consumption and wind power generation using deep echo state network. Renew. Energy **154**, 598–613 (2020)

H. Jaeger, *The "echo state" approach to analysing and training recurrent neural networks - with an erratum note*. Technical report, GMD - German National Research Institute for Computer Science (2001)

H. Jaeger, Short term memory in echo state networks. Technical report, German National Research Center for Information Technology, 2001

H. Jaeger, Discovering multiscale dynamical features with hierarchical echo state networks. Technical report, Jacobs University Bremen (2007)

H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)

H. Jaeger, M. Lukoševičius, D. Popovici, U. Siewert, Optimization and applications of echo state networks with leaky-integrator neurons. Neural Netw. **20**(3), 335–352 (2007)

T. Kim, B.R. King, Time series prediction using deep echo state networks. Neural Comput. Appl. **1–19** (2020)

R. Legenstein, W. Maass, Edge of chaos and prediction of computational performance for neural circuit models. Neural Netw. **20**(3), 323–334 (2007a)

R. Legenstein, W. Maass, What makes a dynamical system computationally powerful. New directions in statistical signal processing: from systems to brain 127–154 (2007b)

Q. Li, Z. Wu, R. Ling, L. Feng, K. Liu, Multi-reservoir echo state computing for solar irradiance prediction: a fast yet efficient deep learning approach. Appl. Soft Comput. **95** (2020)

A. Lugnan, A. Katumba, F. Laporte, M. Freiberger, S. Sackesyn, C. Ma, E. Gooskens, J. Dambre, P. Bienstman, Photonic neuromorphic information processing and reservoir computing. APL Photonics **5**(2) (2020)

M. Lukoševičius, H. Jaeger, Reservoir computing approaches to recurrent neural network training. Comput. Sci. Rev. **3**(3), 127–149 (2009)

W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. **14**(11), 2531–2560 (2002)

Z.K. Malik, A. Hussain, Q.J. Wu, Multilayered echo state machine: a novel architecture and algorithm. IEEE Trans. Cybern. **47**(4), 946–959 (2017)

S. Nichele, A. Molund, Deep learning with cellular automaton-based reservoir computing. Complex Syst. **26**, 319–340 (2017)

R. Pascanu, C. Gulcehre, K. Cho, Y. Bengio, How to construct deep recurrent neural networks (2014), arXiv:1312.6026v5

S. Scardapane, D. Wang, Randomness in neural networks: an overview. Wiley Interdiscip. Rev.: Data Mining Knowl. Discov. **7**(2), e1200 (2017)

J. Schmidhuber, Learning complex, extended sequences using the principle of history compression. Neural Comput. **4**(2), 234–242 (1992)
J. Schmidhuber, Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
B. Schrauwen, M. Wardermann, D. Verstraeten, J. Steil, D. Stroobandt, Improving reservoirs using intrinsic plasticity. Neurocomputing **71**(7), 1159–1171 (2008)
Z. Song, K. Wu, J. Shao, Destination prediction using deep echo state network. Neurocomputing **406**, 343–353 (2020)
X. Sun, T. Li, Q. Li, Y. Huang, Y. Li, Deep belief echo-state network and its application to time series prediction. Knowl.-Based Syst. **130**, 17–29 (2017)
F. Triefenbach, A. Jalalvand, B. Schrauwen, J.P. Martens, Phoneme recognition with large hierarchical reservoirs, in *Advances in Neural Information Processing Systems* (2010), pp. 2307–2315
F. Triefenbach, A. Jalalvand, K. Demuynck, J.P. Martens, Acoustic modeling with hierarchical reservoirs. IEEE Trans. Audio Speech Lang. Process. **21**(11), 2439–2450 (2013)
D. Verstraeten, B. Schrauwen, M. d'Haene, D. Stroobandt, An experimental unification of reservoir computing methods. Neural Netw. **20**(3), 391–403 (2007)
J. Von Neumann, A.W. Burks, *Theory of Self-Reproducing Automata* (University of Illinois Press, Urbana, 1996)
S. Wolfram, Universality and complexity in cellular automata. Phys. D **10**(1–2), 1–35 (1984)
I.B. Yildiz, H. Jaeger, S.J. Kiebel, Re-visiting the echo state property. Neural Netw. **35**, 1–9 (2012)
O. Yilmaz, Reservoir computing using cellular automata (2014), arXiv:1410.0162
B. Zajzon, R. Duartel, A. Morrison, Transferring state representations in hierarchical spiking neural networks, in *Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2018), pp. 1785–1793

On the Characteristics and Structures of Dynamical Systems Suitable for Reservoir Computing

Masanobu Inubushi, Kazuyuki Yoshimura, Yoshiaki Ikeda, and Yuto Nagasawa

Abstract We present an overview of mathematical aspects of Reservoir Computing (RC). RC is a machine learning method suitable for physical implementation, which harnesses a type of synchronization, called Common-Signal-Induced Synchronization. A precise criterion for this synchronization is given by a quantity called the conditional Lyapunov exponent. We describe a class of dynamical systems (physical systems) that are utilizable for RC in terms of this quantity. Then, two notions characterizing the information processing performance of RC are illustrated: (i) Edge of Chaos and (ii) Memory-Nonlinearity Trade-off. Based on the notion (ii), a structure of dynamical systems suitable for RC has been proposed. This structure is called the mixture reservoir. We review the structure and show its remarkable information processing performance.

1 Introduction

Recent developments in computer software, in particular machine learning, have remarkably improved the information processing accuracy such as in speech and image recognition tasks. On the other hand, in a research field of computer hardware, it is expected that an ultrafast computer can be realized based on a novel principle, such as a quantum computer. **Reservoir computing (RC)** is a machine learning method suitable for hardware implementation (Jaeger and Haas 2004). Indeed, many researchers are now actively implementing RC with various physical systems such as optoelectronic systems (Appeltant et al. 2011; Larger et al. 2017; Nakajima et al. 2018; Nakane et al. 2018; Takano et al. 2018), quantum systems (Fujii and Nakajima 2017), and soft materials (Nakajima et al. 2015). It is expected that utilizing physical characteristics for RC appropriately may lead to an innovative "computer" achiev-

M. Inubushi (✉)
Department of Applied Mathematics, Tokyo University of Science, Tokyo 162 - 8601, Japan
e-mail: inubushi@rs.tus.ac.jp

K. Yoshimura · Y. Ikeda · Y. Nagasawa
Department of Information and Electronics, Graduate school of Engineering, Tottori University, Tottori 680 - 8552, Japan

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_5

ing ultrafast processing speed with low energy consumption (see other chapters for various physical implementations).

The physical systems utilized for RC are modeled as **nonlinear dynamical systems**, and dynamical system theory (Morris et al. 2012) and nonlinear physics provide us useful mathematical tools. So far, these tools have enabled us to find some characteristics of RC as we introduce below. Nevertheless, at present, our understanding of the working principles behind RC is far from complete. For instance, we don't have any clear answer to the following fundamental question: "*For a given task, what characteristics and structures of a dynamical system are crucial for RC?*" or "*Does there exist a universal law governing the information processing ability of a dynamical system for RC less dependent on a task?*" If we find clear answers to the above questions, they are not only of theoretical interest but also practically useful since they will give design guidelines of RC. Unfortunately, at the present stage, various physical systems are employed for RC blindly without design guidelines. So, it is quite important to find the answers.

Here, focusing on characteristics and structures of dynamical systems suitable for RC, we overview the mathematical tools and theoretical results reported so far. In Sect. 2, we illustrate a mathematical formulation of RC including its general framework, the surprisingly simple learning method, a class of dynamical systems usable for RC, and a concrete example. In Sect. 3, two notions characterizing the information processing performance of RC are illustrated: (i) Edge of Chaos and (ii) Memory-Nonlinearity Trade-off. Based on the notion (ii), we introduce an effective structure of dynamical systems for RC with some numerical results.

While RC originated from neuroscience, recently a wide variety of physical phenomena are employed experimentally for the implementations. From a theoretical viewpoint, understanding of RC would need a multidisciplinary approach from mathematics, computational science, nonlinear science, neuroscience, and statistical physics. Here, we would like to introduce an aspect of this attractive research field.

2 Mathematical Formulation of Reservoir Computing

First, we formulate the framework of RC in an abstract form, and then, introduce a concrete example. Let a set of two sequences $D = \{s(k), y(k)\}_{k=1}^{K}$ be given, where $s(k)$ and $y(k)$ are one-dimensional input and output signals at time k and there exists some relation between these two sequences. The data set D is referred to as training data. The goal of RC is to learn the relation between s and y from the training data D, and then, after the training period $k > K$, give an estimation $\hat{y}(k)$ for predicting an unknown output $y(k)$ that corresponds to a given new input data $s(k)$. This is a typical supervised learning and the above estimation is called generalization. Considering time series prediction as an example, the goal of RC is to predict $s(k + \tau)$ corresponding to the given input sequence $\{s(i)\}_{i \leq k}$, i.e., $y(k) = s(k + \tau)$, which is called τ-step ahead prediction.

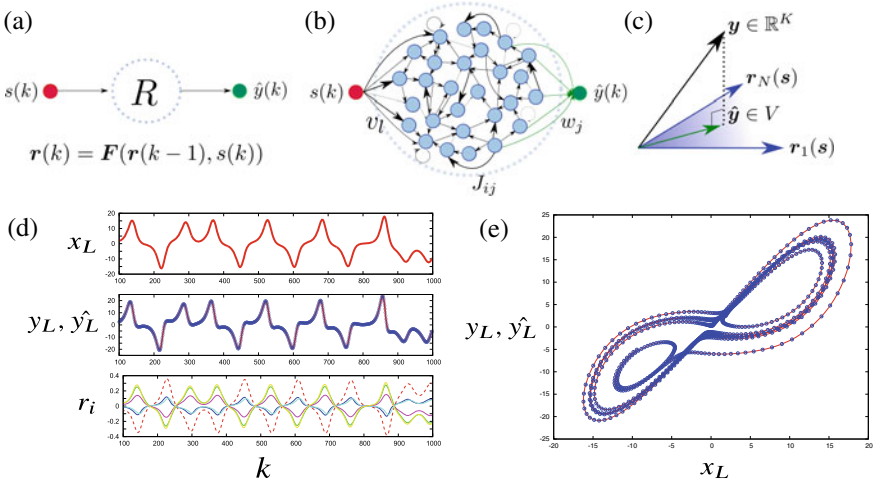


Fig. 1 **a**, **b** Illustration of RC framework. The red circle represents the input node, the blue dot circle represents the reservoir, and the green circle represents the output node. **c** Illustration of the vectors y, $\hat{y}$. **d** A demonstration of RC. The task is to infer the variable $y_L(k)$ of the Lorenz system from a given sequence of the variable $\{\ldots x_L(k-1), x_L(k)\}$ of the same Lorenz system. From top to bottom, the variables x_L, y_L, $\hat{y_L}$ which is inferred by the reservoir, and the time series of the reservoir variable $r_i (i = 1, \ldots, 5)$. **e** The projection of the orbit (the same data as **d**)

RC consists of the input/output nodes and a dynamical system driven by the input sequence $\{s(k)\}$, called *reservoir*. In Fig. 1a, an illustration of the reservoir is shown as "R" encircled by the blue dot circle. The reservoir is described by an N-dimensional dynamical system as follows:

$$\boldsymbol{r}(k) = \boldsymbol{F}(\boldsymbol{r}(k-1), s(k)), \tag{1}$$

where $\boldsymbol{r}(k) \in \mathbb{R}^N$ represents a state of the reservoir at time k which we call reservoir state, and $\boldsymbol{F} : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$ is a map describing the dynamics of the reservoir. In the case of physical implementation, $\boldsymbol{F}$ is determined by the physical law. The reservoir state $\boldsymbol{r}(k) = \{r_i(k)\}_{i=1}^N$ evolves in time according to the map $\boldsymbol{F}$ with the input signal $s(k)$.

The approximation $\hat{y}(k)$ for the desired output $y(k)$ is obtained by "observation" of the reservoir state with linear weights:

$$\hat{y}(k) := \sum_{i=1}^{N} w_i r_i(k). \tag{2}$$

This output weight $\{w_i\}_{i=1}^N$ is optimized so that $\hat{y}(k) \simeq y(k)$ as explained below. The determination of the weight $\{w_i\}_{i=1}^N$ by the training data D is called learning. In short, RC is the information processing method with **learning output weight only**.

In other words, the evolution law, $\boldsymbol{F}$ in the Eq. (1), is not optimized but fixed. Considering the conventional training methods of neural network where both w and $\boldsymbol{F}$ are learned, it may be surprising that information processing can be performed by a task-independent, even randomly generated, evolution law $\boldsymbol{F}$. Since control of physical laws is difficult in general, physical implementations of the conventional neural networks, which require thousands of adjustable parameters and stable controls of them with high precision, are also difficult. On the other hand, the framework of RC does not require such control, and therefore is suitable for physical implementations.

The learning of the linear output weights $\boldsymbol{w}$ is just performed via the least squares method by using the training data D. Let $\boldsymbol{y} = (y(1), \ldots, y(K))^T$ be a sequence of the desired outputs and $\boldsymbol{w} = (w_1, \ldots, w_N)^T$ be a weight vector, and $\Phi_{kj} = r_j(k)$ which is the so-called design matrix. The linear readout (2) can be described as $\hat{\boldsymbol{y}} = \Phi \boldsymbol{w}$. From the condition to minimize the square error $E(w) = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|^2$ where $\|\cdot\|$ is the l_2-norm, we obtain the set of equations $\partial_{w_i} E(\boldsymbol{w}) = 0$ $(i = 1, \ldots, N)$ and its solution as follows:

$$\boldsymbol{w} = (\Phi^T \Phi)^{-1} \Phi^T \boldsymbol{y}. \tag{3}$$

What we need for the learning for RC is just to calculate the inverse matrix of $\Phi^T \Phi$ whose size is the number of the nodes, $N \times N$. Thus, the learning for RC is computationally cheap. While this formulation is similar to the basic least squares method, the difference is that the design matrix has information on the whole of history of the past input signals as $\Phi_{kj} = r_j(k) = r_j(s(k), s(k-1), \ldots)$. For the practically important tasks such as time series prediction or speech recognition, history of the past input signals is essential. Hence, RC enables us to solve the temporal task requiring the past input signals at a low computational cost.

2.1 A Class of Dynamical System Usable for Reservoir Computing: Common-Signal-Induced Synchronization

What kind of a dynamical system can we use for RC? The dynamical system utilized for RC should at least have a property that the same output sequence $\{y(k)\}_{k=1}^T$ is generated when repeatedly given the same input sequence $\{s(k)\}_{k=1}^T$, not depending on the initial condition of the dynamical system. If we use a dynamical system lacking this property, we have different outputs from RC for the same tasks, depending on the initial state of the reservoir. Considering speech recognition tasks, the reservoir computer is useless if it recognizes completely different words for the same voice input.

In order to formulate this property, let us consider two different initial conditions $\boldsymbol{r}(0)$, $\tilde{\boldsymbol{r}}(0)$ of the reservoir states. These states evolve in time according to the map $\boldsymbol{F}$ in Eq. (1) with a common input signal $s(k)$. If these different states always converge to

the same state, not depending on the initial conditions, i.e., $\|\boldsymbol{r}(k) - \tilde{\boldsymbol{r}}(k)\| \to 0\ (k \to \infty)$, we say the signal-driven dynamical system (1) shows **common-signal-induced synchronization (CSIS)**. In other words, there exists a certain synchronized state $\boldsymbol{r}^*(k)$ evolving in time only depending on the input sequence. The synchronized state attracts states in its neighborhood, i.e., asymptotically stable as mentioned later. The input sequence determines the synchronized state $\boldsymbol{r}^*(k)$ except the initial transient period, and also determines uniquely the output sequence: $\hat{y}(k) = \sum_i w_i r_i^*(k)$. This kind of reproducibility to the input signal is also referred to as *Echo State Property* (Maass and Markram 2002) or *Consistency* (Uchida et al. 2004). Any dynamical system possessing this property can be used for reservoir computing in principle.

This property is exhibited by not only a particular dynamical system but also general dynamical systems in the following sense. For instance, it can be shown easily that any dynamical system with a stable fixed point as $\boldsymbol{r}(k) = A\boldsymbol{r}(k-1) + \boldsymbol{s}(k)$ $(\|A\| < 1)$ shows CSIS. It has been shown that the CSIS occurs in a variety of limit-cycle oscillators (Teramae and Tanaka 2004) and one-dimensional delay dynamical systems (Yoshimura et al. 2008a) when they are driven by the white noise input. Moreover, for the colored noise, the previous study (Yoshimura et al. 2007) has shown that the CSIS occurs for the various dynamics including the limit cycle, chaotic dynamics, and one-dimensional time delay system. Considering there exist stable fixed points and limit cycles in many dissipative dynamical systems, we can use various physical systems as a reservoir for a resource of information processing.

A mathematical tool from the dynamical system theory, called **conditional Lyapunov exponent**, is useful to characterize CSIS (Louis et al. 1991; Teramae and Tanaka 2004; Yoshimura et al. 2008a, b). The conditional Lyapunov exponent is defined by the exponential growth/decay rate of the infinitesimal perturbation $\delta\boldsymbol{r}(k)$ to the state $\boldsymbol{r}(k)$ as follows:

$$\lambda := \lim_{T\to\infty} \frac{1}{T} \ln \|\delta\boldsymbol{r}(T)\|, \tag{4}$$

where $\delta\boldsymbol{r}(k)$ is determined by the variational equation

$$\delta\boldsymbol{r}(k) = DF_{\boldsymbol{r}(k-1),s(k)}\delta\boldsymbol{r}(k-1) \tag{5}$$

and $DF_{\boldsymbol{r}(k-1),s(k)}$ denotes the Jacobian matrix of $\boldsymbol{F}$ evaluated at $(\boldsymbol{r}(k-1), s(k))$. Equation (4) can be interpreted as $\|\delta\boldsymbol{r}(k)\| \propto \exp(\lambda k)$, and thus $\lambda < 0$ implies the asymptotic stability, i.e., the common-signal-induced synchronization. Note that the conditional Lyapunov exponent is a characteristic value with respect to the invariant measure (distribution) which is determined by not only the dynamical system, the map $\boldsymbol{F}$, but also the stochastic property of the input sequence, $s(k)$. The Lyapunov exponent introduced above is *conditioned* by the stochastic property of $s(k)$, and thus, we say the conditional Lyapunov exponent. The characterization of the CSIS by the conditional Lyapunov exponent has been introduced and often used in the field of nonlinear physics, and recently, the random dynamical system theory in the

field of mathematics has justified it rigorously under some conditions (Flandoli et al. 2017). In conclusion, any signal-driven dynamical system with a negative conditional Lyapunov exponent can be used for RC.

2.2 Concrete Example: Echo State Network Model

Echo State Network (ESN) is one of the standard structures of the reservoir that uses a recurrent neural network as shown in Fig. 1b. Let k $(= 1, 2, \ldots)$ be the time and $r_i(k)$ be the state of the ith node $(i = 1, 2, \ldots, N)$ at time k. The reservoir state evolves in time as follows:

$$r_i(k) = \phi\left[\sum_{j=1}^{N} J_{ij} r_j(k-1) + v_i s(k)\right], \tag{6}$$

where $\phi[\cdot]$ is called the activation function[1] and typically $\phi[u] = \tanh gu$ is used. $g \in \mathbb{R}$ is a parameter. v_i and J_{ij} are connection weights between nodes in the network, and importantly, the values of them are fixed once they are determined by random numbers at the initial. Supervised machine learning is applied only to the output weight $\{w_i\}_{i=1}^{N}$ as noted before.

As an example, we demonstrate a result of an inference task solved by the ESN model in Fig. 1d, e. The task is to infer the variable y_L of the Lorenz equation[2] from another variable x_L of the Lorenz equation at the same time (Lu et al. 2017). In other words, the input and the output are $s(k) = x_L(k)$ and $y(k) = y_L(k)$, respectively. The Lorenz equation is deterministic, and there exists some relation between these variables, x_L and y_L. ESN learns the relation from the data set $D = \{x_L(k), y_L(k)\}_{k=1}^{K}$, and determines the readout weight $\boldsymbol{w}$ for the inference. We remark that the variable $y_L(k)$ is determined not solely by the variable $x_L(k)$ but the sequence of the variables: $\{\ldots, x_L(k-1), x_L(k)\}$. In other words, generally, RC can approximate the function of the sequence of the input signal as $y(k) = \mathcal{F}\Big(\{\ldots, s(k-1), s(k)\}\Big)$. The top figure in Fig. 1d shows the time series of $x_L(k)$, and the middle shows those of $y_L(k)$ and $\hat{y_L}(k)$. Learning only the readout weight leads to the almost perfect inference. As a reference, the time series of the reservoir variable $r_i (i = 1, \ldots, 5)$ are shown in the bottom figure in Fig. 1d.

---

[1] Note that the activation function ϕ differs from the design matrix Φ.

[2] The Lorenz equation is a simplified model of the thermal convection: $\dot{x} = -\sigma x + \sigma y$, $\dot{y} = rx - y - xz$, $\dot{z} = -bz + xy$, where $\sigma, r, b \in \mathbb{R}$ are the parameters, and, in the main text and later, the subscript L represents the variable of the Lorenz equation. The Lorenz equation has been well studied in the field of nonlinear physics and mathematics as a simple continuous dynamical system exhibiting chaos. In the research field of reservoir computing, the Lorenz equation is used for the time series prediction task, e.g., the input is $s(t) = x_L(t)$ and the output is $y(t) = x_L(t + \tau)$ $(\tau > 0)$, and the inference task of the hidden variable, e.g., the input is $s(t) = x_L(t)$ and the output is $y(t) = z_L(t)$ (Lu et al. 2017; Pathak et al. 2017).

2.3 Geometrical Interpretation of Reservoir Computing

The geometrical interpretation can give insight into the fundamental aspect of RC, in particular shedding light on the difference between RC and the conventional method using the recurrent neural network.

Here, let us consider the readout vector $\hat{\boldsymbol{y}} = \Phi \boldsymbol{w} = \sum_i w_i \boldsymbol{r}_i$ geometrically, where $\boldsymbol{r}_i = (r_i(1), \ldots, r_i(K))^T$. The vector $\hat{\boldsymbol{y}}$ can be regarded as the projection of the vector $\boldsymbol{y} \in \mathbb{R}^K$ to the N-dimensional subspace $V = \mathrm{Span}\{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_N\}$ as follows:

$$\hat{\boldsymbol{y}} = \Phi(\Phi^T \Phi)^{-1} \Phi^T \boldsymbol{y} =: P\boldsymbol{y}, \tag{7}$$

where $P : \mathbb{R}^K \to V$ and we assume $K > N$ since in practice the number of the data set is larger than that of nodes in the network or the dimension of the dynamical system utilized for the reservoir.

The reservoir dynamics, where the common-signal-induced synchronization must occur, is determined by the input sequence except the initial transient period, the state vector $\boldsymbol{r}_i$ is a function of "input vector" $\boldsymbol{s} = (s(1), s(2), \ldots, s(K))^T$, i.e., $\boldsymbol{r}_i = \boldsymbol{r}_i(\boldsymbol{s})$, and the subspace V is also the same $V = V(\boldsymbol{s})$. Considering ESN as a standard model of RC, it can be interpreted that the randomness of the connection weights J_{ij}, v_i varies the response of each node state to the input signal, and enhances the "degree" of linear independence of the vectors $\boldsymbol{r}_i (i = 1, \ldots, N)$. See the illustration of Fig. 1c. As an extreme case, if the connection weights are not random, more precisely J_{ij} and v_i take respective constant values which are independent of i and j, and all nodes in the network become the same state $\boldsymbol{r} \equiv \boldsymbol{r}_i$. In that case, the dimension of the subspace V shrinks to one, and apparently, the approximation by the projection onto V results in failure.

In the conventional method of the recurrent neural network, each connection weight of J_{ij}, v_i is trained as well as the readout weight $\boldsymbol{w}$. This corresponds to generate an appropriate subspace $V'(D)$ spanned by the basis vectors $\boldsymbol{r}_i$ which are determined by the training data D. Since training J_{ij}, v_i requires a high computational cost, it is more difficult to employ a large number of nodes compared with the method of RC. Therefore, in the conventional method of the recurrent neural network, the vector $\boldsymbol{y}$ is approximated within the "well-trained low-dimensional" subspace $V'(D)$. On the other hand, in the method of RC, the vector $\boldsymbol{y}$ is projected onto the "randomly generated high-dimensional" subspace $V(\boldsymbol{s})$.

When a large amount of data is available for the training, it would be natural to assume $K \gg N$. In that case, projection of the vector $\boldsymbol{y} \in \mathbb{R}^K$ to the randomly generated N-dimensional subspace V would be expected to result in a poor approximation. However, even in such a case, RC solves the task well. In order to clarify the reason, it would be necessary to take into account the fact, at least, that (i) the well-solved task by RC has a specific structure, e.g., $y(k)$ does not depend on a long past input $s(k - \tau)$ $(\tau \gg 1)$, and (ii) the response characteristics of the reservoir dynamics, e.g., $r_i(k)$ does not depend on a long past input as well.

3 Characteristics of Dynamical Systems Suitable for Reservoir Computing

Here we introduce two notions (empirical laws), Edge of Chaos and Memory-Nonlinearity Trade-off.

3.1 Edge of Chaos

The conditional Lyapunov exponent is useful not only for conditioning dynamical systems usable for the reservoir, but also for characterizing *good* dynamical systems for the reservoir with respect to information processing. Edge of Chaos is an empirical law for the good dynamical systems in the following sense. The Edge of Chaos law has been supported by many numerical and physical experiments (Bertschinger 2004; Boedecker et al. 2012).

Let us consider two dynamical systems driven by a common input signal. By gradually changing a parameter of the dynamical system, it is often observed that a transition from a synchronized state ($\lambda < 0$) to an asynchronized state ($\lambda > 0$) occurs. For the asynchronized state, a distance between nearby orbits diverges exponentially even if the input signals are the same. The asynchronized state is similar to chaos in deterministic dynamical systems, and hereafter, we refer to it as chaos. For instance, in the ESN model, we observe the transition by increasing the spectral radius [3] of the connection matrix J_{ij}. Edge of Chaos is the empirical law where state RC achieves its highest performance in information processing *slightly before* the transition point, i.e., $\lambda < 0$ and $\lambda \simeq 0$. Many researchers have reported results of numerical and physical experiments supporting Edge of Chaos[4] for various reservoirs and tasks (Bertschinger 2004; Boedecker et al. 2012).

Here we demonstrate the Edge of Chaos law. Figure 2a shows a result of a function approximation task $y(k) = \sin(s(k-1))$ solved by the ESN model. In the upper panel, we show the normalized mean square error (NMSE) in the vertical axis as a standard value for performance evaluation of RC (Appeltant et al. 2011; Inubushi and Yoshimura 2017; Rodan and Tino 2011):

$$\mathrm{NMSE} = \frac{\langle (y(k) - \hat{y}(k))^2 \rangle_T}{\langle (y(k) - \langle y \rangle_T)^2 \rangle_T} \tag{8}$$

[3] For the matrix $J \in \mathbb{R}^{N \times N}$, the definition of the spectral radius is $\rho := \max_i |\mu_i|$ where $\{\mu_i\}_{i=1}^N$ are the eigenvalues of the matrix J. In the absence of the input signal, the origin of the ESN model $\boldsymbol{r} = \boldsymbol{0}$ is the stationary state (fixed point), and the stability of the stationary state is determined by the spectral radius ρ.

[4] The asynchronized state ($\lambda > 0$) is not *deterministic* chaos, and thus, the terminologies of "edge of criticality" or "edge of stability" would be more appropriate.

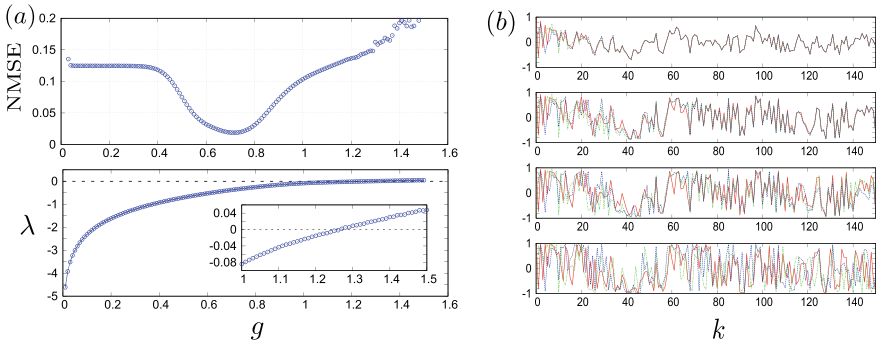


Fig. 2 **a** The upper panel: the normalized mean square error (NMSE) for the function approximation task, which is solved by the ESN model with $N = 100$ nodes. The horizontal axis is the parameter g of the ESN model. The vertical axis is NMSE. The lower panel: the conditional Lyapunov exponent. The horizontal axis is the same as in the upper panel. The transition from CSIS to chaos occurs at $g = 1.25$. **b** The time series of the first component of the state vector, $r_1^\sigma(k)$ $(\sigma = a, b, c)$, from the three different initial conditions $\boldsymbol{r}_1^\sigma(0)$. The time evolutions of ESNs are described by the same Eq. (6) with the common input signals, i.e., the differences are only in the initial conditions. From the top to the bottom, the parameter increases as $g = 1.0, 1.2, 1.3, 1.5$

where the brackets represent the time average $\langle z(t)\rangle_T := 1/T \sum_{t=1}^T z(t)$ for any sequence $\{z(t)\}_{t=1}^T$. The horizontal axis is the parameter g of the ESN model (6). NMSE takes the minimum value at $g \simeq 0.7$. In the lower panel of Fig. 2a, the conditional Lyapunov exponents λ corresponding to the upper panel are shown. The conditional Lyapunov exponent increases with increasing g, and, at $g \simeq 1.25$, we observe the transition from negative to positive, which corresponds to a transition from the synchronized state to the chaotic state. See the inset for the enlarged view around the transition point. The ESN model exhibits the maximum performance for the function approximation task at $g \simeq 0.7$, where the conditional Lyapunov exponent is in the vicinity of zero but a negative value ($\lambda \simeq -0.36$), i.e., slightly before the transition point. Our numerical example also supports the law of Edge of Chaos.

To see the transition from the synchronized state to the chaotic state in detail, we show time series of a variable of the same ESN model with three different initial conditions, i.e., $\boldsymbol{r}^a(0) \neq \boldsymbol{r}^b(0)$, $\boldsymbol{r}^b(0) \neq \boldsymbol{r}^c(0)$, and $\boldsymbol{r}^c(0) \neq \boldsymbol{r}^a(0)$. The numerical settings are the same as the case shown in Fig. 2a. Driven by the common signals, the variables $\boldsymbol{r}^\sigma(k)$ $(\sigma = a, b, c)$ evolve in time according to (6). As an example, the time series of the first component of the state vectors, $r_1^\sigma(k)$ $(\sigma = a, b, c)$, are shown in Fig. 2b. From the top to the bottom, the parameter g is changed as $g = 1.0, 1.2, 1.3, 1.5$.

The conditional Lyapunov exponents at $g = 1.0$ and $g = 1.2$ are negative as shown in Fig. 2a, and correspondingly, the convergence $|r_1^\sigma(k) - r_1^{\sigma'}(k)| \to 0$ $(\sigma \neq \sigma')$, i.e., CSIS, is observed in the upper two panels in Fig. 2b. Note that the convergence occurs at $g = 1.2$ slower than at $g = 1.0$, which is consistent with the absolute values of the conditional Lyapunov exponents. On the other hand, the conditional

Lyapunov exponents at $g = 1.3$ and $g = 1.5$ are positive as shown in Fig. 2a, and correspondingly, CSIS is no longer observed in these two cases in the lower two panels in Fig. 2b.

As demonstrated in Fig. 2b, the closer to the transition point the ESN is, the slower the convergence is, which is trivial by definition of the conditional Lyapunov exponent. It is believed that *the slow convergence leads to a large "memory capacity"*. For an illustration of this statement, let us assume the case that differences in the initial conditions $\boldsymbol{r}^\sigma(0)$ $(\sigma = a, b, c)$ in Fig. 2b are caused by the difference in the input signal $s(0)$. More precisely, these states evolve in time with the common input signal from the remote past, i.e., $\{s(-k)\}_{k=1}^{\infty}$. If the conditional Lyapunov exponent is negative, they should have converged to the same state $\boldsymbol{r}^\sigma(-1) \equiv \boldsymbol{r}(-1)$. And then, if there exist differences in the input signal at $k = 0$ denoted by $s^\sigma(0)$, the differences in the input signal make different states $\boldsymbol{r}^\sigma(0)$ $(\sigma = a, b, c)$ according to (6). Again, ESN receives the common signal $\{s(k)\}_{k=1}^{\infty}$, and the differences in the states dissipate; $\boldsymbol{r}^\sigma(k) \to \boldsymbol{r}(k)$ $(k \to \infty)$ as shown in the top two panels in Fig. 2b.

The memory capacity can be interpreted as an amount of information of the past input signal that can be reconstructed from the present reservoir state. Some working definitions of memory capacity, which measure the accuracy of the reconstruction of the past input signal from the present reservoir state, have been proposed and investigated so far (Jaeger 2002). The reconstruction needs at least the difference in the states. For instance, in the top panel in Fig. 2b, it is impossible to distinguish between the states $\boldsymbol{r}^\sigma(k)$ at $k = 100$, i.e., $\boldsymbol{r}^\sigma(k) \equiv \boldsymbol{r}(k)$. Therefore, at $k = 100$, it is impossible to reconstruct any information of $s^\sigma(0)$ from $\boldsymbol{r}^\sigma(k)$. In this case, it is natural to interpret that the state of ESN at $k = 100$ has no memory of, or forgets, the information in the past input signal $s^\sigma(0)$. As this example shows, the time T required for the convergence of the states dominates the memory in the sense that the ESN retains no memory about the past input signals supplied more than T ago. The time T is roughly estimated by a reciprocal of the conditional Lyapunov exponent $T \propto 1/\lambda$. In this sense, the memory has been referred to as *short term* memory as well.

Approaching the transition point, $\lambda \to -0$, makes the convergence slow, which would lead to a large memory capacity of ESN. That is, the ESN retains a memory of the past input signals of a long time ago. At Edge of Chaos, the memory capacity attains its maximum, which has been shown by using the dynamic mean field theory (Toyoizumi and Abbott 2011). The memory capacity is essential for solving a wide variety of tasks such as the speech recognition tasks and the time series prediction tasks (see the next subsection for an explicit example). Therefore, information processing with RC works well at Edge of Chaos.

Note that in the above argument, we only consider "necessary condition" for the memory. Namely, if ESN can store some memory, i.e., it is possible to reconstruct information about the past input $s^\sigma(0)$, then there exists a difference between the states $\boldsymbol{r}^\sigma(k)$. However, this does not imply the converse, i.e., "if there exists a difference between states, it is possible to reconstruct the past input", and also we cannot

say anything about the *accuracy* of the reconstruction.[5] In this sense, the problem of memory capacity is subtle. Information theoretic quantity such as the mutual information would be more useful than the conditional Lyapunov exponent as discussed in Inubushi and Yoshimura (2017).

3.2 Memory-Nonlinearity Trade-Off

While the memory of the input signal is important, information processing requires in general *nonlinear transformation* of the input signal as well. Here we focus our attention on the two "functions" of the reservoir necessary for information processing, i.e., the short term memory and the nonlinear transformation of the input signal. For the two functions, it is known that there exists a kind of trade-off based on the linearity/nonlinearity of the reservoir dynamics as follows.

Before explaining the trade-off, we introduce an explicit example of a task that requires both memory and nonlinear transformation. The task we consider is a time series prediction. Suppose that we need to predict a future value of a variable $z(t)$, $(z, t \in \mathbb{R})$, and that the variable $z(t)$ is described by an equation $\frac{dz}{dt} = G(z)$, where G is some nonlinear function. Given a time series, $\{\ldots, z(t-h), z(t)\}$ where h is a sampling period, the goal is to predict the future value $z(t+h)$. As one of the numerical schemes of ordinary differential equations, the Adams-Bashforth method has been often used. The explicit (two-step) formula is $z(t+h) = z(t) + h\left(\frac{3}{2}G(z(t)) - \frac{1}{2}G(z(t-h))\right) + O(h^3)$. Considering to predict the value $z(t+h)$ by RC with the Adams-Bashforth method in mind, the reservoir needs to store the memory $z(t-h)$ and perform the nonlinear transformation $G(z(t-h))$. Obviously, these two functions are both essential for the prediction; however, there is a trade-off as described below.

First, we introduce a property of the memory capacity of the input signal. Many researchers have reported so far that the linearity of the reservoir dynamics (the map $\boldsymbol{F}$

[5] In other words, CSIS state $\boldsymbol{r}(k)$ only depends on the recent past input signal, $\{s(k-j)\}_{j=0}^{T}$, and thus, the state vector $\boldsymbol{r}(k)$ is a function of the set of the input signals, i.e., $\boldsymbol{r}(k) = \boldsymbol{r}\left[\{s(k-j)\}_{j=0}^{T}\right]$. Using these notations, the above arguments can be expressed as follows: if the derivatives vanish,

$$\frac{\partial \boldsymbol{r}\left[\{s(k-j)\}_{j=0}^{T}\right]}{\partial s(k-K)} = \boldsymbol{0} \quad (\text{for}^{\forall} k) \tag{9}$$

then, in ESN at the time k, there is no information about $s(k-K)$. However, its inverse, i.e., "if the above derivative does not vanish, then it is possible to reconstruct the information about the past input $s(k-K)$", is not necessarily true. Considering the chaotic regime as an extreme case, $T \to \infty$ and the above derivative is not zero in general due to the sensitive dependence on the initial condition. However, it seems to be difficult to reconstruct the past input from the chaotic state. Hence, we cannot conclude the negation holds.

Table 1 Memory-Nonlinear Trade-off. In the column, the functions of the reservoir, the memory or the nonlinear transformation of the input signal, are shown. In the row, the types of the map $\boldsymbol{F}$ of the reservoir in (1) (the activation function ϕ in the ESN model (6)), linear or nonlinear, are shown

	Short term memory	Nonlinear transformation
Linear	◯	×
Nonlinear	×	◯

in (1)) is preferable for the large memory capacity (Dambre et al. 2012; Ganguli et al. 2018; Toyoizumi 2012). In other words, the nonlinearity of the reservoir dynamics reduces the memory capacity.

Next, we consider the nonlinear transformation of the input signal. Focusing only on the memory capacity, the best strategy seems to be just using the linear reservoir; however, apparently, the linear reservoir cannot perform the nonlinear transformation of the input signal by definition. One of the advantages of RC is to be able to solve linearly non-separable problems via mapping the input signal into a higher dimensional space in a nonlinear way. Thus, the nonlinearity of the reservoir dynamics plays an essential role in general information processing.

Strengthening the nonlinearity of the reservoir increases the ability of the nonlinear transformation of the input signal, but decreases the memory capacity. On the other hand, weakening the nonlinearity of the reservoir increases the memory capacity, but decreases the ability of the nonlinear transformation (Table 1). This relation between these two functions of the reservoir is referred to as *Memory-Nonlinearity Trade-off*, which has been supported numerically (Dambre et al. 2012; Inubushi and Yoshimura 2017; Verstraeten et al. 2010).

Here we give a numerical result illustrating the existence of Memory-Nonlinearity Trade-off. For simplicity, we employ the function approximation task $y(k) = \sin(\pi \nu s(k-\tau))$, where $\nu \in \mathbb{R}$, $\tau \in \mathbb{N}$ are the task parameters. The input signal $s(k)$ is the random variable which is independently and identically drawn from the uniform distribution: $\mathcal{U}(-1, +1)$ at each time t. Let us consider solving this task by the readout from the reservoir state at time k. To this end, two functions are needed: storing the information of the past input signal $s(k-\tau)$ for τ steps (short term memory), and approximating the sin function (nonlinear transformation). The task parameters (ν, τ) control, respectively, the "strength" of the nonlinear transformation and the memory capacity required for solving the task.

To confirm the trade-off summarized in Table 1, we use the ESN model described in (6) and compare the performance of ESNs with linear function $\phi[a] = a$ and the nonlinear function $\phi[a] = \tanh a$. We refer to ESNs with $\phi[a] = a$ and with $\phi[a] = \tanh a$, respectively, as linear ESN and nonlinear ESN. Figure 3a shows the results of the direct comparison in the task parameter space (ν, τ). For a given set of parameters (ν, τ), if the error with the linear ESN is lower than that with the nonlinear ESN, we mark a red square at (ν, τ) in the diagram. If the error with the nonlinear ESN is lower than that with the linear ESN, we mark a blue circle. See Inubushi and Yoshimura (2017) for the details of this diagram.

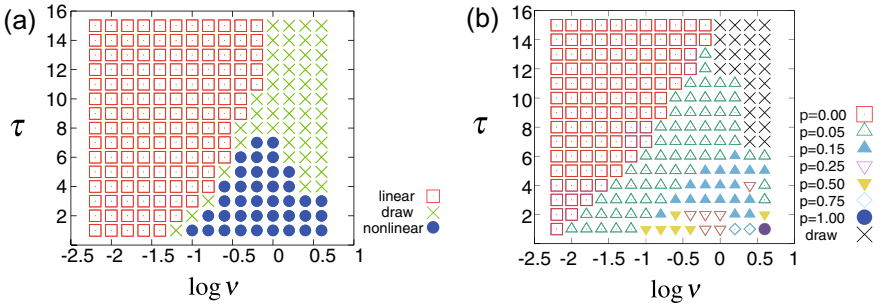


Fig. 3 Performance comparison diagram in task parameter space (ν, τ) (Inubushi and Yoshimura 2017). **a** Comparison between the linear ESN and nonlinear ESN for the demonstration of Memory-Nonlinearity Trade-off. **b** Comparison between the mixture ESNs with various mixture rates p which will be discussed in Sect. 4. Figure reproduced and modified with permission from Springer Nature

The results shown in Fig. 3a are summarized as follows: for the task requiring "large memory capacity and weak nonlinear transformation", e.g., $\log \nu \lesssim -1.0$, $\tau \gtrsim 4$, the linear ESN outperforms the nonlinear one. This observation corresponds to the first row in Table 1. For the task requiring "strong nonlinear transformation and less memory capacity", e.g., $\log \nu \gtrsim -0.5$, $\tau \lesssim 4$, the nonlinear ESN outperforms the linear one. This observation corresponds to the second row in Table 1. While these results are obtained by employing a particular functional form $f(x) = \sin x$ for the task, we confirmed that qualitatively the same results are obtained by employing other function forms $f(x) = \tan x$ and $x(1 - x^2)$. In this sense, the above direct comparison clearly shows the memory-nonlinearity trade-off, which is consistent with previous studies (Dambre et al. 2012; Verstraeten et al. 2010).

What is the mechanism behind the trade-off? In Table 1, the property in the right column is trivial by the definition; however, the property in the left column is nontrivial. Thus, the goal is to uncover the dynamical mechanism behind the degradation of the memory by the nonlinear dynamics of the reservoir. A possible mechanism illustrating the phenomenon has been proposed based on the variational Eq. (5) (Inubushi and Yoshimura 2017).

4 Dynamical Structure Suitable for Reservoir Computing

The simplest method to overcome Memory-Nonlinearity Trade-off would be to use a dynamical system consisting of both linear dynamics and nonlinear dynamics as a reservoir (Inubushi and Yoshimura 2017). Here, we introduce the reservoir where linear dynamics and nonlinear dynamics coexist; hereinafter, we refer to this type of reservoir as *mixture reservoir*. Some numerical results are shown to indicate that the mixture structure is suitable for RC.

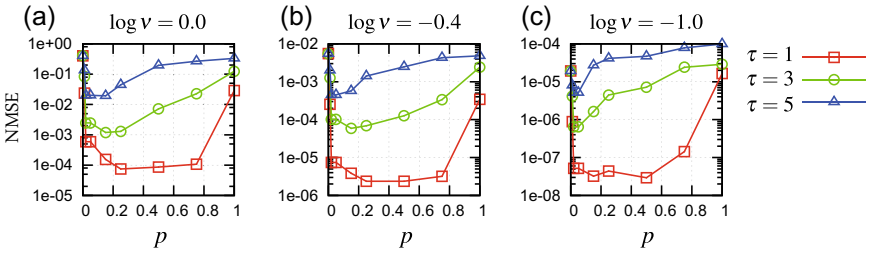


Fig. 4 Performance of the mixture ESN for the function approximation with various task parameters (Inubushi and Yoshimura 2017). The horizontal axis is the mixture rate p, and the vertical axis is the approximation error (NMSE). The task parameters are **a** $\log \nu = 0.0$, **b** $\log \nu = -0.4$, and **c** $\log \nu = -1.0$. The red squares, green circles, and blue triangles correspond to the task parameters $\tau = 1, 3, 5$, respectively. Figure reproduced and modified with permission from Springer Nature

Let us consider the ESN model consisting of N nodes again as an example of the mixture reservoir. In order to overcome the trade-off, it is expected to be effective to use an ESN consisting of "linear node", having the linear activation function $\phi[u] = u$, and "nonlinear node", having the nonlinear activation function $\phi[u] = \tanh u$. More precisely, we connect the linear nodes and the nonlinear nodes with random weights J_{ij}, v_i. As the conventional ESN model, the only readout weight $\{w_i\}$ is adjusted by the training data. We refer to this type of ESN as *mixture* ESN. The number of the nonlinear nodes is denoted by N_{NL}, and we introduce a mixture rate by $p = N_{NL}/N$ which plays a key role in the following discussion.

To study the performance of the mixture ESN, we employ the function approximation task $y(k) = \sin(\pi \nu s(k - \tau))$ as before. Figure 4 shows the numerical results of the function approximation by the mixture ESN with the total number of nodes $N = 100$. The horizontal axis is the mixture rate p, and the vertical axis is the approximation error (NMSE). The mixture ESN at $p = 0$ reduces to the linear ESN, and that at $p = 1$ reduces to the nonlinear, i.e., conventional ESN. The task parameters are (a) $\log \nu = 0.0$, (b) $\log \nu = -0.4$, and (c) $\log \nu = -1.0$. The red squares, green circles, and blue triangles correspond to the task parameters $\tau = 1, 3, 5$, respectively.

In all of the cases shown in Fig. 4, the mixture ESNs outperform the linear ESN ($p = 0$) and the nonlinear ESN ($p = 1$). In particular, for the task with small τ, the approximation error by the mixture ESN is surprisingly reduced in the vicinity of $p = 1$, compared with that by the nonlinear ESN. In other words, **replacing a small number of the nonlinear nodes with linear nodes in the conventional ESN drastically improves the performance**. These drastic improvements in performance are observed in the vicinity of $p = 0$ as well.

In order to study this remarkable improvement of the performance by introducing the mixture structure in more detail, we show, in the right panel of Fig. 5, the performance of a conventional ESN ($p = 1$) with *larger network sizes*. The task is the function approximation task again with $\nu = 1$ and $\tau = 1, 3, 5$. As a reference for the same task, Fig. 4a is shown in the left panel of Fig. 5. The conventional ESN shows better performance by increasing the network size (Rodan and Tino 2011)

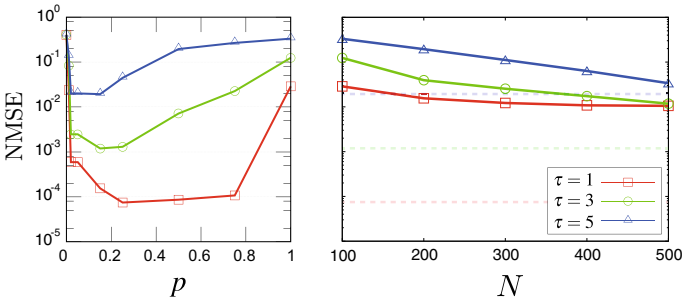


Fig. 5 The performance improvements by introducing the mixture structure versus those by increasing the network size. The left panel is the same as Fig. 4a for the reference. The right panel represents the performance improvements by increasing the network size from $N = 100$ to $N = 500$, where the vertical axis is NMSE. The dotted lines are the values of NMSE by the mixture ESNs at the optimal mixture rates for each task (see the left panel)

in general. Indeed, for each task ($\tau = 1, 3, 5$), the approximation errors are reduced monotonically by increasing the number of nodes ($N = 100, 200, 300, 400, 500$). However, the improvement of the performance by introducing the mixture ESN with fixed network size ($N = 100$) is considerably more effective than that by increasing the network sizes with a fixed mixture rate ($p = 1$). In the right panel of Fig. 5, the error values at the optimal mixture rates are plotted by dotted lines for each task. For instance, regarding the task with $\tau = 5$, while the conventional ESN reduces the error by one-tenth with $N = 500$ nodes, the mixture ESN at the optimal mixture rate can reduce the error at the same level (the blue dotted line) with *only* $N = 100$ nodes. For the tasks with $\tau = 1, 3$, the mixture ESNs at the optimal mixture rate clearly outperform the conventional ESN even with $N = 500$.

It is interesting that the optimal mixture rate $p_{\text{opt.}}$, where the mixture ESN shows the best performance, changes depending on the tasks. In fact, as shown in Fig. 4a, for the tasks with $\tau = 1$, $\tau = 3$, and $\tau = 5$, the optimal mixture rate is $p_{\text{opt.}} \simeq 0.25$, $p_{\text{opt.}} \simeq 0.15$, and $p_{\text{opt.}} \simeq 0.1$, respectively. Note that the larger the parameter τ is, the smaller the optimal mixture rate $p_{\text{opt.}}$ is. This observation is consistent with Memory-Nonlinearity Trade-off. Since the linear nodes play a role to increase the memory capacity, a mixture ESN with a larger number of the linear nodes is effective for a task requiring a larger memory capacity.

To study this dependency, we show a performance comparison diagram in Fig. 3b (Inubushi and Yoshimura 2017). As shown in Fig. 3a, for a set of given task parameters (ν, τ), the optimal mixture rate $p_{\text{opt.}}(\nu, \tau)$ is depicted with different symbols, where the minimal value is numerically found in the set $p \in \{0.00, 0.05, 0.15, 0.25, 0.50, 0.75, 1.00\}$. See Inubushi and Yoshimura (2017) for the details. From this diagram, it is clarified that the optimal mixture rate depends on the task gradually, and, significantly, the mixture ESN ($0 < p < 1$) outperforms the linear and nonlinear reservoir ($p = 0, 1$) over a broad region in the task parameter space.

The above numerical results lead to a conjecture that *the mixture reservoir is one of the dynamical structures suitable for RC in general*. Does the mixture reservoir work well, being less dependent on the tasks or details of the reservoir? For studying this question, we change the topology of the mixture ESN as a detail of the reservoir. We conducted numerical experiments for two other types of the mixture ESNs with the random sparse coupling and the ring coupling. In both cases, nonzero elements of coupling weights J_{ij} ($\neq 0$) are determined by random numbers (Yoshimura et al. 2018). The performance of the mixture ESNs with the random sparse coupling and the ring coupling are shown in Fig. 6 (a) and (b), respectively. The same function approximation tasks are employed as in Fig. 4. Illustrations are shown in the right bottom of each graph in Fig. 6, where the nodes colored with yellow represent the linear nodes. The qualitative features found in the results of the random sparse coupling case and the ring coupling case (Fig. 6a, b) are almost the same as those of the full coupling case (Fig. 4).

Moreover, to confirm the independence of tasks, it has been reported that some standard tasks, the time series prediction for Santa Fe Laser data set and the so-called NARMA task, can be solved by the mixture ESN effectively (Inubushi and Yoshimura 2017). The mixture ESN is effective also for the time series prediction task for the Hénon map as shown in Fig. 7 (Yoshimura et al. 2018). In Fig. 7, the vertical axis shows NMSE for m-step ahead prediction of chaotic signal generated from Hénon map $z(k+1) = 1 - 1.4z(k)^2 + 0.3z(k-1)$, i.e., the input is $z(k)$ and the desired output is $z(k+m)$. The prediction errors with the mixture ESN for the cases of the full coupling, the random sparse coupling, and the ring coupling are shown in Fig. 7 (a), (b), and (c), respectively. In all the cases, the mixture ESNs are effective independently of the network topology. These numerical results so far support the above conjecture.

Introducing the mixture reservoir significantly improves the information processing performance. This improvement occurs independently of tasks and details of the reservoir. This suggests that the mixture rate p is an effective hyperparameter for

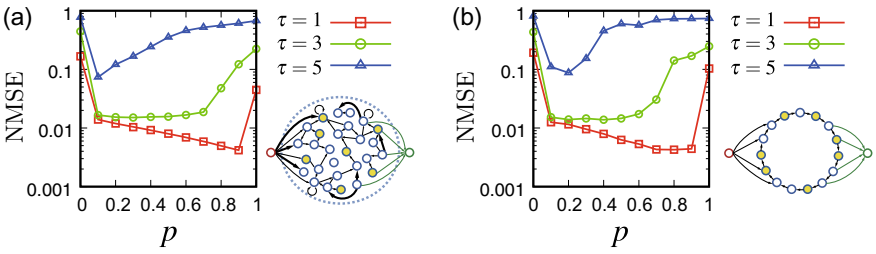


Fig. 6 Performance of the mixture ESN for the function approximation task ($\nu = 1.5$). The horizontal axis is the mixture rate p, and the vertical axis is the approximation error (NMSE). As a topology of the network, two types of the network are studied: **a** the network with the sparse random coupling, **b** the network with the ring coupling. In both cases, nonzero elements of coupling weights J_{ij} ($\neq 0$) are determined by random numbers (Yoshimura et al. 2018). Illustrations are shown in the right bottom of each graph, where the nodes colored with yellow represent the linear nodes

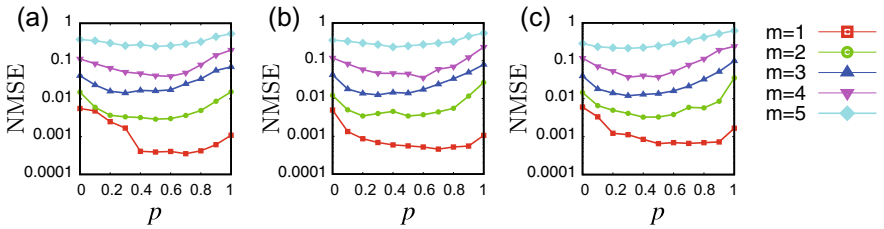


Fig. 7 Performance of the mixture ESN for m-step ahead prediction of the chaotic signal generated from the Hénon map. The horizontal axis is the mixture rate p, and the vertical axis is the prediction error (NMSE). As a topology of the network, three types of the network are studied: the network with **a** the full coupling, **b** the sparse random coupling, and **c** the ring coupling

the optimization of the reservoir. Considering ESN as an example, it is in general difficult to predict the response of the reservoir to changes in the parameters such as g. However, one can expect that decreasing the mixture rate p leads to an increase in the memory capacity of the reservoir. In this sense, the reservoir response to a change in the mixture rate is much simpler than that in the other parameters. This simpleness may make easy the optimization of p.

5 Conclusions and Future Works

In this article, we have given an overview of the mathematical aspects of RC, focusing on the characteristics and structures of the reservoir suitable for information processing. From this viewpoint, we here discuss two open problems of theoretical importance.

(i) Characterization of *good* reservoir:
The reservoirs (physical systems) have many parameters in general, and thus, some optimization over the parameters is required. Although Edge of Chaos is one of the crucial laws useful for optimization, the performance of RC cannot be determined solely by the conditional Lyapunov exponent, which just reduces the dimension of the parameter space by one. Is there a universal quantity, i.e., weakly dependent on tasks, that characterizes a good reservoir completely? In other words, does there exist some general property of dynamical systems which ensures the reservoir computer with high processing performance?

(ii) Dynamical structure suitable for RC:
While the mixture reservoir would be one of the candidates of the suitable structures for RC, the quest for more efficient structures is important. For instance, the configuration of the linear nodes and the nonlinear nodes in the mixture ESN of our study is determined randomly. It is expected that optimizing the configuration improves the performance further. According to the results from on-going numerical experiments, better performance is obtained by a mixture ESN with one-way coupling

from the nonlinear nodes to the linear nodes. The application of the theoretical findings to physical implementation is also important. For instance, adding some linear dynamics to the conventional physical reservoir, "physical mixture reservoir" could outperform the conventional one significantly. Apart from the mixture reservoir, for instance, Deep ESN has been studied theoretically (see Chapter *Deep Reservoir Computing*), and recently the first physical implementation of a deep reservoir, "photonic deep RC", has been reported (Nakajima et al. 2018). It is an interesting future work to find the suitable structures of dynamical systems for RC combining theoretical approaches with experimental approaches.

While we have described mainly the mathematical understandings of RC in this article, finally here we discuss the future development of the whole research field of RC. First, more and more of the various physical systems will be utilized for RC. As illustrated in Sect. 2, if the number of nodes (degree of freedom) of the reservoir is the same as that of the conventional RNN, RC cannot outperform the conventional RNN in principle. Thus, in our opinion, the RC method shows its true ability when one implements it by harnessing physical systems with a huge degree of freedom. Hence, the physical systems with a huge degree of freedom such as nonlinear spatiotemporal systems having high scalability would be promising for the future reservoir. Needless to say, it is expected that the future physical implementation, e.g., using optical systems, realizes the ultrafast information processing with the low energy consumption, which is one of the great advantages of RC.

Recently, many researchers in the field other than information science have studied RC actively as well. Nonlinear physicists in Maryland have proposed the inference method of dynamical variable (Lu et al. 2017), the prediction method of spatiotemporal chaos (Pathak et al. 2018), and the calculation method of Lyapunov exponent from time series (Pathak et al. 2017). Moreover, these methods have been applied to study turbulence physics, where a combination of *transfer learning* approach and RC is proposed for efficient inference of physical quantities (Inubushi and Goto 2019, 2020a, b). It is expected that these findings will be "re-imported" to information science in the future.

While there still remain mysterious points in the RC method, it has great advantages of the suitability for physical implementation and simplicity of the training method. If the theoretical open problems mentioned above are solved, then we can obtain a solid foundation for design principles for the physical reservoir, which may lead to the wide practical use of RC in a future society.

References

L. Appeltant, M.C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C.R. Mirasso, I. Fischer, Information processing using a single dynamical node as complex system. Nat. Commun. **2**, 468 (2011)

N. Bertschinger, T. Natschläger, Real-time computation at the edge of chaos in recurrent neural networks. Neural Comput. **16**, 1413–1436 (2004)

J. Boedecker, O. Obst, J. Lizier, N. Mayer, M. Asada, Information processing in echo state networks at the edge of chaos. Theory Biosci. **131**, 205–213 (2012)
J. Dambre, D. Verstraeten, B. Schrauwen, S. Massar, Information processing capacity of dynamical systems. Sci. Rep. **2** (2012)
F. Flandoli, B. Gess, M. Scheutzow, Synchronization by noise. Probab. Theory Relat. Fields **168**(3), 511–556 (2017)
K. Fujii, K. Nakajima, Harnessing disordered-ensemble quantum dynamics for machine learning. Phys. Rev. Appl. **8**, 024030 (2017)
S. Ganguli, D. Huh, H. Sompolinsky, Memory traces in dynamical systems. Proc. Natl. Acad. Sci. **105**(48), 18970–18975 (2008)
M.W. Hirsch, S. Smale, R.L. Devaney, *Differential Equations, Dynamical Systems, and An Introduction to Chaos* (Academic, 2012)
M. Inubushi, K. Yoshimura, Reservoir computing beyond memory-nonlinearity trade-off. Sci. Rep. **7**(10199), 1–10 (2017)
M. Inubushi, S. Goto, Transferring reservoir computing: Formulation and application to fluid physics, in *Lecture Notes in Computer Science*, vol. 11731 (Springer, 2019)
M. Inubushi, S. Goto, Transfer learning for nonlinear dynamics and its application to fluid turbulence. Phys. Rev. E **102**, 043301 (2020)
M. Inubushi, S. Goto, Inference of the energy dissipation rate of turbulence by machine learning (2020b). In preparation
H. Jaeger, Short term memory in echo state networks, vol. 152. GMD-Report (2002)
H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)
L. Larger, A. Baylón-Fuentes, R. Martinenghi, V.S. Udaltsov, Y.K. Chembo, M. Jacquot, High-speed photonic reservoir computing using a time-delay-based architecture: million words per second classification. Phys. Rev. X **7**, 011015 (2017)
Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, E. Ott, Reservoir observers: model-free inference of unmeasured variables in chaotic systems. Chaos **27**(4) (2017)
W. Maass, H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations. Theor. Comput. Sci. **2560**, 2531–2560 (2002)
K. Nakajima, H. Hauser, T. Li, R. Pfeifer, Information processing via physical soft body. Sci. Rep. **5**, 10487 (2015)
M. Nakajima, M. Inubushi, T. Goh, T. Hashimoto, Coherently driven ultrafast complex-valued photonic reservoir computing, in *Conference on Lasers and Electro-Optics*. OSA Technical Digest (online) (Optical Society of America, 2018), p. SM1C.4
M. Nakajima, S. Konisho, K. Tanaka, T. Hashimoto, Deep reservoir computing using delay-based optical nonlinear oscillator, in *Conference in Cognitive Computing 2018* (2018)
R. Nakane, G. Tanaka, A. Hirose, Reservoir computing with spin waves excited in a garnet film. IEEE Access **6**, 4462–4469 (2018)
J. Pathak, Z. Lu, B.R. Hunt, M. Girvan, E. Ott, Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. Chaos **27**(12) (2017)
J. Pathak, B. Hunt, M. Girvan, L. Zhixin, E. Ott, Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach. Phys. Rev. Lett. **120**(2), 24102 (2018)
L.M. Pecora, T.L. Carroll, Driving systems with chaotic signals. Phys. Rev. A **44**(4), 2374–2383 (1991)
A. Rodan, P. Tino, Minimum complexity echo state network. Trans. Neural Netw. **22**(1), 131–144 (2011)
K. Takano, C. Sugano, M. Inubushi, K. Yoshimura, S. Sunada, K. Kanno, A. Uchida, Compact reservoir computing with a photonic integrated circuit. Opt. Express **26**(22), 29424–29439 (2018)
J.N. Teramae, D. Tanaka, Robustness of the noise-induced phase synchronization in a general class of limit cycle oscillators. Phys. Rev. Lett. **93**(20), 1–4 (2004)
T. Toyoizumi, Nearly extensive sequential memory lifetime achieved by coupled nonlinear neurons. Neural Comput. **2699**, 2678–2699 (2012)

T. Toyoizumi, L.F. Abbott, Beyond the edge of chaos: amplification and temporal integration by recurrent networks in the chaotic regime. Phys. Rev. E Stat. Nonlinear Soft Matter Phys. **84**(5), 1–8 (2011)

A. Uchida, R. McAllister, R. Roy, Consistency of nonlinear system response to complex drive signals. Phys. Rev. Lett. **93**, 244102 (2004)

D. Verstraeten, J. Dambre, X. Dutoit, B. Schrauwen, Memory versus non-linearity in reservoirs, in *Proceedings of the International Joint Conference on Neural Networks* (2010)

K. Yoshimura, M. Inubushi, Y. Ikeda, Y. Nagasawa, T. Kogahara, Computation performance of mixture-unit echo state networks, in *Proceedings of 2018 International Symposium on Nonlinear Theory and Its Applications* (2018), pp. 404–407

K. Yoshimura, I. Valiusaityte, P. Davis, Synchronization induced by common colored noise in limit cycle and chaotic systems. Phys. Rev. E Stat. Nonlinear Soft Matter Phys. **75**, 026208 (2007)

K. Yoshimura, J. Muramatsu, P. Davis, Conditions for common-noise-induced synchronization in time-delay systems. Phys. D **237**, 3146–3152 (2008a)

K. Yoshimura, P. Davis, A. Uchida, Invariance of frequency difference in nonresonant entrainment of detuned. Prog. Theor. Phys. **120**(4), 1–13 (2008b)

Reservoir Computing for Forecasting Large Spatiotemporal Dynamical Systems

Jaideep Pathak and Edward Ott

Abstract Forecasting of spatiotemporal chaotic dynamical systems is an important problem in several scientific fields. Crucial scientific applications such as weather forecasting and climate modeling depend on the ability to effectively model spatiotemporal chaotic geophysical systems such as the atmosphere and oceans. Recent advances in the field of machine learning have the potential to be an important tool for modeling such systems. In this chapter, we review several key ideas and discuss some reservoir-computing-based architectures for purely data-driven as well as hybrid data-assisted forecasting of chaotic systems with an emphasis on scalability to large, high-dimensional systems.

1 Motivation

This chapter is motivated by problems in the forecasting of large, complex, spatiotemporally chaotic systems and by the possibility that machine learning might be a useful tool for the significant improvement of such forecasts. Examples of the type of potential tasks we have in mind are forecasting ocean conditions; forecasting conditions in the solar wind, Earth's magnetosphere, and ionosphere (so-called 'space weather', important for Earth-orbiting spacecraft, GPS accuracy, power-grid disruptions, etc.); forecasting the spatial distribution of plant growth in response to environmental changes; forecasting the development of forest fires and their response to fire fighting strategies; and weather forecasting.

Focusing on weather forecasting as perhaps the most important such example, we note the following two relevant points: (i) weather forecasts impact the lives of many millions of people, e.g., by providing warnings of destructive events, like hurricanes or snowstorms; (ii) currently used weather forecasting employs physics-based models (the equations of fluid dynamics, radiative heat transfer, etc.), plus

J. Pathak (✉) · E. Ott
University of Maryland, College Park, MD, USA
e-mail: jpathak@umd.edu

E. Ott
e-mail: edott@umd.edu

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_6

geographical knowledge of mountains, oceans, etc. The models in (ii), however, have substantial errors, which, for example, may arise due to imperfect modeling of crucial subgrid-scale dynamics (like clouds, turbulent atmospheric motions, and interactions with small-scale geographic features).

Can machine learning from data potentially correct such knowledge deficits and thus contribute to significant improvement of forecasts? For other recent work which addresses the issue of using machine learning for analyzing and forecasting spatiotemporal dynamical systems, see Brunton et al. (2016), Lusch et al. (2018), Raissi et al. (2019), Vlachas et al. (2018), and Wan et al. (2018). In this paper, focusing on reservoir computing, we discuss and summarize some recent research that may be relevant to this question.

2 Background: Prediction of 'Small' Chaotic Systems

Machine Learning (ML) prediction of the ergodic chaotic evolution of a dynamical system was considered by Jaeger and Haas (2004) in the context of reservoir computing (Jaeger 2001). The basic idea of their scheme is illustrated (in its discrete time version) in Fig. 1. Given time-series training data $\mathbf{u}(n)$, obtained from sampling measurements of some unknown chaotic dynamical system on an ergodic attractor (with a sampling time interval Δ), for $n = -T, -T+1, -T+2, \ldots, -1$, the ML system is trained to output $\mathbf{u}(n+1)$, when $\mathbf{u}(n)$ is the input (Fig. 1). If the vector state of the unknown dynamical system is denoted by $\mathbf{x}$, we can represent the measurements by a 'measurement function' $\mathbf{H}$ such that the measurement vector $\mathbf{u}$ is given by

$$\mathbf{u}(n) = \mathbf{H}(\mathbf{x}(t)), \qquad t = n\Delta. \tag{1}$$

Since the dimension of $\mathbf{u}$ may be less than twice the dimension of the attractor of the dynamical system in $\mathbf{x}$-space, Eq. 1 may not be an embedding (Sauer et al. 1991), and, to compensate for this, it is important that the ML system has memory. That is, the current state of the ML device depends on its current input and on the history of the inputs. (Memory can also be realized or supplemented by incorporating time-delayed measurements as additional components of $\mathbf{u}(t)$; however, for simplicity we will not further consider this possibility here.)

As an example, assume that the goal is to predict the future value of the measurements. Following Jaeger and Haas (2004), when, after the training phase (Fig. 1a), the time to predict comes, the input from the measured state is no longer available and, as shown in Fig. 1b, is replaced by the ML system output; i.e., the output is fed back into the input. Thus, at the beginning step of the forecasting phase, $\mathbf{u}(0)$ is made the input, which, if all goes well, then produces a new (forecasted) output $\mathbf{u}(1)$, which, when fed back, outputs a forecast for $\mathbf{u}(2)$, which when fed back outputs a forecast for $\mathbf{u}(3)$, and so on. Of course, there is always some small error in the output (e.g., if the input is $\mathbf{u}(0)$, the output is only approximately $\mathbf{u}(1)$), and due to the

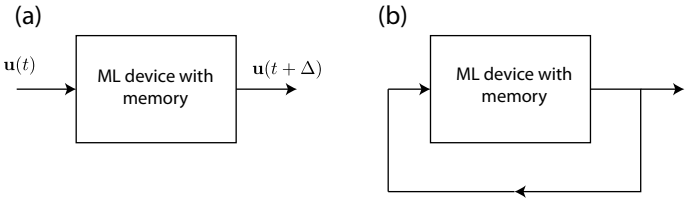


Fig. 1 Schematic illustration of the **a** training phase, and **b** prediction phase for a simple ML forecasting system

assumed chaos of the dynamical system generating the measurements, these errors build up as the feedback loop is successively traversed. Thus, as is typical for chaotic processes, the prediction accuracy eventually breaks down. So good prediction can only be expected for several Lyapunov times.

Note that the closed-loop system shown in Fig. 1a may itself be regarded as an autonomous dynamical system. Thus in Lu et al. (2018), Lu et al. have employed dynamical systems theory concepts (especially the concepts of 'generalized synchronization' Afraimovich et al. 1986; Kocarev and Parlitz 1996; Pecora et al. 1999; Rulkov et al. 1995 and Lyapunov exponents Abarbanel 2012; Kantz and Schreiber 2004; Ott 2002; Ott et al. 1994) to analyze conditions on the ML system that make for good reproduction of the dynamics of the unknown system that produces the data.

With regard to the time step Δ, one might have the following question. Assuming that one is interested in forecasting a chaotic process forward by an amount of time T, why not simply set $\Delta = T$ and do one prediction step (thus eliminating the need for the closed-loop configuration in Fig. 1b)? The answer is that for typical cases, one is often interested in prediction times T that may be as large as several Lyapunov times (a Lyapunov time is a typical time it takes a small orbit perturbation to grow by a factor of e). In such cases, with $\Delta = T$, small changes in $\mathbf{u}(n)$ can lead to relatively large changes in $\mathbf{u}(n+1)$. Thus, the functional relationship that the ML system is trained to learn is relatively complex ('wiggly') making its task relatively hard. Accordingly, it has been found that using smaller Δ with the feedback loop as shown in Fig. 1b is advantageous.

One aspect of the forecasting scheme illustrated in Fig. 1 is that the various versions of ML can in principle be employed. Since memory is typically required, and is, in any case, expected to be advantageous for prediction tasks, the two most natural candidates for consideration are reservoir computing (as in the paper of Jaeger and Haas 2004) and Long Short-Term Memory (Hochreiter and Schmidhuber 1997) (as in the paper of Vlachas et al. 2018). In this chapter, we consider reservoir computing due to its appreciably shorter training times and its potential for advantageous physical implementations discussed in other chapters of this book.

3 Machine Learning and the Forecasting of Large, Complex, Spatiotemporally Chaotic Systems

The scheme (Jaeger and Haas 2004) described in Fig. 1 and Sect. 2 works well for small to moderate size systems. However, we find that straightforward scaling of, e.g., a reservoir computing implementation of Fig. 1 to very large size results in requirements that appear to be unfeasible or, at least, very demanding, in practical terms (e.g., with respect to the reservoir size required, amount of training data, and computations for training). Thus we seek ways of mitigating this problem. Specifically, we wish to apply prior physical knowledge of the system to be forecasted and integrate this prior knowledge with a machine learning approach via a suitable prediction system architecture. In particular, we consider two types of prior knowledge as described below.

First, we note that information in spatially extended physical systems generally propagates at a finite speed. Thus, a perturbation applied at some point in space will not immediately affect the state at some distant point. We refer to this as 'the locality of short-term causal interactions' (LSTCI). Assuming, for illustrative purposes, that space is one dimensional, if we want to predict the state at time $t + \Delta$ in the region $x_0 - l_0 < x < x_0 + l_0$ from the state at time t, we only need to consider the state at time t within the region, $x_0 - (l_0 + d) < x < x_0 + (l_0 + d)$, where d is large enough such that information affecting the prediction of the state in the region, $x_0 - l_0 < x < x_0 + l_0$, does not propagate fast enough to move a distance d over the one-step prediction time Δ. Thus, a parallel approach (Sect. 4) can be employed in which multiple ML systems predict $\mathbf{u}$ in corresponding limited overlapping spatial regions where the lengths of the overlap between regions are at least d. This will be discussed in Sect. 4.3 (note that this consideration provides an added motivation for considering Δ to be small).

The second type of physical knowledge comes from knowledge-based modeling, typically in the form of inaccurate partial differential equations like those discussed in Sect. 2 in the context of weather forecasting. In Sect. 5, we discuss a hybrid technique that utilizes both an imperfect knowledge-based model and a relatively small Reservoir Computing ML forecaster (see Wan et al. 2018 for a similar implementation using Long Short-Term Memory ML). In the training of the hybrid system, the state variables of *both* the ML system and the knowledge-based system are combined via a set of adjustable 'weights' in such a way as to very closely fit the desired prediction system outputs as determined by the training data. Thus, we view the training as being designed to take the best aspects of the predictions of the ML component and the knowledge-based component and to combine these good aspects in a semi-optimal fashion. Indeed, as we later show (Sect. 5), even in a case where the knowledge-based system error and the relatively small size of the ML component were such that each acting alone gave relatively worthless forecasts, that, when incorporated into our hybrid scheme, excellent forecasts can result. Furthermore, as we will document elsewhere, the machine learning component typically requires less

training data for use in the hybrid scheme than would be the case for a much larger, pure machine learning system.

In Sect. 6, we discuss and illustrate a prediction system architecture for combining the parallel and hybrid schemes so as to create a methodology that is potentially scalable to very large complex systems. In this combined scheme, we envision the knowledge-based component of this combined system to be global and not based on the LSTCI assumption (e.g., like models currently used for weather prediction).

4 Distributed Parallel Prediction

In this section, we describe how to efficiently train a Reservoir Computer to predict time-series from high-dimensional spatiotemporal chaotic systems. This scheme was introduced in Pathak et al. (2018a). As mentioned in Sect. 3, we will exploit the locality of short-term causal interactions (LSTCI), present in many spatiotemporal systems of interest, to divide the computational task over many independent computing units or 'cores'. The key idea behind this division is our assumption that the near-term future of the state of a particular spatial region of the spatiotemporal system is only affected by dynamics occurring nearby (in a spatial sense), and the dynamics occurring far away from it has no effect. This assumption presupposes the absence of short-term long-range interactions in the spatiotemporal system.

For simplicity, in most of what follows, we consider a spatiotemporal dynamical system defined by some set of equations evolving a scalar state variable $y(x, t)$ forward in time (t) in a one-dimensional spatial domain (x) with periodic boundary conditions. Thus, $x \in [0, L)$ and $y(x + L, t) = y(x, t)$. We assume that the measurement vector $\mathbf{u}(n)$, is $K-$dimensional with each scalar element of $\mathbf{u}(n)$ being the state variable $y(x, t)$, measured at regular intervals of time Δ and over a uniformly spaced spatial grid with K grid points: $x = (L/K), (2L/K), (3L/K), \ldots, L$.

4.1 Partitioning the Spatial Grid

We thus have a set of K time series $u_k(n)$, $1 \leq k \leq K$ on K grid points where $t = n\Delta$ and n is an integer. The measurement vector $\mathbf{u}(n)$ is thus a $K-$vector whose kth scalar element is $u_k(n)$. As shown in Fig. 2, we then use the $K-$dimensional measurement vector $\mathbf{u}(n)$ to form a set of M vectors $\{\mathbf{v}_i(n)\}_{i=1,2,\ldots,M}$ where each such vector i has dimension $(K/M) + 2l$, and its entries consist of the y values at time n in *overlapping* regions where each region i has K/M central nodes supplemented by overlap buffer regions to its left and right of length l nodes each (e.g., in the schematic illustration shown in Fig. 2, $l = 2$ and $(K/M) = 4$). The choice of l is made using our LSTCI assumption and requiring that the time Δ prediction of the K/M central nodes of vector $\mathbf{v}_i(n)$ is (to a very good approximation) not influenced by nodal states not included in $\mathbf{v}_i(n)$.

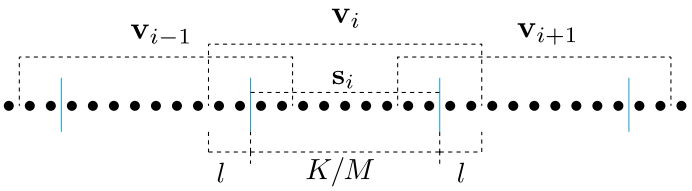


Fig. 2 Partitioning scheme

The K grid points are also partitioned into M *non-overlapping* groups with K/M-dimensional state vectors, $\mathbf{s}_i$ for $i = 1, 2, \ldots, K/M$, as illustrated in Fig. 2 (from Fig. 2, we see that $\mathbf{v}_i$ becomes $\mathbf{s}_i$ when l is set to zero). The task of obtaining the time Δ prediction of each such group is assigned to M separate ML systems which will be trained to learn the local group dynamics and predict the future state of the time series at those grid points. Making use of our LSTCI assumption, we suppose that Δ is small enough that $\mathbf{s}_i(n+1)$ depends on $\mathbf{s}_i(n+1)$, but is independent of $\mathbf{v}_{i\pm k}(n)$ for $k \geq 1$.

Note that l and M are 'hyperparameters' of our model and can be tuned while optimizing with respect to factors such as computational cost and prediction results. We use the term hyperparameters to denote a small set of parameters that are not adjusted via the training procedure and that characterize gross overall features of the ML device (e.g., the hyperparameters we deal with are the amount of nonlinearity and memory, the reservoir size, input coupling strength, and training regularization). Hyperparameters are often set by the user on an empirical, trial-and-error basis so as to achieve 'good' results on a test data set (typically, the test data set is separate from the training data set so as to ensure generalization of training). Hyperparameters can also be set more systematically via optimization techniques.

The above description (e.g., Fig. 2) is for the case of a spatially one-dimensional system. The generalization to higher dimensions is straightforward and is indicated for the two-dimensional case in Fig. 3, in which the solid black lines divided space into square patches labeled by the two subscripts (i, j), the vector $\mathbf{s}_{i,j}$ (analogous to $\mathbf{s}_i$ in the one-dimensional case) specifies the system state within patch (i, j), and the vector $\mathbf{v}_{i,j}$ (analogous to $\mathbf{v}_i$ in the one-dimensional case) specifies the system state in the expanded overlapping regions indicated by the dashed square in Fig. 3. (A similar scheme was also used in Zimmermann and Parlitz 2018 to infer unmeasured state variables.)

4.2 Training

Now specializing in the case of Reservoir Computing, using a simple procedure outlined in Jaeger and Haas (2004), we generate M reservoir systems each based on a $D \gg K/M$ node recurrent artificial neural network characterized by a weighted

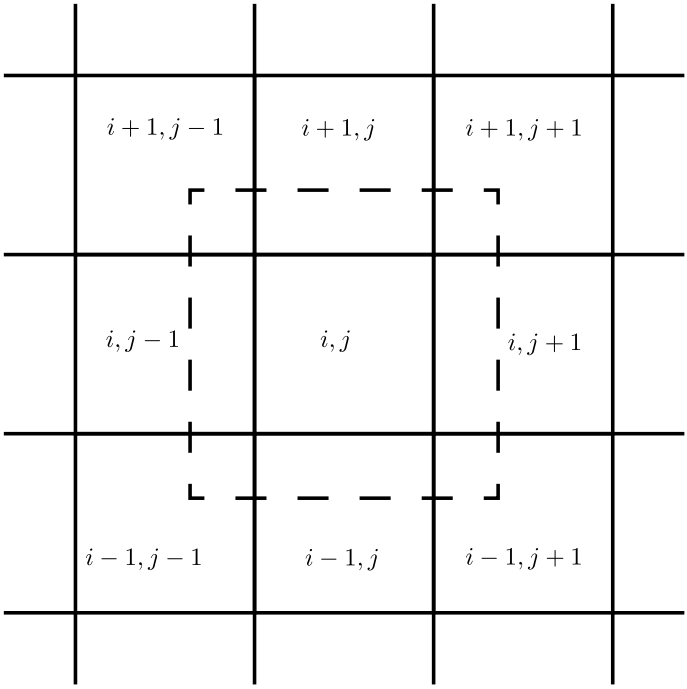


Fig. 3 Spatial regions $\mathbf{s}_{i,j}$ and $\mathbf{v}_{i,j}$ for two-dimensional ML parallel prediction. See also Zimmermann and Parlitz (2018)

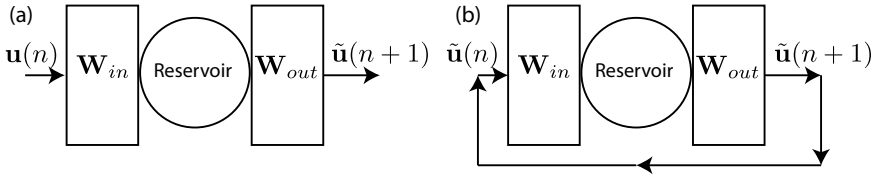


Fig. 4 **a** Open-loop 'training' configuration; **b** Closed-loop 'prediction' configuration for the reservoir computing prediction scheme

adjacency matrix $\mathbf{A}_i$. Assuming that initial start-up transient activity is omitted, during the training phase, which we take to run from $n = -T$ to $n = -1$, each of the M reservoir networks (Fig. 5) evolves according to the following equation:

$$\mathbf{r}_i(n+1) = \tanh\left[\mathbf{A}_i \mathbf{r}_i(n) + \mathbf{W}_{\text{in},i}\mathbf{v}_i(n)\right], \tag{2}$$

where $\mathbf{r}_i$ is the $D-$vector whose elements are the states (which are here taken to be scalars) of each of the network nodes, and the $D \times [(K/M) + 2l]$ matrix $\mathbf{W}_{\text{in},i}$ couples the ith input training vector $\mathbf{v}_i$ to nodes of the recurrent reservoir network. The reservoir states $\mathbf{r}_i(n)$, $-T \leq n \leq -1$ are stored in a matrix $\mathbf{R}_i$, such that the T

columns of $\mathbf{R}_i$ are the vectors $\mathbf{r}_i(n)$. The state vector $\mathbf{r}_i$ is then used to produce an output vector whose dimension is K/M. In the simplest case (not necessarily the only choice), this is done via a $(K/M) \times D$ output coupling matrix $\mathbf{W}_{\text{out},i}$, such that the output is $\mathbf{W}_{\text{out},i}\mathbf{r}_i$. We regard the parameters formed by the elements of $\mathbf{W}_{\text{in},i}$ and $\mathbf{A}_i$ as fixed and use only the parameters of the output coupling function, i.e., the matrix elements of $\mathbf{W}_{\text{out},i}$ for training. That is, we adjust $\mathbf{W}_{\text{out},i}$ so that a desired training output results. In accord with the left panel of Fig. 1, we desire the output to approximate $\mathbf{s}_i(n+1)$ when the training input is $\mathbf{v}_i(n)$,

$$\mathbf{W}_{\text{out},i}\mathbf{r}_i(n+1) \simeq \mathbf{s}_i(n+1), \tag{3}$$

for $-T \leq n \leq -1$. Calculating a matrix $\mathbf{W}_{\text{out},i}$ that satisfies Eq. (3) is referred to as 'training' the neural network. This is illustrated in Fig. 5. In Eq. 3, the training problem is incompletely defined as we have not specified what exactly we mean by the '$\simeq$' sign. The simplest possible choice is to require that the right-hand side and left-hand side of Eq. (3) be approximately equal in the sense of their ℓ_2 norms. Thus, one might choose the matrix $\mathbf{W}_{\text{out},i}$ that minimizes the sum of squared deviations of the output from its desired target value,

$$\epsilon = \sum_{n=-T}^{-1} \|\mathbf{W}_{\text{out},i}\mathbf{r}_i(n+1) - \mathbf{s}_i(n+1)\|^2. \tag{4}$$

However, this can often be problematic, and to avoid overfitting to the training data and better promote the generalizability of the training to cases beyond the training data, a regularization procedure is typically employed. To this end, $\mathbf{W}_{\text{out},i}$ is often

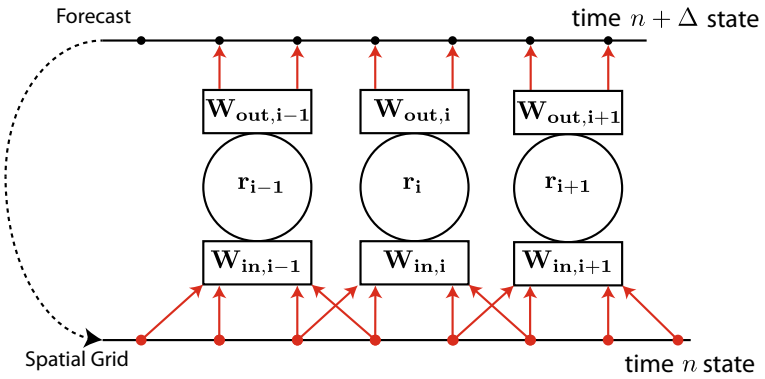


Fig. 5 Parallelized prediction scheme. $(K/M) = 2$, $l = 1$. The open-loop configuration corresponds to this figure with the dashed line ignored. The closed-loop configuration is represented by the dashed line which indicates that the required inputs to the reservoirs are taken from the corresponding outputs

required to minimize $\epsilon + \epsilon_r$, where ϵ_r is a term that penalizes excessively large values of the training parameters,

$$\epsilon_r = \beta \sum_j \|\mathbf{W}_{\text{out},i}\|^2_{:j}. \tag{5}$$

In Eq. (5), $\|\cdot\|_{:j}$ denotes the ℓ_2 norm of the jth column of a matrix and β is a hyperparameter called the regularization constant that determines the strength of the regularization term. Note that this minimization is a standard linear regression problem with a well-known matrix-based solution (e.g., see Eq. (6) below). Alternatively, if the matrix inverse is computationally onerous (as might be the case for very large D), one can minimize $(\epsilon + \epsilon_r)$ by the steepest descent. Another device, used in what follows to increase the expressive power of the reservoir computing network without sacrificing the training simplicity afforded by linear regression is to construct a vector $\mathbf{r}^*$ from $\mathbf{r}$ such that the elements r_j of $\mathbf{r}$ and r^*_j of $\mathbf{r}^*$ are related by the following rule: $r^*_j = r^2_j$ if j is even and $r^*_j = r_j$ if j is odd. Putting all of this together, we obtain the expression for $\mathbf{W}_{\text{out},i}$,

$$\mathbf{W}_{\text{out},i} = \left(\mathbf{R}^*_i \mathbf{R}^{*T}_i + \beta \mathbf{I}\right)^{-1} \mathbf{R}^*_i \mathbf{S}^T_i \tag{6}$$

where $\mathbf{R}^*_i$ is the $D \times T$ matrix with columns given by the vectors $\mathbf{r}^*_i(n)$, and $\mathbf{S}_i$ is the $(N/M) \times T$ matrix whose columns are the training data time-series vectors $\mathbf{s}_i(n)$. Note that each of the individual reservoir systems i is trained independently, and thus training is parallelized. Furthermore, the input and output dimensions can be much smaller than the size of the global measurement state. Thus, the individual parallel reservoirs can be much smaller than would be the case without making use of LSTCI. Having determined $\mathbf{W}_{\text{out},i}$, we rewrite Eq. (3) as

$$\mathbf{W}_{\text{out},i} \mathbf{r}^*_i(n+1) = \tilde{\mathbf{s}}_i(n+1) \tag{7}$$

where $\tilde{\mathbf{s}}_i(n)$ denotes the machine learning approximation to the true group i state vector $\mathbf{s}_i(n)$. Similarly, for later reference, we will also use $\tilde{\mathbf{v}}_i$ to denote the corresponding approximation to $\mathbf{v}_i$.

For later reference, in all of the numerical experiments reported in what follows, adjacency matrices are random Erdős–Renyi matrices of fixed average degree that are scaled by multiplication by a constant so as to fix the matrix spectral radius (magnitude of its largest eigenvalue) at a pre-selected value denoted by ρ, and the elements of the input coupling matrices are each randomly selected numbers with uniform probability in $[-\sigma, \sigma]$. Both ρ and σ are hyperparameters. For the tasks we study, we find that the reservoir computer performance is largely insensitive to the reservoir network topology. For specificity in the example given in this paper, we use random Erdős–Renyi networks with an average degree equal to 3.

4.3 Prediction

At the end of the minimization procedure described above, we obtain the set of matrices $\mathbf{W}_{\text{out},i}$ that maps the internal state of the reservoir $\mathbf{r}_i(n)$ at a given instant of time to a good approximation of the state of the vector $\mathbf{u}(n)$. Since $\mathbf{r}_i(n)$ is dependent on past inputs $(\mathbf{v}_i(n-k), k \geq 1)$, we hope to have trained the reservoir to perform single step forecasts of step size Δ. We know this to be true on the training data. Whether the training generalizes to 'out-of-sample' data, i.e., the time series $\mathbf{u}(n)$ outside the interval $-T \leq n \leq -1$ is a separate question. This question will be addressed empirically by numerical experiments in sections to follow. In the prediction stage, we re-configure our parallel ML system to generate forecasts for $n > 0$ (see Fig. 5 as follows):

$$\begin{aligned} &\text{step 1:} && \tilde{\mathbf{s}}_i(n) = \mathbf{W}_{\text{out},i}\mathbf{r}_i^*(n) \\ &\text{step 2:} && \text{Construct } \tilde{\mathbf{v}}_i(n) \text{ from } \tilde{\mathbf{s}}_i(n), \tilde{\mathbf{s}}_{i\pm1}(n) \\ &\text{step 3:} && \mathbf{r}_i(n+1) = \tanh\left[\mathbf{A}_i\mathbf{r}_i(n) + \mathbf{W}_{\text{in},i}\tilde{\mathbf{v}}_i(n)\right] \end{aligned} \tag{8}$$

That is, the one-step-ahead output predictions of $\mathbf{s}_j(n+\Delta)$ for $j = i-1, i, i+1$, are used to construct a prediction for $\mathbf{v}_i(n+\Delta)$, which is then fed back to the input of the reservoir system i, producing a new output prediction of $\mathbf{s}_i(n+2\Delta)$, and this process is cyclically repeated. To summarize, the prediction algorithm in Eq. 8 has three key steps. In the first step, we compute the output of each reservoir network $\tilde{\mathbf{s}}_i(n)$ to get a Δ-step prediction. Next, we construct the overlapping partitions $\tilde{\mathbf{v}}_i(n)$ from $\tilde{\mathbf{s}}_i(n)$, $\tilde{\mathbf{s}}_{i\pm1}(n)$. The vector $\tilde{\mathbf{v}}_i$ is the feedback from the output of the reservoir network to the input. Equation 8 forms an autonomous dynamical system that predicts the future states of the dynamical system that it was trained on.

4.4 Re-synchronization

The prediction scheme described in Sect. 4.3 will be expected to generate accurate predictions for a finite amount of time determined by the fundamental properties of the chaotic dynamical system being predicted, especially the average error e-folding time (the inverse of the largest Lyapunov exponent of the chaotic process of interest Ott 2002). Because of the chaos, the predictions will necessarily diverge from the ground truth after some amount of time. After the predicted trajectory of the dynamical system has diverged far enough from the true trajectory, the predictions made by the reservoir system are no longer accurate enough to be useful. If, after such a divergence, one wishes to restart prediction from some later time, the reservoir system does not have to be re-trained in order to generate accurate new predictions. Rather, we find that it is sufficient to re-synchronize the reservoir network states $\mathbf{r}_i(n)$ with the ground truth by simply running the reservoir networks without feedback according to Eq. (2) for ξ time steps prior to the desired beginning of prediction.

Importantly, we find that the necessary re-synchronization time is very much smaller than the necessary training time $\xi \ll T$. Thus, we emphasize that the output weights $\mathbf{W}_{\text{out},i}$ do not need to be re-computed for subsequent predictions at later times.

4.5 An Example: A Lorenz 96 Model

We consider one of the classes of 'toy' models of atmospheric dynamics proposed in a 1996 paper of Lorenz (1996) and use it as a testbed for our parallelized prediction setup. The particular model we use is defined as a set of interacting scalar variables $X_j(t), 1 \leq j \leq N$ on a spatially periodic grid ($X_{j+N}(t) = X_j(t)$) with the dynamics given by the coupled ordinary differential equations,

$$\frac{dX_j}{dt} = -X_j + X_{j-1}X_{j+1} - X_{j-1}X_{j-2} + F. \tag{9}$$

We numerically integrate Eq. (9) using a standard fourth-order Runge–Kutta scheme and generate simulated time-series data. We sample the simulated data at the time step intervals $\Delta = 0.01$ to generate the training data vectors $\mathbf{u}(n)$ in the interval $-T \leq n \leq -1$, where $T = 80{,}000$. We also generate a test data set $\mathbf{y}(n)$ in the interval $0 \leq n \leq 20{,}000$. The training data set is used to train the reservoir computing system according to the scheme outlined in Sect. 4 while the test data set is used to validate the accuracy of the predictions. The validation scheme is as follows:

- Step 1: We generate a random set of 50 'initial time points' n_k, $1 \leq k \leq 50$, such that $1 \leq n_k \leq 20{,}000 - \tau - \xi$.
- Step 2: The test data $\mathbf{y}(n)$ is used to synchronize the reservoirs to the true trajectory for ξ time steps between n_k and $n_k + \xi$ according to the synchronization procedure outlined in Sect. 4.4.
- Step 3: The reservoir network is run in prediction mode according to Eq. (8) for τ time steps. We evaluate the prediction error in this interval by comparing the predicted data with the ground truth. The spatially averaged RMS prediction error at time n is evaluated according to

$$\mathbf{e}(n) = \frac{\|\tilde{\mathbf{u}}(n) - \mathbf{u}(n)\|}{\sqrt{\langle \|\mathbf{u}(n)\|^2 \rangle_n}}. \tag{10}$$

- Steps 2 and 3 are repeated for the next point in the set, n_{k+1}.

The reservoir and model hyperparameters are listed in Table 1. Figure 6 shows the results of forecasting a single interval. Panel (a) of Fig. 6 shows a direct numerical solution of Eq. (9) for the value of X_j (represented on a color scale) as a function of spatial position j (vertical axis) and of time Λt plotted along the horizontal axis where Λ denotes the maximum Lyapunov exponent of the chaotic process. (Note the wave-like behavior visible in this pattern. This wave-like behavior is purposely

Table 1 Lorenz 96 prediction hyperparameters

Hyperparameter	Value	Hyperparameter	Value
ρ	0.6	D	5000
σ	0.1	T	80,000
ξ	32	β	10^{-4}
l	2	τ	1000

induced by Lorenz's design of the model so as to mimic the presence of atmospheric Rossby waves.) The largest Lyapunov exponent for this system with parameters $N = 40$, $F = 8$ is $\Lambda = 1.4$. See Karimi and Paul (2010). Panel (b) shows the error in the ML prediction starting at time zero, where the error is the ML predicted value of X_j (plotted in panel (b)) minus the 'true' value of X_j (plotted in panel (a)). We see that, for this case, a useful forecast (error near zero over a significant spatial region) is obtained out to about four Lyapunov times. Panel (c) shows the spatially averaged RMS error $e(t)$ corresponding to panel (b) versus time, showing how that prediction quality degrades with time. In order to illustrate the typical forecasting quality for this system and the variance in the forecast quality on different parts of the attractor, Fig. 7 shows the RMS error for 50 trajectories of length τ (cyan curves) plotted along with the mean (black curve). Additionally, Fig. 8 shows the effect of changing the number (M) of parallel reservoirs (and thus, changing the computational power) on the quality of prediction. We see that longer forecasting times result as M increases. We emphasize that our results are for the case of perfectly accurate measurements, and that prediction quality degrades as the measurements are corrupted by noise. For example, for large enough noise the improvement of prediction with an increase of M from 5 to 20 seen in Fig. 8 might be absent when the noise becomes the prediction limiting factor.

4.6 Another Example: The Kuramoto–Sivashinsky Equation

We now report on tests of our parallel reservoir prediction scheme using the Kuramoto–Sivashinsky system defined by the partial differential equation,

$$\frac{\partial y}{\partial t} + y\frac{\partial y}{\partial x} + \frac{\partial^2 y}{\partial x^2} + \frac{\partial^4 y}{\partial x^4} = 0. \tag{11}$$

Here, $y(x, t)$ is a scalar field defined on the spatial domain $x \in [0, L)$ with periodic boundary conditions so that $y(x + L, t) = y(x, t)$. We numerically integrate Eq. (11) using a pseudo-spectral scheme described in Kassam and Trefethen (2005). The time-series data is sampled at $\Delta = 0.25$ and used to create a training data set with $T = 80{,}000$ time steps and a test data set of length 20,000 time steps. We follow the same

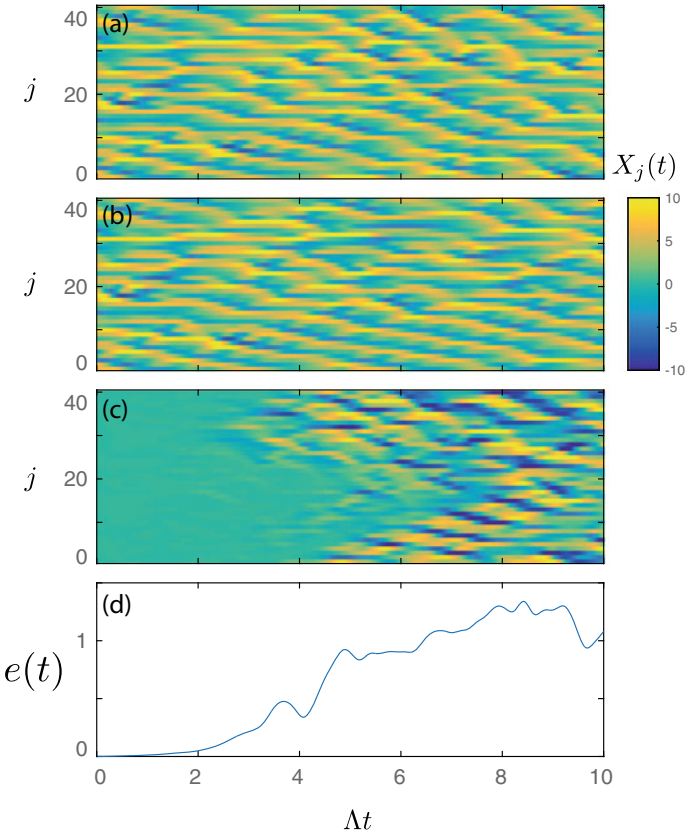


Fig. 6 Lorenz 96 prediction results for parameter value $F = 8$, $N = 40$. Panel **a** shows the true trajectory to be predicted by the reservoir. Panel **b** shows the reservoir prediction with $M = 20$ reservoirs and a locality parameter $l = 2$. Panel **c** shows the difference between the reservoir prediction and the true trajectory. Panel **d** shows the normalized RMS error $e(t)$ in the reservoir prediction as a function of time

validation procedure as outlined in Sect. 4.5 and test the accuracy of our forecasting results. Figure 9 shows the results for a single prediction interval. Figure 10 shows the variability in the prediction accuracy (as measured by the RMS error) on different intervals of the attractor. The effect of changing the number of reservoirs (M) used in the prediction setup is qualitatively similar to those resulting from our tests on the Lorenz 96 model, Eq. (9) shown in Fig. 8.

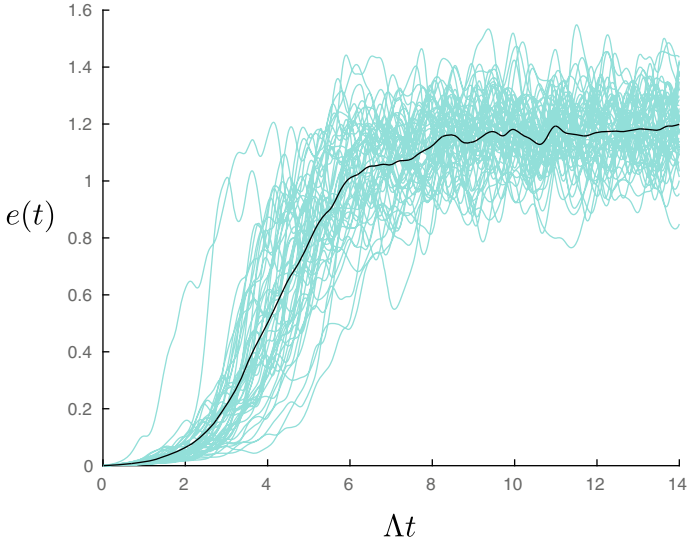


Fig. 7 RMS error in reservoir predictions over multiple intervals starting from different points on the attractor. The parameters of the Lorenz model are $F = 8$ and $N = 40$. The reservoir system is composed of $M = 20$ reservoirs with hyperparameters given in Table 1

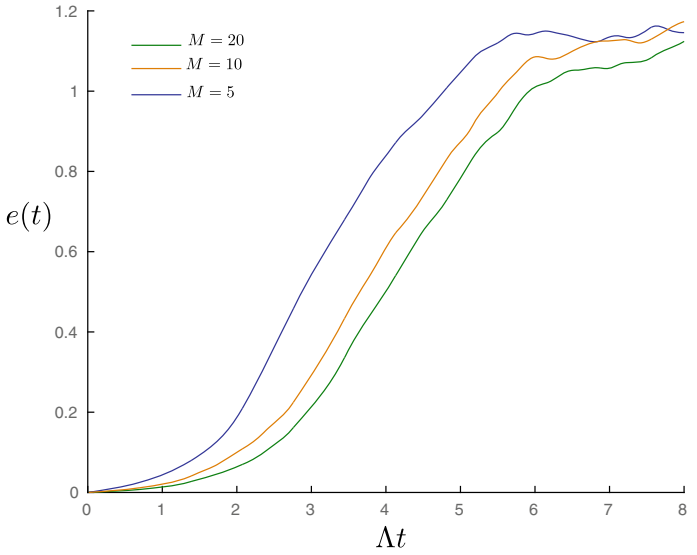


Fig. 8 RMS error $e(n)$ averaged over 50 prediction intervals in the Lorenz 96 prediction for parameter value $F = 8$, $N = 40$. The number of reservoirs used (M) in the parallel prediction is as indicated in the legend

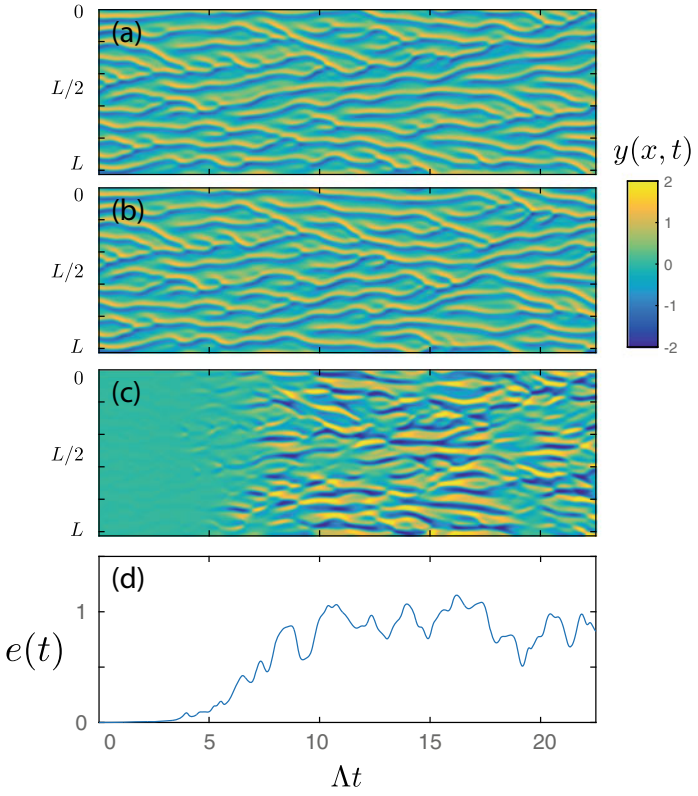


Fig. 9 Kuramoto–Sivashinsky prediction results. Panel **a** shows the true trajectory to be predicted by the reservoir. The periodicity length is $L = 100$. The $K = 256$ time series is predicted using $M = 32$ reservoir and a locality parameter $l = 6$. Panel **b** shows the reservoir prediction. Panel **c** shows the difference between the reservoir prediction and the true trajectory, i.e., **b** minus **a**. Panel **d** shows the normalized RMS error $e(t)$ in the reservoir prediction as a function of time

5 Hybrid Forecasting

We next consider the important and frequently encountered situation where a physical, knowledge-based model of a dynamical system is available but is imperfect in the sense that its dynamics deviates from that of the system that it is meant to model. This kind of model error can significantly degrade the predictions made by such a knowledge-based model. In this section, we show that machine learning can be a very useful tool for mitigating deficiencies of typical knowledge-based prediction systems. The hybrid forecasting configuration used in this section was introduced in Pathak et al. (2018b).

Figure 11 illustrates our scheme for implementing a hybrid machine learning setup that combines an imperfect knowledge-based model of a dynamical system with a

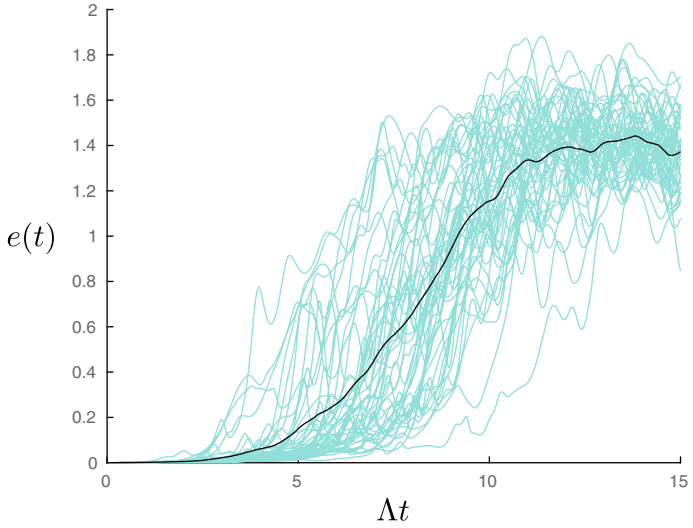


Fig. 10 RMS error in reservoir predictions over multiple intervals starting from different points on the attractor. The periodicity length of the KS system is $L = 100$. The reservoir system is composed of $M = 32$ reservoirs with hyperparameters given in Table 2

Table 2 Kuramoto–Sivashinsky system prediction hyperparameters

Hyperparameter	Value	Hyperparameter	Value
ρ	0.6	D	5000
σ	1	T	80,000
ξ	32	β	10^{-4}
l	6	τ	1000

reservoir-computing-based machine learning setup. In the next section, we describe the training and prediction scheme illustrated in Fig. 11.

5.1 Training

We assume that we have a training data set $\mathbf{u}(n)$, $-T \leq n \leq -1$, of measurements from the dynamical system of interest that have been sampled on a time interval Δ. Further, we assume that we have an imperfect model of the dynamical system denoted by $\mathcal{M}$, so that

$$\tilde{\mathbf{u}}_{\mathcal{M}}(n+1) = \mathcal{M}[\mathbf{u}(n)] \tag{12}$$

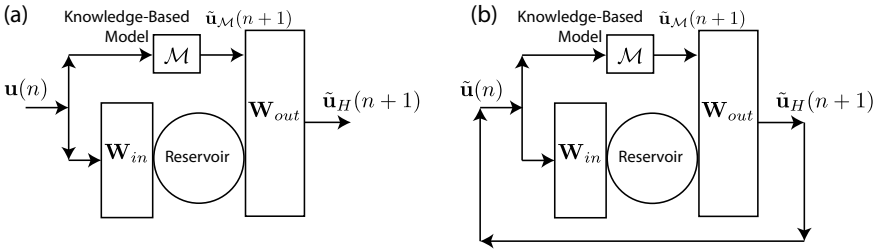


Fig. 11 Hybrid Forecasting Scheme for combining the reservoir prediction with an imperfect knowledge-based model. During the training phase **a**, the system is in an open-loop configuration while in the prediction phase **b**, the system is in a closed-loop configuration

gives us an approximate Δ-step prediction of $\mathbf{u}(n)$. We construct a reservoir with D nodes connected according to the adjacency matrix $\mathbf{A}$ and denote the state of the reservoir by $\mathbf{r}$. In the interval $-T \leq n \leq -1$, we evolve the reservoir according to the equation,

$$\mathbf{r}(n+1) = \tanh(\mathbf{A}\mathbf{r}(n) + \mathbf{W}_{\text{in}}\mathbf{u}(n)), \tag{13}$$

and collect the states $\mathbf{r}(n)$ for $-T \leq n \leq -1$. We also collect the imperfect model forecasts $\tilde{\mathbf{u}}_{\mathcal{M}}(n+1) = \mathcal{M}[\mathbf{u}(n)]$ for $-T \leq n \leq -1$. Let $\mathbf{h}(n)$ be the vector formed by concatenation of the reservoir state and the imperfect model forecast as $\mathbf{h}(n) = [\mathbf{r}(n); \tilde{\mathbf{u}}_M(n+1)]$. The trained output weights of the hybrid reservoir forecasting system are calculated similar to Eq. (6) so that

$$\mathbf{W}_{\text{out}}\mathbf{h}(n+1) = \tilde{\mathbf{u}}_H(n+1). \tag{14}$$

Note that, by the minimization carried out by the training procedure, it is reasonable to think of this output as a semi-optimal combination of the ML component and the imperfect knowledge-based component.

5.2 *Prediction*

After the training weights have been computed, we can start prediction at any time $n_0 \geq 0$. For example, if $n_0 > \xi$ we do the following steps:

- Step 1: Synchronize the hybrid system to the ground truth for ξ steps by running the open-loop system, Eqs. 12, 13, from $n = n_0 - \xi$ to $n = n_0$.
- Step 2: Predict for the next τ ($\gg \xi$) steps using the closed-loop system illustrated in Fig. 11b and described by the equations,

$$\tilde{\mathbf{u}}_H(n) = \mathbf{W}_{\text{out}}[\mathbf{r}(n); \tilde{\mathbf{u}}_{\mathcal{M}}(n+1)], \tag{15}$$

$$\mathbf{r}(n+1) = \tanh(\mathbf{A}\mathbf{r} + \mathbf{W}_{\text{in}}\tilde{\mathbf{u}}_H(n)). \tag{16}$$

5.3 An Example: Kuramoto–Sivashinsky Equations

We demonstrate the hybrid prediction setup using the Kuramoto–Sivashinsky equation. In this section, we consider training data generated by numerically simulating Eq. (11). Let the imperfect model be given by the following equation:

$$\frac{\partial y}{\partial t} + y\frac{\partial y}{\partial x} + (1+\epsilon)\frac{\partial^2 y}{\partial x^2} + \frac{\partial^4 y}{\partial x^4} = 0 \tag{17}$$

with $x \in [0, L)$ where L is the periodicity length. In Eq. (17), the parameter ϵ describes how closely the equation models the true dynamical system given by Eq. (11). A larger value of ϵ indicates a larger model error, and thus, a less accurate model. We consider a KS system with $L = 35$. Figure 12 illustrates the advantage of the hybrid forecasting scheme over either a pure machine learning approach or the imperfect model by itself. The top panel shows the result of the direct numerical solution of the KS equation (i.e., Eq. 11). The next three panels [(a), (b), (c)] show the error of the reservoir prediction [panel (a)], of the imperfect model [panel (b)] and of the hybrid [panel (c)] for a case with a moderate size reservoir ($D = 5000$ nodes) and a relatively small amount of model error ($\epsilon = 0.01$). We see from panels (a)–(c) that the predictions from the reservoir and from the imperfect knowledge-based model both yield prediction results that are of reasonable value (duration of useful predictions lasting about 1.5 and 2.3 Lyapunov times, respectively), but that the hybrid yields a substantially longer duration of useful prediction (about 6.2 Lyapunov times) than either of its two components. Panels (d), (e), and (f) are for a situation in which the reservoir is substantially smaller ($D = 500$) and the error in the knowledge-based model is substantially greater ($\epsilon = 0.1$). Panels (d) and (e) show that for this case, the predictions of both the reservoir and the knowledge-based model are fairly worthless. Nevertheless, when these two components are combined in a hybrid, they yield substantial prediction power as indicated in panel (f) (i.e., prediction time of about 4.5 Lyapunov times).

6 Parallel/Hybrid Forecasting

For our goal of using machine learning to improve forecasting of large complex spatiotemporally chaotic systems, we believe that the most useful strategy will be to combine the parallel approach of Sect. 4 with the hybrid approach of Sect. 5. This combination (1) will, via the parallelization, effectively exploit the LSTCI properties

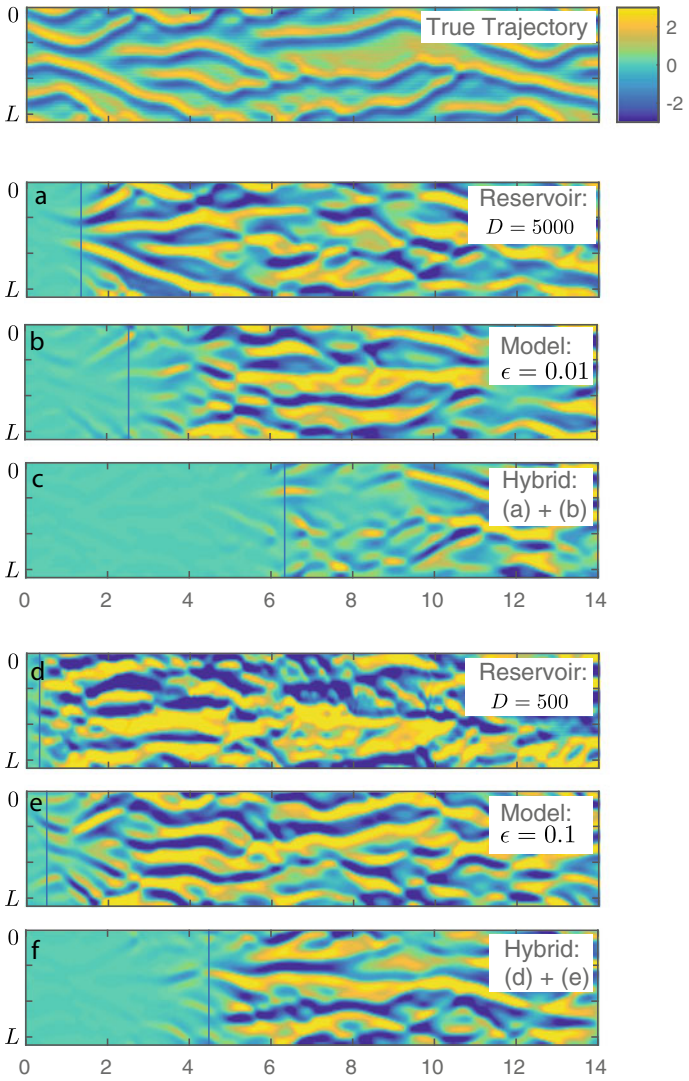


Fig. 12 Top Panel: True trajectory of the KS equation being predicted (Eq. (11) with $L = 35$). Panels **a**–**f** are forecast errors using the indicated schemes. Panels **a** and **d** are prediction errors using a reservoir-only scheme with the indicated reservoir size. Panels **b** and **e** are prediction errors using only the knowledge-based model with the indicated model error (ϵ). Panels **c** and **f** are the prediction errors upon using the hybrid scheme that combines the reservoir and the knowledge-based schemes. Panel **c** combines a reservoir of size $D = 5000$ with the knowledge-based model with error $\epsilon = 0.01$. Panel **f** combines a reservoir of size $D = 500$ with the knowledge-based model with error $\epsilon = 0.1$

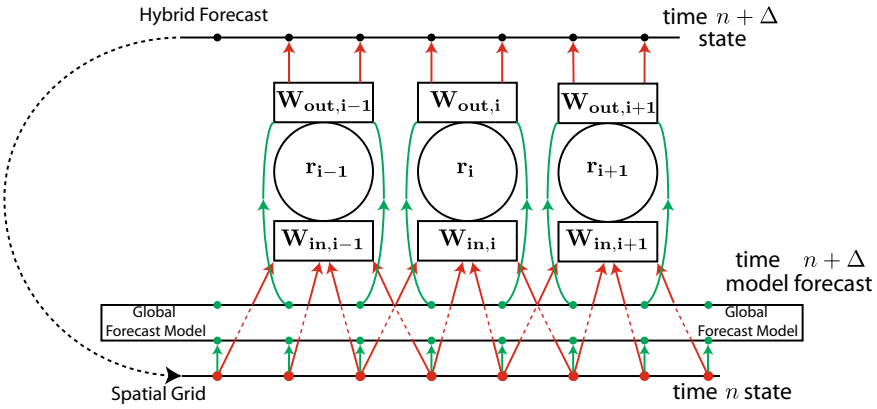


Fig. 13 A suggested architecture for a parallelized hybrid forecasting scheme. The machine learning component is implemented in a manner similar to Sect. 4, with an additional global forecast model to assist in the forecast

of the dynamical system and allow for computational efficiency in the machine learning component with respect to reservoir size, training, and amount of training data, and (2) will, via our hybrid scheme, provide a very effective platform for simultaneously utilizing data and first-principles system knowledge embodied by an imperfect global model.

Figure 13 shows a possible implementation of such a parallel/hybrid forecasting scheme. The spatial grid is partitioned similar to Sect. 4 (Figs. 2, 5). Additionally, a global knowledge-based forecast model (shown as the horizontally oriented long, thin rectangle) makes predictions which are distributed to the reservoirs and combined with the machine learning forecasts similar to Sect. 5 (Fig. 11). Since the knowledge-based predictions are global, if there is any aspect of the dynamics for which our LSTCI assumption (used in the ML parallelization) is deficient, we expect that the training process will allow potentially reasonable modeling of such an aspect via the knowledge-based model. Note that the grid for the ML component need not be the same as that for the global knowledge-based component, and that the ML grid density can be inhomogeneous. This freedom can be utilized by making the ML grid denser than that of the global knowledge-based component to provide extra resolution, or by restricting a denser ML component to a limited area for regional forecasting. The parallel/hybrid forecasting approach outlined here has been examined in further detail in Wikner et al. (2020).

7 Conclusion

In this chapter, we have considered the situation in which one desires to forecast the evolution of a large complex spatiotemporally chaotic system for which there is some, possibly incomplete and/or inaccurate, descriptive knowledge that can be used to

formulate an imperfect computational model. The limitations of such a computational model may stem from deficiencies in our knowledge of basic processes determining the system evolution, or of computational feasibility of modeling such processes (e.g., in situations where there is a very wide range of relevant scales), or a combination of these. While a model of this type may have deficiencies, we wish to utilize whatever capabilities it has to further our end of forecasting.

On the other hand, machine learning purely from past time-series data of an evolving dynamical system has had success in forecasting for certain situations. However, when systems are large and complex and the system state description to be forecasted is correspondingly large and complex, it appears that a purely machine learning approach might often not be feasible.

Thus, it makes sense to try and combine these two very different approaches. The combination of the two approaches may potentially be capable of outperforming either one of them acting alone. Even so, implementation of an ML system combined with a knowledge-based computational model still faces a substantial challenge due to the size and complexity of the states that we desire to forecast. Thus, in this chapter we have addressed what we believe are two key issues in this approach to forecasting large complex spatiotemporally chaotic systems: (i) how to wed (hybridize) machine learning with an imperfect knowledge-based computational model, and (ii) how to parallelize the machine learning component in combination with global knowledge-based code, in a manner feasible for the machine learning component. Our review of these two key issues leads us to conclude that preliminary results provide a possible path that may be effective in enabling improved forecasting of large complex spatiotemporally chaotic systems.

However, we emphasize that many issues remain. The task of weather forecasting provides a basis for assessing difficulties and directions for future work aimed at ultimately using this hybrid/parallel approach for forecasting large, complex, spatiotemporal systems. Primary among these is the issue of cyclic prediction and data assimilation. Typically, a new set of weather forecasts is made every 6 h. At the beginning of each 6 h cycle, new measurements of the atmospheric state are used to correct an estimate of the probability distribution of the atmospheric state provided by the 6 h forecast from the previous cycle, and the new probability distribution estimate is used as an initial condition for an atmospheric model, which is then integrated forward to make a new set of probabilistic forecasts. The process by which the previous forecast is combined with the new data is called ‘data assimilation’. Moreover, the nature of the data for weather forecasting is itself complex, resulting from measurements with stochastic errors from an array of different types of diagnostic sources (e.g., balloons, satellites, ground stations, ships, aircraft, and radar), and these data sources can vary greatly in space and time (e.g., balloon measurements are typically dense over technologically developed regions, typically less dense over technologically less developed or more sparsely populated regions, and typically very much less dense over oceans).

Thus, to proceed past the preliminary promising results of this chapter, many issues such as incorporation of cyclic data assimilation and data source heterogeneity await. We hope, however, that this chapter will provide a basis for moving forward in this area.

Acknowledgements We thank our colleagues, particularly Brian Hunt, Michelle Girvan, and Istvan Szunyogh, for their contributions. We also acknowledge support from DARPA contract HR00111890044.

References

H. Abarbanel, *Analysis of Observed Chaotic Data* (Springer Science & Business Media, 2012)
V. Afraimovich, N. Verichev, M.I. Rabinovich, Radiophys. Quantum Electron. **29**, 795 (1986)
S.L. Brunton, J.L. Proctor, J.N. Kutz, Proc. Natl. Acad. Sci. **201517384** (2016)
S. Hochreiter, J. Schmidhuber, Neural Comput. **9**, 1735 (1997)
H. Jaeger, GMD Technical report **148**, 13. German National Research Center for Information Technology, Bonn, Germany (2001)
H. Jaeger, H. Haas, Science **304**, 78 (2004)
H. Kantz, T. Schreiber, *Nonlinear Time Series Analysis*, vol. 7 (Cambridge University Press, Cambridge, 2004)
A. Karimi, M.R. Paul, Chaos: Interdiscipl. J. Nonlinear Sci. **20**, 043105 (2010)
A.-K. Kassam, L.N. Trefethen, SIAM J. Sci. Comput. **26**, 1214 (2005)
L. Kocarev, U. Parlitz, Phys. Rev. Lett. **76**, 1816 (1996)
E.N. Lorenz, in *Proceedings of Seminar on Predictability*, vol. 1 (1996)
Z. Lu, B.R. Hunt, E. Ott, Chaos: Interdiscip. J. Nonlinear Sci. **28**, 061104 (2018)
B. Lusch, J.N. Kutz, S.L. Brunton, Nat. Ccommun. **9**, 4950 (2018)
E. Ott, *Chaos in Dynamical Systems* (Cambridge University Press, Cambridge, 2002)
E. Ott, T. Sauer, J.A. Yorke, *Wiley Series in Nonlinear Science* (Wiley, New York, 1994)
J. Pathak, B. Hunt, M. Girvan, Z. Lu, E. Ott, Phys. Rev. Lett. **120**, 024102 (2018a)
J. Pathak, A. Wikner, R. Fussell, S. Chandra, B.R. Hunt, M. Girvan, E. Ott, Chaos: Interdiscip. J. Nonlinear Sci. **28**, 041101 (2018b)
L. Pecora T. Carroll, J. Heagy, *Handbook of Chaos Control*, vol. 227 (1999)
M. Raissi, P. Perdikaris, G. Karniadakis, J. Comput. Phys. **378**, 686 (2019)
N.F. Rulkov, M.M. Sushchik, L.S. Tsimring, H.D. Abarbanel, Phys. Rev. E **51**, 980 (1995)
T. Sauer, J.A. Yorke, M. Casdagli, J. Stat. Phys. **65**, 579 (1991)
P.R. Vlachas, W. Byeon, Z.Y. Wan, T.P. Sapsis, P. Koumoutsakos, Proc. R. Soc. A **474**, 20170844 (2018)
Z.Y. Wan, P. Vlachas, P. Koumoutsakos, T. Sapsis, PloS One **13**, e0197704 (2018)
A. Wikner J. Pathak, B. Hunt, M. Girvan, T. Arcomano, I. Szunyogh, A. Pomerance, E. Ott, Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems (2020), arXiv:2002.05514 [cs.LG]
R.S. Zimmermann, U. Parlitz, Chaos: Interdiscip. J. Nonlinear Sci. **28**, 043118 (2018)

Part II
Physical Implementations of Reservoir Computing

Reservoir Computing in Material Substrates

Matthew Dale, Julian F. Miller, Susan Stepney, and Martin A. Trefzer

Abstract We overview Reservoir Computing (RC) with physical systems from an Unconventional Computing (UC) perspective. We discuss challenges present in both fields, including *encoding* and *representation*, or how to manipulate and read information; ways to search large and complex configuration spaces of physical systems; and what makes a "*good*" computing substrate.

1 Introduction

As of 2018, there are many "flavours" and interpretations of physical reservoir computing systems. A recent review (Tanaka et al. 2019) classifies and groups these systems according to reservoir types based on physical properties exploited, such as chemical, optical, and mechanical based. These systems can vary immensely in terms of architecture, dynamics, degrees of freedom, ease to manipulate, size, complexity, and internal timescales. However, a general framework for physical reservoir computing and unconventional computing is still missing. The wide variety of potential computing systems presents several challenges: how best to design physical systems, how to determine what computational tasks they are best suited to, and how to assess and compare across different architectures.

In this chapter, we discuss three important aspects for describing any physical [reservoir] computing system: representation and instantiation; manipulation and programming; and what makes a "good" computing substrate. No matter what future

M. Dale · S. Stepney (✉)
Department of Computer Science, University of York, York, UK
e-mail: susan.stepney@york.ac.uk

M. Dale
e-mail: matt.dale@york.ac.uk

J. F. Miller · M. A. Trefzer
Department of Electronic Engineering, University of York, York, UK
e-mail: julian.miller@york.ac.uk

M. A. Trefzer
e-mail: martin.trefzer@york.ac.uk

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_7

systems emerge, these three aspects will always play a pivotal role in the design and application of any physical (and virtual) reservoir computing system.

Physical systems tend to have high degrees of freedom or ways to be configured. Whether each configuration induces a small or large change in dynamical behaviour is more important to the system's computing ability. A substrate that can compute many different problems will possess a high degree of dynamical freedom, with the ability to instantiate many "reservoirs" in a single substrate.

Here, we use the term *reservoir* to refer to a specific configuration of some non-linear system, substrate, or device that facilitates computation. A configuration is a set of values of the parameters that control and influence the behaviour of the system. For reservoirs, these parameter values are typically static: they are held constant during the operation of the reservoir. In a simulated recurrent neural network, parameters may be weights or global scaling parameters. In an optical delay-based system, parameters define the sample-and-hold procedure, the length of a fibre optic delay line, or injection currents that shape the non-linearity of the system. In other systems, parameters include cell update rules in a cellular automaton, the volume of the reactor in a chemical reaction system, and the strength of the coupling between neighbouring oscillators in a mechanical oscillator network.

A reservoir is thus an abstract *representation* of a physical or simulated system, which is realised through the instantiation or configuration of physical parameters.

To successfully compute with a reservoir, in general, requires a physical system with a number of dynamical properties that exist internally, which can be driven by an external input signal. These properties, either present in a "natural" state or in a configured state, govern the information processing capabilities and capacity of the system: how information is stored, transmitted, distributed, and processed.

There are three known basic properties for reservoir computing. The *echo state* property (Jaeger 2001a) represents a fading memory, an essential property for learning time-dependent relationships for the prediction of future states based on previous states. The *separation* property represents a system's ability to project the input space into the high-dimensional phase space of the system. The *approximation* property (Maass et al. 2002) is the reservoir's ability to generalise given similar, or noisy, input signals, that is, to converge to the same attractor in terms of dynamics. Other "basic" properties may exist that are at present unknown.

Dambre et al. (2012) show that all dynamical systems featuring such properties possess the same total normalised capacity to process information. They identify and explain the importance of the non-linearity and memory trade-off for computing tasks, as well as the detrimental effect noise has on the computational capacity of dynamical systems. These insights, demonstrated with three dynamical systems, validate the RC framework's application to a range of underlying dynamical systems.

What type of dynamical systems the RC framework can be applied to is fairly broad, covering both discrete and continuous systems. However, common patterns exist for all systems. In each, information is processed via an *intrinsic* function, the physical system dynamics that transforms and maps input signals into observable system states. The details of how this intrinsic function works are generally irrelevant to the reservoir's programming process. The substrate, and subsequent reservoir, is a

black box. The model harnesses potentially unknown physical processes, performing functions on macroscopic behaviours resulting from unseen microscopic interactions.

In terms of programming physical and unconventional computers, the configuration, input, and output methods encompass the unconventional "program" of the system. Unlike conventional programs, instructions induce and exploit the intrinsic dynamics and state of the system without explicitly telling the system how to do so, or what to do at the lowest physical level. These programs are typically "learned" rather than hand-crafted. For example, the training of RC systems can be partitioned into multiple stages. Substrate parameters may be selected randomly (typically within a constrained domain) or trained through some optimisation process (Abubakar et al. 2018). The readout stage is typically learnt through linear regression, reducing the error between the reservoir output and the desired output signal. In most cases, this layer is simply a linear weighted combination of system states; however, more complex and non-linear readouts (and training methods) are possible.

The variety of intrinsic properties exploitable from physical systems—how they store and process information—is what makes these systems useful, powerful, and diverse. With physical systems, there are greater possibilities for faster, less-expensive, and more energy-efficient computing. This is in part due to less top-down design, fewer data conversion steps and constraints, removal of traditional data transfer bottlenecks, and architectures that tend to be more robust and less error-prone. However, there are still many gaps left in our understanding of what makes these systems compute.

In the remainder of this chapter, we highlight areas of unconventional computing to discuss and map out challenges and areas still requiring further development. We outline the importance of representation and instantiation in Sect. 3 and discuss the programming of reservoirs in Sect. 4. In the final section, we discuss a newly proposed framework to evaluate what makes a good computing substrate, providing a new perspective on how to build and compare physical reservoir computers.

2 Computing with Physical Systems

Biological systems vastly outperform certain aspects of classical computing paradigms, from possessing inherent fault-tolerance to forming highly parallel machinery. Much of this performance is achieved by exploiting physicality and embodiment (Stepney 2007), sharing and distributing computational effort throughout the system, from small-scale micro-organisms to large-scale swarms. Biological systems exploit physical interactions through feedback with the real world, utilising features such as morphology and direct and indirect changes to the environment. Many of these systems comprise simple elements (e.g., molecules and cells) that emerge and coalesce into more complex (e.g., multi-cell organisms and ecosystems), but robust, structural layers working across different spatial and temporal scales. Such grounding properties (and many more) have enabled these complex systems to thrive and evolve, adapting and co-evolving with their local ecosystem.

In modern science and engineering, there are many attempts to mimic the computational properties, efficiency, and behaviours of biological systems on conventional machines. In many ways, such attempts are flawed, or at least inefficient. Techniques and models attempt to imitate the performance of an embodied system in a non-embodied abstraction, in a process that requires an often cumbersome and inefficient transformation to a symbolic representation. In essence, such transformations detract from many of the physical aspects that make natural systems so powerful.

The limitations of conventional computing paradigms are well defined, and we are rapidly approaching the limitations of current CMOS technology (Lloyd 2000). For technology to move forward, many of these limitations need to be overcome, by using the same classical paradigms or alternative ones.

The conventional von Neumann computing architecture, based on the stored-program computer concept, although expertly refined over decades, has some fundamental inefficiencies. For example, classical computers require the transformation between high-level languages to low-level machine code, a process that requires layers of conversion through a compiler stack, making it computationally costly, slow, and highly susceptible to faults and errors. These systems typically succumb to many issues in speed, from both an inability to deal with concurrent computations and the bottleneck created by the transfer of data between separate memory and processing entities. Because of these architectural weaknesses, other intertwined issues arise, such as an increase in power consumption, system size, and design complexity.

2.1 Unconventional Computing

The field of *unconventional computing* has for many years attempted to address the limitations of conventional computing by providing alternative architectures, systems, and models that typically exploit the underlying physics and many-scale interactions of the real world. Many forms of unconventional systems now exist, from the mathematical reversible and chaos computing concepts to physical–chemical and neuromorphic computing. Recent collections of theory and practice of unconventional systems include (Adamatzky 2016a, b; Stepney et al. 2018).

Unconventional systems have a long history. Some systems link back to the mid-twentieth century cybernetics movement, such as Pask's experiments with electrochemical assemblages in ferrous sulphate solutions (Cariani 1993; Pask 1959). Others include Turing's unorganised machines (A- and B-type random networks) (Turing 1969) constructed from simple components in a random structure capable of learning. Even further back there are analog computing models and systems such as the differential analyser (early twentieth century), Babbage's Difference Engine (nineteenth century), slide rules, Orrerys (mechanical models of the solar system), and astronomical clocks.

Pask's work—a novel example of unconventional systems and methods at the time—was conducted exclusively in the physical domain and sought to evolve functional dendrite-like structures by passing current directly through immersed elec-

trodes. To alter the growth of these structures, he selected which electrodes to pass current between, with the resulting linkage conductance representing synaptic weights. In his experiments, Pask manually selected and "evolved" linkages sensitive to perturbations caused by sound, or magnetic fields, to create a self-assembling "ear" that could be trained to discriminate between different frequencies.

Turing's work, on the other hand, was more theoretically based, drawing on analogies with the human brain. What makes Turing's machines particularly interesting here is the clear analogies to the current reservoir computing paradigm.

In general, exploiting computation directly at the substrate level, as biological systems do, is expected to offer advantages over classical computing architectures, such as exploiting physical and material constraints that could offer solutions "for free", or at least computationally cheaper (Stepney 2008). Extracting computation from these systems and physically *programming* them, however, is challenging. Sometimes, this is further complicated by a desire for minimal abstraction: exploiting emergent physical phenomena directly whilst maintaining a level of programmability that enables the system to perform a variety of tasks.

Hybrid digital–analog computers potentially solve this programming problem, where the digital system is trained to extract computation performed implicitly from an analog substrate. This is typically where many physical reservoir computers align, with the readout and training often carried out in the digital domain. An excellent example of a programmable hybrid system is Mills' Kirchhoff–Lukasiewicz Machines (KLM) (Mills 1995) based on Rubel's computational model of analog computation (the extended analog computer Rubel 1993). In Mills' work, the KLM device is controlled by a specially designed vector of bits, referred to as the "overlay" (Mills 1995). This overlay, representing the semantic "program", is used to define the reconfigurable layer that exploits the implicit material function. The computational functions being utilised are therefore a result of the material's configuration, typically achieved through the selection of inputs, outputs, and control functions/signals. Mills describes this as a paradigm of *analogy* (Mills 2008), where the computing device is not explicitly told to perform an operation and provide a readable output, but rather trained to exploit an implicit function that results from the material's configuration. Thus, it is an analogy of the program, rather than an algorithmic function to be implemented.

For analog computers, the program and architecture may be indistinguishable or inseparable: to program the machine requires a change in the machine architecture. However, placing additional layers within the machine's architecture may reduce the amount of change required; for example, adding a standardised reconfigurable "middle" layer or a trainable readout in reservoir computing.

2.2 Configuring Physical Systems to Compute

Conventional computers are designed to be substrate-independent, where a symbolic virtual "machine" is engineered into physical hardware. Yet, not every com-

puting problem requires abstraction to a symbolic virtual machine, e.g., filtering, control problems, and solving differential equations. The digital computing pipeline from physical-to-abstract-to-physical tends to consume much more power than an equivalent analog counterpart, be constrained by serial-processing, and present many vulnerabilities in terms of security and coding errors.

Material/substrate-based computers are machines in which some computational process, or physical mechanism, may be extracted from a behaviourally diverse dynamical system. In essence, information processing can be exploited from what the substrate does naturally, for example, how the system reacts and dynamically adapts to some input stimulus (Stepney 2008). Informally speaking, this can be viewed as "kicking" the dynamical system and observing its behaviour to some given stimulus, where the method of perturbation and observation may vary in type: electrical, mechanical, optical, etc.

Given some material has potential properties useful for computing, the question is whether they are extractable, and whether the system can be trained, configured, or engineered to consistently exploit these properties.

Before the first physical reservoir computers emerged, a computing paradigm reminiscent of Pask's self-assembling ear was developed for novel substrates. This conceptual idea was named "evolution *in materio*" (EiM) by Miller and Downing (2002). Inspired by Thompson's seminal work on "intrinsic" hardware evolution (Thompson 1997), the principal idea is to use artificial evolution as a search method to find configurations that directly exploit the computational properties of complex materials.

To date, work with EiM has explored liquid crystal substrates, composites of randomly dispersed carbon nanotubes mixed with polymers, carbon nanotubes mixed with liquid crystal, and networks of gold nanoparticles (Broersma et al. 2017; Harding and Miller 2004; Massey et al. 2016; Miller and Downing 2002). Each of these substrates is evolved to achieve interesting computational properties on a variety of specific tasks, without it being known exactly how best to program them to perform those tasks. Applying evolved mappings and external "control" signals, the unknown internal properties of the composites are configured to produce physical solutions to computational tasks.

Current progress of EiM and physical reservoir computing still remains at the substrate level: single devices/systems and simple architectures. This limits the complexity and types of tasks that are solved. Solving problems with increased complexity will require layers of (non-symbolic) abstraction, hierarchy, and possibly multi-substrate designs, whilst maintaining substrate-level exploitation and efficiency. How to implement and program such architectures is still unclear.

Another non-trivial task is how to analyse what is being exploited, intrinsically, externally, and in terms of general computational and dynamical properties of the substrate. This closely links with how to determine if a substrate is suitably "rich" to solve computational tasks; and, if the substrate is suitable, what class of tasks is appropriate.

At the basic level, a generic methodology to characterise the substrate, suitable architectures, and higher-level constructs that naturally fit the substrate is missing.

For example, in EiM, often too little is known about the substrate being evolved; in principle this is the point, but in practice it limits its full potential. A minimal criterion for evolution to excel, or for reservoir computing to work, needs to be established. At the same time, the realistic potential of the substrate needs to be determined and categorised. This includes whether a substrate is compatible with other substrates, what role it can fulfil in a hierarchy or high-level program, and what other basic physical limitations exist.

3 Reservoir Computing with Physical Systems

A key advantage to using the reservoir computing paradigm is that there is no requirement to control individual elements within the system. This makes it applicable to many complex structures where the exact arrangement and manipulation of internal elements are either too time-consuming, too delicate/complex to implement, or impossible to achieve. However, to determine whether a substrate is computationally exploitable depends upon both underlying physical characteristics and the observable phase space.

In many cases, a physical substrate is computationally useful only when configured or perturbed. Therefore, forming a useful reservoir requires the tuning of physical parameters. This itself implies that a single substrate instance can realise a range of reservoirs of varying quality through different physical configurations.

Using the reservoir model, we argue that any substrate and subsequent configuration is represented by its abstract "quality" in a space of all possible reservoirs, and that this space is very different from the configuration space. Methods for "pre-training" reservoirs (or selecting appropriate parameter ranges) to navigate this reservoir space are therefore essential tools to find and discover functional, possibly optimal, reservoirs within all possible material states. Figure 1 shows a conceptual representation of a single substrate's space of all possible physical configurations and its abstract reservoir equivalence.

Different substrates have their own such spaces; some perhaps have no functioning reservoirs; some perhaps possess many configurations in distant regions of the space that produce ideal properties for reservoir computing. When substrate "richness" and complexity increases, these complex structured spaces have a greater probability to be "deceptive" and difficult to navigate. Configurations close to an ideal solution may themselves be computationally uninteresting; working configurations may be unstable or critical when perturbed by noise and other external signals.

To create and navigate these spaces, basic mechanisms must be defined: how to *encode* and *represent* information flowing both in and out of the system. Depending on this, each space will vary drastically. For example, the size and density of functional reservoirs in these spaces will increase or decrease given different encodings. The combination of multiple encodings and representations will also have an effect, possibly leading to even larger and richer spaces. A prime example of how encoding affects computability is the delay-based reservoir computing technique using a

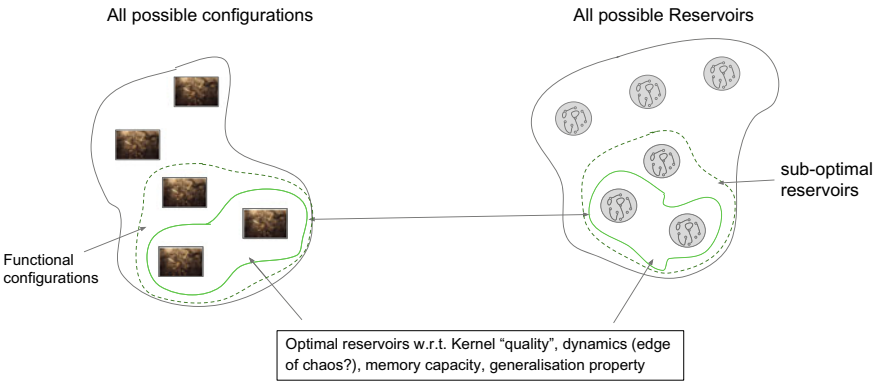


Fig. 1 Mapping between substrate configuration space and reservoir quality; from Dale (2015)

single non-linear node with delayed feedback, synonymous with optoelectronic and photonic systems (Appeltant et al. 2011, 2014; Brunner et al. 2013; Paquot et al. 2012). In these systems, the input encoding defines a network of virtual processing nodes in the time-domain rather than implementing a physical spatial network of nodes.

3.1 Encoding and Representation in Reservoir Computers

Conventional programs and algorithms represent idealised mathematical objects, irrespective of their underlying hardware. In a physical system, say a biological system, computation is embodied and behaviour is challenging, if not impractical, to capture using a closed mathematical model. As such, trying to program these embodied systems requires different techniques.

Unconventional computers have the potential to be faster, and/or consume less power, but in order to do so requires that the model of computation naturally fits the material rather than imposing an inappropriate model that fights its implementation. As Caravelli and Carbajal (2018) describe it: "the problem needs to be specified using the language of the computing device. Using the wrong language increases the difficulty of the problem, and consequently decreases performance."

Most unconventional systems have only a primitive, or even no explicit, computational model. This makes it difficult to build higher-order representations, leaving the programming side heavily underdeveloped. As a result, programs are typically stuck at the equivalent assembly language level, with all the problems associated with a lack of abstraction and usability.

Representation and encoding are therefore critical to future progress; identifying what representations work best, and designing high-level programs that naturally fit the reservoir model, need to be further developed.

3.2 Abstraction/Representation (A/R) Theory

To discuss the encoding and representation problem for physical reservoirs, we first need to define as to when a physical system computes.

In order to distinguish when computation is occurring in a physical system, as opposed to the system simply just "doing its thing", Abstraction/Representation (A/R) theory has been developed (Horsman et al. 2014, 2017, 2018). In A/R theory, *physical computing is the use of a physical system to predict the outcome of an abstract transformation.*

The theory identifies objects in the domain of *physical systems* (including computers), the domain of *abstract objects* (including computational models), and the *representation relation* which links the two.

The *representation relation* is primitive and maps from physical to abstract objects, $\mathcal{R} : \mathbf{P} \rightarrow M$, where $\mathbf{P}$ is the set of physical objects, and M is the set of abstract objects. When two objects are connected by $\mathcal{R}$, we write them as $\mathcal{R} : \mathbf{p} \rightarrow m_{\mathbf{p}}$. The abstract object $m_{\mathbf{p}}$ is then said to be the *abstract representation* of the physical object $\mathbf{p}$. Instantiation is a map from abstract object to physical object: $\widetilde{\mathcal{R}} : M \rightarrow \mathbf{P}$. When two objects are connected by $\widetilde{\mathcal{R}}$, we write them as $\widetilde{\mathcal{R}} : m_{\mathbf{p}} \rightarrow \mathbf{p}$. The physical object $\mathbf{p}$ is then said to be the *physical instantiation* of the abstract object $m_{\mathbf{p}}$.

Abstract evolution maps abstract objects to abstract objects, which we write as $C : m_{\mathbf{p}} \rightarrow m'_{\mathbf{p}}$. If we instantiate $m_{\mathbf{p}}$ in $\mathbf{p}$, then its corresponding physical evolution map is given by $\mathbf{H} : \mathbf{p} \rightarrow \mathbf{p}'$. If we now apply $\mathcal{R}$ to the outcome state of the physical evolution, we get its abstract representation $m_{\mathbf{p}'}$. See Fig. 2.

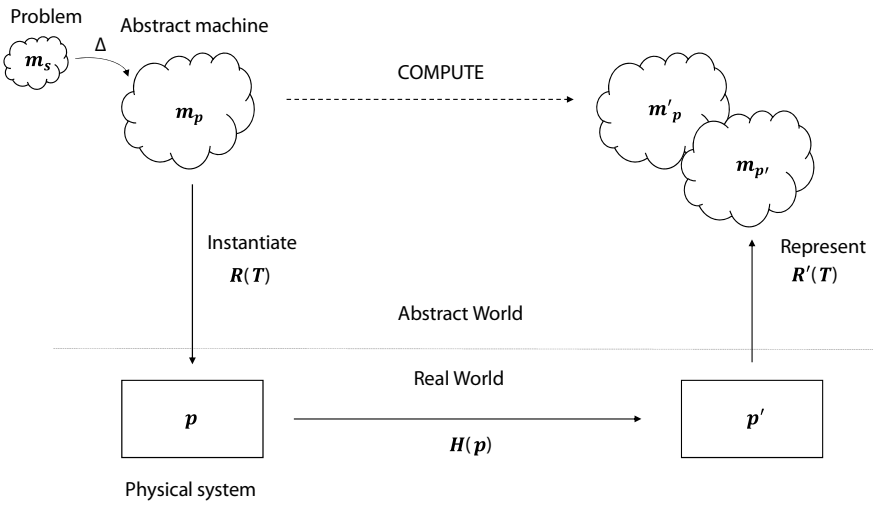


Fig. 2 Commuting diagram: mapping between abstract space and physical space. T represents appropriate theory for encoding and representation. Δ is translation of abstract problem into abstract machine. Other definitions are given in the text

We now have two abstract objects, $m'_{\mathbf{p}}$ and $m_{\mathbf{p}'}$. If these are *sufficiently close*, then we say that the physical system has computed the desired result of the abstract computation. For a *well-engineered* system, one where we can rely on this sufficient closeness, we no longer have to compare the two answers, and can take $m_{\mathbf{p}'}$ as the prediction of the desired result of the computation $m'_{\mathbf{p}}$.

To extend the theory to multiple physical systems, as many practical unconventional devices will most likely consist of, the *heterotic* computing framework has been developed (Horsman 2015; Kendon et al. 2011, 2015). The term "heterotic" captures the "hybrid vigour" of complex systems, where the whole is greater than the sum of its parts. Possible heterotic systems could consist of multiple devices and different computational models combined to create more expressive and programmable computers.

The concept of hybrid reservoir systems, "mixing and matching" different unconventional systems, both virtual and physical, is largely unexplored in the literature. However, on the surface, it looks promising for further research as the benefits of hierarchical reservoir systems come into light (Dale 2018a; Gallicchio et al. 2017).

3.3 Observing Reservoir States

Here, we define our representation $\mathcal{R}$ of physical system states. To interpret a physical substrate as a reservoir, the definition in Konkoli et al. (2018) is adapted, where the observed reservoir states $x(n)$ form a combination of the substrate's implicit function and its *discrete* observation:

$$x(n) = \Omega(\mathcal{E}(W_{\text{in}}u(t), u_{\text{config}}(t))) \tag{1}$$

where $\Omega(n)$ is the observation of the substrate's macroscopic behaviour and $\mathcal{E}(t)$ the continuous microscopic substrate function when driven by the input $u(t)$. W_{in} symbolises a set of input weights common in all reservoir systems. The variable $u_{\text{config}}(t)$ represents the substrate's configuration, whether that be through external control, an input–output mapping, or other method of configuration. Equation 1 is a simple case where no feedback is returned to the system. To add feedback, the input variables y and feedback weights W_{fb} are added to $\mathcal{E}(.)$.

This formalisation of the reservoir states separates the system into contributing parts, including the observation and configuration method, which as a whole represents the overall physical reservoir computer.

An exploitable feature of this definition is that there is no need for the input mechanism to be the same as the observation method. For example, an input may be encoded as an electrical signal, and the reservoir states observed as deformations in physical structure. This additional channel of communication leads to an increase in system bandwidth to exploit, allowing multiple measurement and stimulation techniques to be used in tandem.

In Eq. 1, note that the intrinsic substrate function ($\mathcal{E}$) is presumed to be fixed, clamped, or set by u_{config}. However, depending on the system, $\mathcal{E}$ may change when interacted with or observed (commonly known as the "observer" effect), and may therefore be non-deterministic.

When physical systems are connected to stimulation and recording equipment, it is critical not to overlook that the whole interface will, in some way, affect the computing process, which may or may not be included in the model. In most experimental works, it is imperative that the substrate being exploited is sufficiently isolated from the rest of the system, e.g., from its controlling equipment and its environment. In Dale (2018b), it was shown how evolution could find ways to include and exploit the interfacing equipment as part of the reservoir system. Despite its positive contribution, generally improving performance, the dynamics of the substrate became less important to the computing system. For cases such as this, we include the Ω function in Eq. 1, as the observation process itself may affect the system output.

This phenomenon, unique to physical substrates, is a trade-off problem, presenting itself more within certain systems. If the substrate is fixed permanently to the same measuring equipment, the contribution of the interface to the overall "computation" is considered less crucial. However, if the substrate is to be trained on one device and then applied on another, the substrate under-test should be the main, if not the only, contributor.

The final output $y(n)$, the last part of the system, is determined by the readout function g on the observation $x(n)$:

$$y(n) = g(x(n)) \tag{2}$$

In many reservoir systems, the readout function g tends to be a linear weighted combination of system states for RC, and once trained remains static.

As previously mentioned, the readout layer can be implemented within a different medium or domain from the reservoir substrate itself. For example, an analog material may be combined with a digital readout layer. This flexibility adds extra programmability, and even compatibility with other devices, but may come with specific costs and benefits.

At present, methods that take advantage of $u_{\text{config}}(t)$ and $g(x(n), n)$ changing with time are uncommon. Although this comes at the price of complexity, abstract programs where both implement functional primitives, providing higher-order representations in $y(n)$, are theoretically possible.

4 The Search for Reservoirs

The classical approach to programming and manipulating physical systems requires (to some extent) a good understanding of the properties and interactions within that system. This results in the traditional top-down programming approach. Reservoir computing and other "intelligent" systems apply alternative mechanisms, e.g.,

through training, learning, and heuristics. For each of these programming approaches, the details of the system are exploited in a controlled manner typically without explicit instructions from a human. For example, the approach has learned by itself where best to clamp global or local dynamics using a bias signal, or the exact strength and combination of control signals required to exploit and alter the structure.

The actual "programs" implemented by the substrate and the system as a whole, however, are different. The previous approaches therefore act as meta-programming approaches. Following the previous discussion, we can think of these programs as containing details about the encoding and representation of the system and its physical parameters. Different programs may exploit the same physical phenomena (e.g., electromagnetism, electrochemical, and optoelectronic) but encode and represent data using only specific physical parameters (e.g., electric fields), or, alternatively, use different encodings and physical parameters (e.g., magnetic fields and chemical reactions). This can lead to a combinatorically vast search space of potential computer programs, e.g., reservoirs on a single device.

Reservoir substrates typically possess fixed dynamics after applying carefully selected parameters and parameter ranges to exhibit desirable dynamical properties. Echo State Networks are an exemplar of this tuning problem, requiring global scaling parameters to induce specific dynamics, such as possessing the echo state property (Jaeger 2001a). With many unconventional systems, the parameter space is complex and therefore difficult to navigate. As complexity increases, simple (unconstrained) random configurations often result in sub-optimal performance, as the probability of stumbling upon optimal parameters is low. In general, no matter the substrate, good parameter selection and encoding—the basic program—are essential to produce high-performing reservoir systems.

In the RC community, many meta-programming approaches have been used to optimise the parameters of virtual reservoirs. For example, techniques include particle swarm optimisation (Basterrech et al. 2014; Sergio and Ludermir 2012), Bayesian optimisation (Yperman and Becker 2016), gradient-based information (Yuenyong 2016), multi-objective optimisation (Krause et al. 2010), and evolutionary algorithms (Chatzidimitriou and Mitkas 2010; Ferreira and Ludermir 2011; Jiang et al. 2008; Matzner 2017; Qiao et al. 2017). Other heuristics continue to be added; see the recent survey (Abubakar et al. 2018).

These approaches are simple to apply due to the RC framework's training partition. What often separates reservoir computers from other learning systems is this division in training. Reservoirs are typically generated or configured (partially programmed) using one technique, and the readout is trained using another (fully programmed), avoiding challenges when training complex networks.

Using this separation, a physical substrate can be partially programmed, then quickly retrained/programmed to solve different tasks by manipulating only the readout. This extra level of programmability provides advantages in time to train and sometimes performance.

The typical focus when programming is to optimise the substrate to a specific task. This approach provides little insight into the wider computing ability of such substrates, however, such as its bottlenecks, strengths, and weaknesses, its "sweet-

spots", etc. An optimal configuration may solve a specific problem very well, but by itself, be uninteresting. Exploring the configuration space and abstract space of all reservoirs/programs could uncover more general characteristics of the system, and inform us more about what makes it compute and what is computable.

To map large complex spaces, typically with vast numbers of parameters (e.g., for physical substrates or deep/hierarchical structures) requires efficient search methods, rather than random or grid search. This is where a new class of *open*-ended *Quality Diversity* (QD) algorithms (Pugh et al. 2016) could have a significant impact, improving design, understanding, and exploration of new physical computing devices. These algorithms have experienced great success in the embedded setting of evolutionary robotics (Mouret and Doncieux 2009, 2012) and hard exploration problems (Ecoffet et al. 2019). These algorithms include novelty search with local competition (NSLC) (Lehman and Stanley 2011), and multi-dimensional archive of phenotypic elites (MAP-elites) (Mouret and Clune 2015).

The basic concept is to define a low-dimensional feature space, or *behaviour* space, separate from the high-dimensional parameter space, within which performance and behaviour are explored rather than optimised directly. (However, these algorithms can also be used to optimise directly, making them a superset of optimisation algorithms Mouret and Clune 2015.) During the search process, novel behaviours represent markers—points to seed or diverge from, or elites that represent the best behaviours in the low-dimensional search space, promoting the discovery of new solutions that would be otherwise too difficult to reach using standard optimisation algorithms.

These particular algorithms specialise in domains where the fitness landscape—the space of all solutions with respect to performance— is often deceptive and sparse. This deceptiveness, in evolutionary algorithm terms, typically relates to the *indirect* encoding between the genotype (encoding) and phenotype (physical instantiation), where information in the genome may affect many parts of the phenotype.

They overcome deceptive spaces in part by tuning the trade-off between exploration and exploitation, promoting diverse solutions as well as opimising local niches. As a by-product, the process produces a holistic view of the entire search space, rather than a single optimised solution. Mouret and Clune refer to these algorithms as a new class of "Illumination algorithms" (Mouret and Clune 2015), mapping the highest performing solutions in each region of the feature space and determining each region's overall potential.

In summary, the programming of single substrates for reservoir computing is sometimes simple, or complex. It is simple when there is a relatively good understanding of the substrate parameters, but considerably more complicated when little is known about parameters and resulting computational properties.

It is important to remember that a physical system's program encompasses encoding as well as representation, which may be altered. This in turn results in an enormous range of potential programs even for a single substrate.

As architectures move from single devices to multiple devices, programming will become increasingly challenging. To fully utilise such architectures, we first need to understand more about the computing capabilities of substrates. To do this, we have

mentioned one class of search algorithms that focuses on characterising the space of all reservoirs implemented by a single substrate rather than optimising parameters towards specific tasks.

5 What Makes a Good Physical Reservoir?

The ability to perform useful information processing is an almost universal characteristic of dynamical systems, provided a fading memory and linearly independent internal variables are present (Dambre et al. 2012). However, each dynamical system will tend to suit different tasks, with only some performing well across a range of tasks.

In terms of all reservoirs realisable by a substrate, many may perform well, given the selection of suitable parameters. Yet, many will also be unstable and unusable. This selection process may be easy for well-understood and constrained systems, but poses a serious challenge when less is known about the substrate and how it will react to stimulation, or different parameter ranges and encodings.

Until recently, no experimental framework has tried to explore, map, and compare the complete computational expressiveness—including encodings and representations—of physical and virtual substrates. That is, no practical method has been proposed to characterise the substrate and its individual instantiations to build a comprehensive view of the computing *quality* of substrates. A/R theory, Sect. 3.2, is a framework to *define* physical computing, but it does not provide guidance on how to determine an appropriate encoding, instantiation, or representation.

To tackle this non-trivial problem, we have developed the CHAracterisation of Reservoir Computers (CHARC) framework (Dale et al. 2019b). The framework describes characterisation phases and adaptable levels to measure *quality*, defined as the total capacity to realise distinct reservoirs in terms of different dynamical properties.

5.1 Framework Outline

To characterise a test substrate, two phases must be completed: quality assessment of a *reference* substrate (phase one), and characterisation of the *test* substrate (phase two). Phase one provides something to compare to and is typically carried out only once, provided a suitable reference is chosen. The basic structure and flow of each phase are shown in Fig. 3.

The first level is the *definition* level (Fig. 3, step 1). Here, the *behaviour space* of all abstract reservoirs is defined. This abstract space represents the dynamical behaviour of the substrate when configured. To create this n-dimensional space, n independent property measures are identified and used. Each point in this space is a behavioural representation of a substrate's configuration according to different measures.

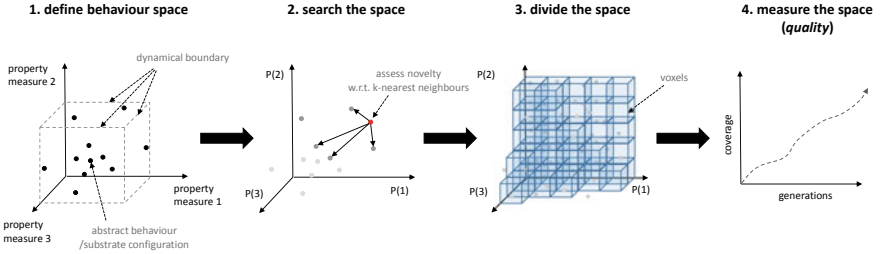


Fig. 3 CHARC framework workflow. This is typically carried out for both phases, using the same framework meta-parameters

More measures of distinct properties should result in greater model reliability and therefore greater confidence in the quality measure. However, when defining the behaviour space some properties are difficult, if not impossible, to measure across all substrates. Any measure used with the framework should represent the observed behaviour of the system, rather than specific properties related to its implementation, or measures unpractical/impossible to carry out with physical and/or black-box systems.

To demonstrate the basic framework structure, three measures are applied in Dale et al. (2019b) to define the behaviour space: Kernel Rank, Generalisation Rank, and Memory Capacity.

Kernel rank (KR) measures the reservoir's ability to produce a rich non-linear representation of the input. Generalisation rank (GR) is a measure of the reservoir's capability to generalise given similar input streams. More information about these measures can be found in Legenstein and Maass (2007). Low ranking values in both measures typically represent a system in an ordered regime, and both having high values equate to chaotic regimes. According to Büsing et al. (2010), the most interesting reservoirs are found to exhibit a high kernel rank and a low generalisation rank. However, in terms of matching reservoir dynamics to specific tasks, the right balance varies.

The measure for memory capacity (MC) in Dale et al. (2019b) captures the linear short-term memory capacity of a reservoir. The linear measure was first outlined in Jaeger (2001b) to quantify the echo state property. For the echo state property to hold, the dynamics of the input-driven reservoir must asymptotically wash out any information resulting from initial conditions. This property therefore implies a fading memory exists, characterised by the short-term memory capacity. As demonstrated in Dambre et al. (2012), other measures quantifying the non-linear, quadratic, and cross-memory capacities are possible, providing a more complete picture of memory in dynamical systems.

As explained in Dale et al. (2019b), the three selected measures are important, but by themselves do not capture every interesting dynamical property of dynamical systems. Therefore, more measures are still desired. For example, the total capacity

measures defined in Dambre et al. (2012) could replace, or add to, the axes of the behaviour space.

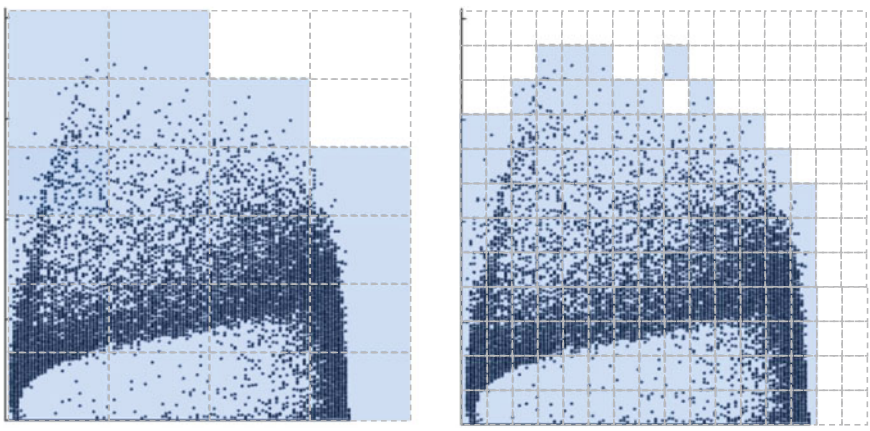
The *Exploration* level (Fig. 3, step 2) encompasses the search method and mapping process. At this level, the mapping between abstract reservoir and substrate configuration is explored.

Exploration is carried out in the behaviour space using an implementation of novelty search (Lehman and Stanley 2008). Novelty search, an open-ended genetic algorithm, navigates the behaviour space searching for novel solutions until some user-defined termination point, e.g., after so many evolutionary evaluations, or possibly when the rate of exploration has stalled.

Given enough time (search evaluations), the exploration process can outline the boundaries of the system dynamics, i.e., the boundary defining what behaviours are possible and not possible. In Dale et al. (2019b), this is demonstrated for a physical carbon nanotube composite. Due to the limited behavioural freedom of that substrate, its boundaries are relatively constrained, showing that any tasks requiring more than minimal memory requirements tend to be unsuited to the substrate. The results also demonstrate that exploration can be used to identify the practical use, if any, of the substrate, or whether the selected method of encoding, representation, and configuration is appropriate.

The final *Evaluation* level (Fig. 3, steps 3, 4) defines the mechanisms to evaluate quality. To measure quality, the behaviour space is divided into voxels/cells. Figure 4 demonstrates how the behaviour space may be divided and measured depending on the experimentally defined quality *resolution*.

The number and size of voxels/cells depend on the spaces being compared. For example, the voxel size can be large and coarse, grouping many local behaviours,



(a) Low resolution (b) High resolution

Fig. 4 Example of a 2-D behaviour space divided into different voxel sizes. Each dot represents a measured behaviour in a generic 2-D behaviour space. Squares represent 2-D voxels

signifying the extent of the region of behaviours discoverable (Fig. 4a), or an increase in resolution to provide a more fine-grained view of how the space is filled (Fig. 4b).

Overall, the total number of voxels/cells occupied forms the measure of quality. The quality value therefore represents an approximation of the system's dynamical freedom, or the substrate's capacity to instantiate different distinct reservoirs.

5.2 *Characterising Substrate Quality*

The initial evaluation and validation of the CHARC framework (Dale et al. 2019b) assess the quality of three very different dynamical systems: a recurrent neural network, a numerical simulation of a Mackey–Glass oscillator with a delay line, and a physical carbon nanotube-based substrate.

Artificial recurrent neural networks such as echo state networks (ESN) are considered state-of-the-art reservoir substrates, so ESNs of various sizes form the reference substrate. The simulated Mackey–Glass oscillator with a delay line (referred to as DR in this chapter) is a reservoir substrate known to perform remarkably well across different reservoir computing benchmarks (Appeltant et al. 2011; Paquot et al. 2012; Duport et al. 2012). The physical substrate comprising a mixed carbon nanotube–polymer (poly-butyl-methacrylate) composite (CNT) is known to perform well on several simple computational tasks (Mohid et al. 2016; Dale et al. 2016b, 2017), but struggles with some harder reservoir computing benchmarks (Dale et al. 2016a).

Results in Dale et al. (2019b) show the behavioural freedom of each substrate is significantly different when compared using a voxel size of $10 \times 10 \times 10$ behavioural units. The quality of each substrate is measured as the total number of voxels occupied after each experimental run of 2000 search generations. Figure 5 shows the number of voxels occupied as the exploration process progresses (every 200 generations), with error bars displaying the min-max values for 10 independent evolutionary runs.

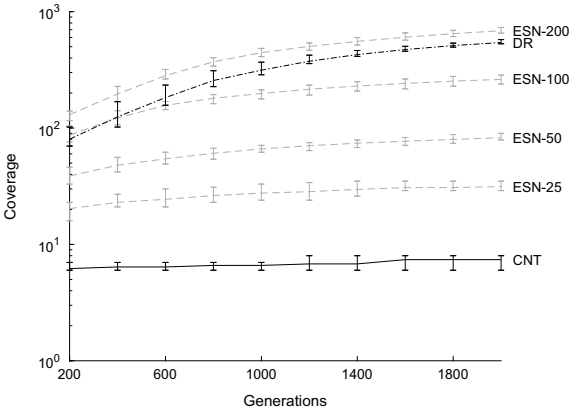
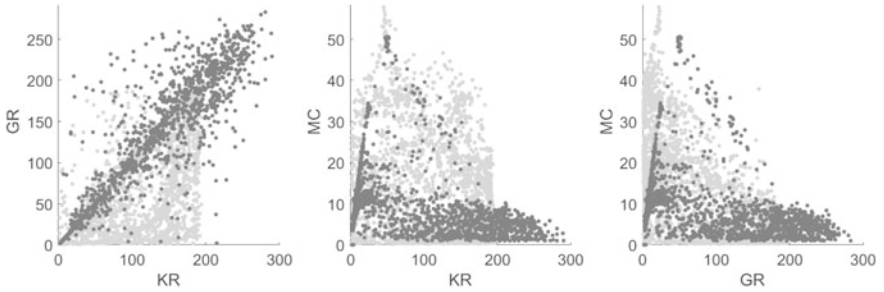
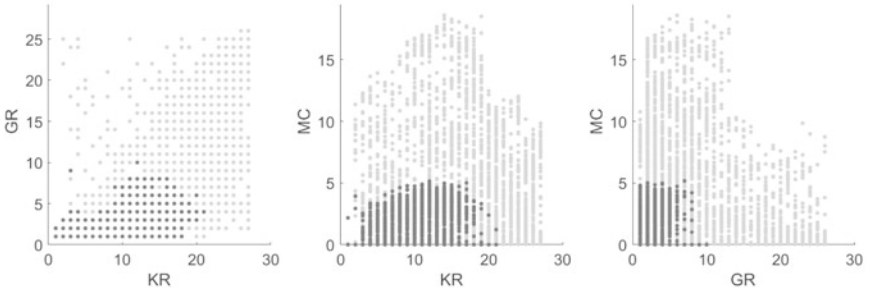


Fig. 5 Voxel measure of coverage as the number of search generations increases. Test substrates are shown as solid black lines (DR, CNT), and reference substrates (ESNs) are dashed grey lines. Error bars display the min-max values for all search runs. Note the logarithmic coverage scale



(a) 200 node ESN (light grey) with DR (dark grey)



(b) 25 node ESN (light grey) with CNT (dark grey)

Fig. 6 Behaviours discovered when exploring the ESN, CNT, and DR substrates. To visually compare substrates, each test substrate is plotted over the reference substrate with the most similar quality

The largest measured ESN networks (200 nodes) have the highest quality. The DR substrate (with 400 virtual nodes) is closely equivalent to the 200 node ESN substrate in terms of quality. The CNT substrate, on the other hand, had a considerably smaller quality than an ESN of just 25 nodes.

The quality value alone gives only a single numerical representation of quality. It is valuable to perform a visual inspection of the explored behaviour space to get a more nuanced view of a substrate's performance.

The discovered behaviours of the DR and CNT substrates are shown in Fig. 6. In the plots, the behaviours for each test substrate are presented in the foreground and the reference substrate (ESN) with the most similar quality (200 node ESN in Fig. 6a and 25 node ESN in Fig. 6b) in the background.

The DR behaviours (Fig. 6a) extend into regions that the 200 node ESN cannot reach, resulting in what appears to be fewer occupied regions than the ESN. However, this does not imply that such regions cannot be occupied. Given more search generations, these regions would likely be filled, as similar behaviours are already discovered.

The CNT substrate (Fig. fig:bsofsubstratesb) struggles to exhibit enough (stable) internal activity to create a strong non-linear projection, and to effectively store recent

input and state information, agreeing with previous results (Dale et al. 2016a, b, 2017). This suggests why only a limited range of tasks are suitable for the substrate, and why small ESNs tend to be good models of the substrate.

Overall, the results of the CNT substrate show that, using its current encoding and representation, only a limited set of behaviours is possible. In future work, quality might be improved using other types of input stimulus and control signals, or reading/combining other electrical output signals.

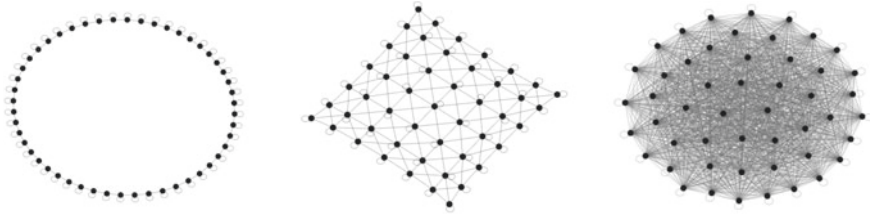
5.3 Using Quality to Assess Substrate Design

In this section, we demonstrate how the CHARC framework can be used to assess, and potentially inform, the design of future reservoir substrates.

In Rodan and Tiňo (2010) and Rodan and Tino (2011), it is shown that simple and deterministic connection topologies tend to perform as well as, or better than, standard (fully connected) randomly generated reservoir networks on a number of benchmark tasks. If the design of state-of-the-art reservoirs requires only simple structures and topologies, physical reservoirs become much easier to design and implement.

To investigate what effect structure and connection complexity have on dynamical behaviour, CHARC has been applied to neural graphs of varying complexities and sizes (Dale et al. 2019a). The effect of network topology and connection complexity (in this case, *directed* and *undirected* graphs) is investigated by evaluating three simulated recurrent network topologies: *ring*, *lattice*, and *fully connected* networks.

The ring topology (Fig. 7a), with the least complexity, has nodes with a single self-connection and one connection to each of its direct neighbours. The basic ring topology is the simplest network to implement in physical hardware as the number of connections required is small. Ring structures have already been applied to many reservoir computing systems, including minimum complexity echo state networks (ESN) (Rodan and Tino 2011), DNA reservoirs (deoxyribozyme oscillators) (Goudarzi et al. 2013), Cycle reservoirs with regular jumps (CRJ) (Rodan and



(a) ring (b) lattice (c) fully-connected

Fig. 7 Network structures investigated in Dale et al. (2019a)

Tiňo 2010), and delay-based reservoirs using a single non-linear node with a delay line (Appeltant et al. 2011; Paquot et al. 2012).

The lattice topology (Fig. 7b) has greater connection complexity. In Dale et al. (2019a), a square grid of neurons with each connected to its Moore neighbourhood is defined. In this configuration, each node (except for the perimeter nodes) has eight connections to neighbours and one self-connection, resulting in a maximum of nine connections per node. Lattice networks/models are common in physics. Examples include discrete lattices like the Ising model with variables representing magnetic dipole moments of atomic spins, and the Gray–Scott reaction–diffusion model to simulate chemical systems (Pearson 1993). Physical substrates often have a regular grid of connections. Lattice networks are therefore more realistic representations of many physical systems that would be considered for reservoir computing.

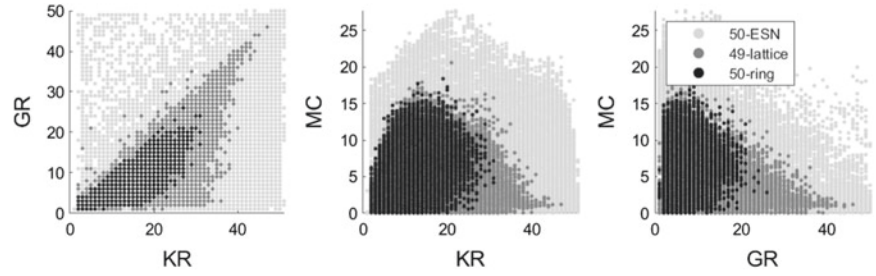
The fully connected topology (Fig. 7c) has the most connections and is considered the most complex. This type of network is challenging to implement in physical hardware.

The results in Dale et al. (2019a) show that a directed and fully connected topology (the ESN network) occupies a greater area of the behaviour space, possessing a higher quality than the others, independent of size. The other topologies struggle to reach similar levels of coverage. The behavioural limits of these simpler networks appear to have been reached.

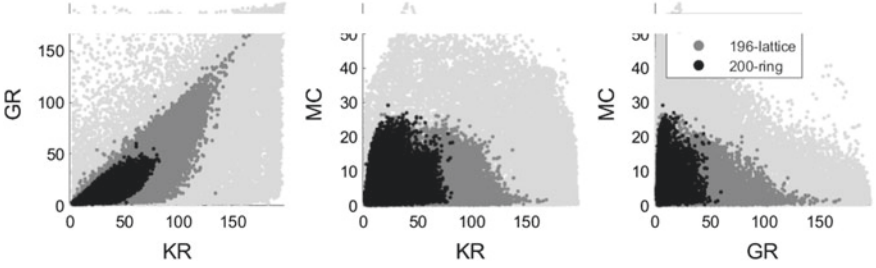
A common pattern found across all topologies and connection types is that quality increases with network size. Simpler structures can produce similar quality to more complex networks simply by increasing network size. A visualisation of how two network sizes (50 and 200-neuron) cover the behaviour space using each network topology is given in Fig. 8. ESNs tend to occupy regions the others cannot, such as chaotic regions (both high KR and high GR), and regions with larger memory capacities. The ring and the lattice topologies appear to have similar maximum memory capacities; however, lattices typically exhibit greater non-linearity and chaotic behaviour (higher KR and GR values) than rings.

Directed and undirected connection types are also assessed in Dale et al. (2019a). Connection type is equally as important as network topology. The coverage of a 100-neuron ring and lattice is shown in Fig. 9. Each plot shows the directed connection type (grey) and undirected type (black). Directed connections typically result in broader dynamical behaviour, producing more "challenging" behaviours (high KR and high MC). The difficulty in producing such behaviours exists because non-linearity (KR) and ordered dynamics (MC) are often conflicting.

Adding more accessible parameters (i.e., more connections) does not always lead to more dynamical behaviours. Networks with fewer free parameters (e.g., directed rings) could still produce similar qualities to more structurally complex networks (e.g., undirected lattices). How weights are structured and *directed* has a much greater effect on the quality of the network (Dale et al. 2019a), supporting similar results testing different hierarchical structures for reservoir computing (Dale 2018a).



(a) 50-neuron



(b) 200-neuron

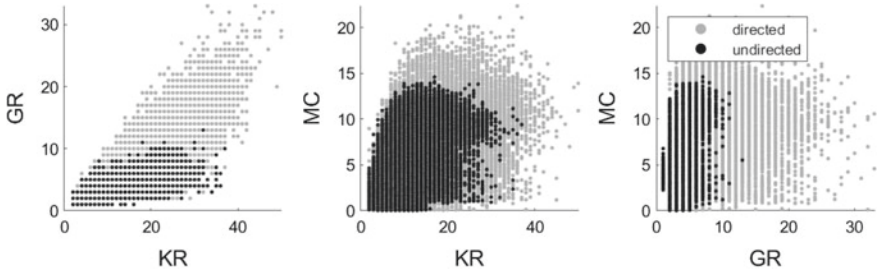
Fig. 8 Topology effects. Superimposed behaviour space of **a** 50-neuron network, **b** 200 neuron network. Showing 10 runs of 2000 generations each. Only the directed topologies are shown. From Dale et al. (2019a)

5.4 CHARC Conclusions

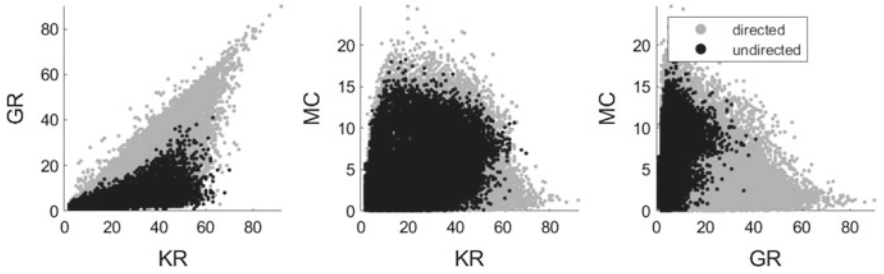
The CHARC framework offers techniques to address several significant challenges outlined for reservoir computing (Goudarzi and Teuscher 2016). It provides (i) a framework to describe the computational expressiveness and complexity of substrates as a function of parameters and behaviours, (ii) a means to assess whether measures better represent the qualities/properties of the system, and (iii) a measure to assess the limits of physical RC substrates.

As shown in Dale et al. (2019a, b), the framework has diverse applications. For example, it can be used to evaluate and compare different substrates and design choices, such as the role of structure in RC systems, without needing extensive testing on tasks (see Sect. 5.3). In Dale et al. (2019b), the framework is also used to model the non-trivial relationships between reservoir properties and performance. The model can reliably predict task performance based on behaviour, across different substrates, without the need to evaluate tasks directly on the substrate.

The framework could be used in evaluating and determining the benefits of physical RC implementations against non-physical ones, a question raised in Goudarzi and Teuscher (2016). The framework could be used to improve substrate design, or extended to other models of computation (Stepney 2019; Dale et al. 2020). The



(a) 100-neuron ring



(b) 100-neuron lattice

Fig. 9 Directed versus undirected topology effects. Superimposed behaviour space of 100-neuron ring and lattice network. Showing 10 runs of 2000 generations each. From Dale et al. (2019a)

flexibility of the framework allows for iterative improvements and expansion, for example, extending it to evaluate hybrid models using heterotic computing. With hybrid models, the behaviour space may be used as a repertoire of functional primitives, or modules, to build or program complex architectures, such as hierarchical reservoirs (Bürger et al. 2015; Dale 2018a; Gallicchio et al. 2017).

6 Conclusion

In a field that continues to diversify and excel, basic questions and theory in reservoir computing, and computing with physical systems, are being left behind. The rapid shift in recent years from virtual reservoir systems to physical ones has resulted in fundamental theory struggling to keep up with practice, with experiments highlighting the lack in theory, rather than theory proposing experiments.

In this chapter, we have highlighted areas of fundamental theory still undeveloped in computing with physical systems, and challenges in building unconventional physical [reservoir] systems. We have discussed the importance of appropriate encodings and representation, e.g., whether the computational model naturally fits

the computational substrate; programming and parameter selection, including thinking beyond single-task optimisation. And we have described a recent framework that provides some useful techniques to understand physical and reservoir systems, including a method to assess and compare the expressiveness of computing substrates, an exploratory method to configure systems when little is known about internal workings, a means to understand how substrate design choices affect behavioural/reservoir range, and how the abstract behaviour space maps to the physical space.

At present, reservoir computing is not a complete model for physical computing, however, it does have fundamental properties that make it simple, versatile, and highly efficient. Physical RC is therefore a *stepping stone* towards a more mature, principled general methodology for discovering and exploiting good natural computational models for material and physical computing.

Acknowledgements We thank the anonymous referees for their constructive comments, which helped improve this chapter. This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) SpInspired project [grant number EP/R032823/1].

References

B. Abubakar, I. Idris, I. Rosdiazli, M.S. Sadiq, Applications of metaheuristics in reservoir computing techniques: a review. IEEE Access **6**, 58012–58029 (2018)

A. Adamatzky (ed.), *Advances in Unconventional Computing: Volume 1: Theory* (Springer, 2016a)

A. Adamatzky (ed.), *Advances in Unconventional Computing: Volume 2 Prototypes, Models and Algorithms* (Springer, 2016b)

L. Appeltant, M.C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C.R. Mirasso, I. Fischer, Information processing using a single dynamical node as complex system. Nat. Commun. **2**, 468 (2011)

L. Appeltant, G. Van der Sande, J. Danckaert, I. Fischer, Constructing optimized binary masks for reservoir computing with delay systems. Sci. Rep. **4**, 3629 (2014)

S. Basterrech, E. Alba, V. Snášel, An experimental analysis of the echo state network initialization using the particle swarm optimization, in *Sixth World Congress on Nature and Biologically Inspired Computing (NaBIC 2014)* (IEEE, 2014), pp. 214–219

H. Broersma, J.F. Miller, S. Nichele, Computational matter: evolving computational functions in nanoscale materials, in *Advances in Unconventional Computing* (Springer, 2017), pp. 397–428

D. Brunner, M.C. Soriano, C.R. Mirasso, I. Fischer, Parallel photonic information processing at gigabyte per second data rates using transient states. Nat. Commun. **4**, 1364 (2013)

J. Bürger, A. Goudarzi, D. Stefanovic, C. Teuscher, Hierarchical composition of memristive networks for real-time computing, in *Proceedings of the 2015 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)* (IEEE, 2015), pp. 33–38

L. Büsing, B. Schrauwen, R. Legenstein, Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons. Neural Comput. **22**(5), 1272–1311 (2010)

F. Caravelli, J. Carbajal, Memristors for the curious outsiders. Technologies **6**(4), 118 (2018)

P. Cariani, To evolve an ear. Epistemological implications of Gordon Pask's electrochemical devices. Syst. Res. **10**(3), 19–33 (1993)

K.C. Chatzidimitriou, P.A. Mitkas, A NEAT way for evolving echo state networks, in *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)* (IOS Press, 2010), pp. 909–914

M. Dale, Unconventional reservoir computers: exploiting materials to perform computation, in *Eighth York Doctoral Symposium on Computer Science & Electronics* (2015), p. 69
M. Dale, Neuroevolution of hierarchical reservoir computers, in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2018)* (ACM, 2018a), pp. 410–417
M. Dale, Reservoir computing in materio. PhD thesis, University of York (2018b)
M. Dale, J.F. Miller, S. Stepney, M.A. Trefzer, Evolving carbon nanotube reservoir computers, in *International Conference on Unconventional Computation and Natural Computation (UCNC 2016)* (Springer, 2016a), pp. 49–61
M. Dale, J.F. Miller, S. Stepney, M.A. Trefzer, Reservoir computing in materio: an evaluation of configuration through evolution, in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)* (2016b), pp. 1–8
M. Dale, J.F. Miller, S. Stepney, M.A. Trefzer, Reservoir computing in materio: a computational framework for in materio computing, in *International Joint Conference on Neural Networks (IJCNN 2017)* (2017), pp. 2178–2185
M. Dale, J. Dewhirst, S. O'Keefe, A. Sebald, S. Stepney, M.A. Trefzer, The role of structure and complexity on reservoir computing quality, in *International Conference on Unconventional Computation and Natural Computation (UCNC 2019)*. LNCS, vol. 11493 (Springer, 2019a)
M. Dale, J.F. Miller, S. Stepney, M.A. Trefzer, A substrate-independent framework to characterise reservoir computers. Proc. R. Soc. A **475**(2226), 20180723 (2019b)
M. Dale, S. Stepney, M. Trefzer, Designing computational substrates using open-ended evolution, in *Artificial Life Conference Proceedings* (MIT Press, 2020), pp. 665–667
J. Dambre, D. Verstraeten, B. Schrauwen, S. Massar, Information processing capacity of dynamical systems. Sci. Rep. **2**, 514 (2012)
F. Duport, B. Schneider, A. Smerieri, M. Haelterman, S. Massar, All-optical reservoir computing. Opt. Express **20**(20), 22783–22795 (2012)
A. Ecoffet, J. Huizinga, J. Lehman, K.O. Stanley, J. Clune, Go-explore: a new approach for hard-exploration problems (2019), arXiv:1901.10995
A.A. Ferreira, T.B. Ludermir, Comparing evolutionary methods for reservoir computing pre-training, in *International Joint Conference on Neural Networks (IJCNN 2011)* (IEEE, 2011), pp. 283–290
C. Gallicchio, A. Micheli, L. Pedrelli, Deep reservoir computing: a critical experimental analysis. Neurocomputing **268**, 87–99 (2017)
A. Goudarzi, C. Teuscher, Reservoir computing: Quo vadis? in *Proceedings of the 3rd ACM International Conference on Nanoscale Computing and Communication* (ACM, 2016), p. 13
A. Goudarzi, M.R. Lakin, D. Stefanovic, DNA reservoir computing: A novel molecular computing approach, in *DNA Computing and Molecular Programming (DNA 2013)* (Springer, 2013), pp. 76–89
S. Harding, J.F. Miller, Evolution in materio: Initial experiments with liquid crystal, in *2004 NASA/DoD Conference on Evolvable Hardware* (IEEE, 2004), pp. 298–305
C. Horsman, S. Stepney, R.C. Wagner, V. Kendon, When does a physical system compute? Proc. R. Soc. A **470**(2169), 20140182 (2014)
D.C. Horsman, Abstraction/representation theory for heterotic physical computing. Philos. Trans. R. Soc. A **373**(2046), 20140224 (2015)
D. Horsman, S. Stepney, V. Kendon, The natural science of computation. Commun. ACM **60**, 31–34 (2017)
D. Horsman, V. Kendon, S. Stepney, Abstraction/representation theory and the natural science of computation, in *Physical Perspectives on Computation, Computational Perspectives on Physics*, ed. by M.E. Cuffaro, S.C. Fletcher (Cambridge University Press, Cambridge, 2018), pp. 127–149
H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. GMD Technical Report **148**, 34. German National Research Center for Information Technology, Bonn, Germany (2001a)
H. Jaeger, Short term memory in echo state networks. GMD-Forschungszentrum Informationstechnik (2001b)

F. Jiang, H. Berry, M. Schoenauer, Supervised and evolutionary learning of echo state networks, in *Parallel Problem Solving from Nature (PPSN X)* (Springer, 2008), pp. 215–224
V. Kendon, A. Sebald, S. Stepney, M. Bechmann, P. Hines, R.C. Wagner, Heterotic computing, in *International Conference on Unconventional Computation (UCNC 2011)* (Springer, 2011), pp. 113–124
V. Kendon, A. Sebald, S. Stepney, Heterotic computing: past, present and future. Philos. Trans. R. Soc. A: Math. Phys. Eng. Sci. **373**(2046), 20140225 (2015)
Z. Konkoli, S. Nichele, M. Dale, S. Stepney, Reservoir computing with computational matter, in [68] (2018), pp. 269–293
A.F. Krause, V. Dürr, B. Bläsing, T. Schack, Multiobjective optimization of echo state networks for multiple motor pattern learning, in *18th IEEE Workshop on Nonlinear Dynamics of Electronic Systems (NDES 2010)* (IEEE, 2010)
R. Legenstein, W. Maass, Edge of chaos and prediction of computational performance for neural circuit models. Neural Netw. **20**(3), 323–334 (2007)
Lehman, K.O. Stanley, Exploiting open-endedness to solve problems through the search for novelty, in *ALife XI* (2008), pp 329–336
J. Lehman, K.O. Stanley, Evolving a diversity of virtual creatures through novelty search and local competition, in *Proceedings of the 13th annual conference on Genetic and Evolutionary Computation (GECCO 2011)* (ACM, 2011), pp. 211–218
S. Lloyd, Ultimate physical limits to computation. Nature **406**(6799), 1047 (2000)
W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. **14**(11), 2531–2560 (2002)
M. Massey, A. Kotsialos, D. Volpati, E. Vissol-Gaudin, C. Pearson, L. Bowen, B. Obara, D. Zeze, C. Groves, M. Petty, Evolution of electronic circuits using carbon nanotube composites. Sci. Rep. **6**, 32197 (2016)
F. Matzner, Neuroevolution on the edge of chaos, in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2017)* (ACM, 2017), pp. 465–472
J.F. Miller, K. Downing, Evolution in materio: Looking beyond the silicon box, in *NASA/DoD Conference on Evolvable Hardware 2002* (IEEE, 2002), pp. 167–176
J.W. Mills, Polymer processors. Technical report TR580, Department of Computer Science, University of Indiana (1995)
J.W. Mills, The nature of the extended analog computer. Phys. D **237**(9), 1235–1256 (2008)
M. Mohid, J.F. Miller, S.L. Harding, G. Tufte, M.K. Massey, M.C. Petty, Evolution-in-materio: solving computational problems using carbon nanotube-polymer composites. Soft. Comput. **20**(8), 3007–3022 (2016)
J.B. Mouret, J. Clune, Illuminating search spaces by mapping elites (2015), arXiv:1504.04909
J.B. Mouret, S. Doncieux, Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity, in *IEEE Congress on Evolutionary Computation (CEC 2009)* (IEEE, 2009), pp. 1161–1168
J.B. Mouret, S. Doncieux, Encouraging behavioral diversity in evolutionary robotics: an empirical study. Evol. Comput. **20**(1), 91–133 (2012)
Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, S. Massar, Optoelectronic reservoir computing. Sci. Rep. **2**, 287 (2012)
G. Pask, Physical analogues to the growth of a concept, in *Mechanisation of Thought Processes, National Physical Laboratory Symposium 10, HMSO*, vol. II (1959), pp. 877–922
J.E. Pearson, Complex patterns in a simple system. Science **261**(5118), 189–192 (1993)
J.K. Pugh, L.B. Soros, K.O. Stanley, Quality diversity: a new frontier for evolutionary computation. Front. Robot. AI **3**, 40 (2016)
J. Qiao, F. Li, H. Han, W. Li, Growing echo-state network with multiple subreservoirs. IEEE Trans. Neural Netw. Learn. Syst. **28**(2), 391–404 (2017)
A. Rodan, P. Tiňo, Simple deterministically constructed recurrent neural networks, in *International Conference on Intelligent Data Engineering and Automated Learning* (Springer, 2010), pp. 267–274

A. Rodan, P. Tino, Minimum complexity echo state network. IEEE Trans. Neural Netw. **22**(1), 131–144 (2011)
L.A. Rubel, The extended analog computer. Adv. Appl. Math. **14**(1), 39–50 (1993)
A.T. Sergio, T.B. Ludermir, PSO for reservoir computing optimization, in *International Conference on Artificial Neural Networks* (Springer, 2012), pp. 685–692
S. Stepney, Embodiment, in *In Silico Immunology*, ed. by D. Flower, J. Timmis (Springer, 2007), pp. 265–288
S. Stepney, The neglected pillar of material computation. Phys. D **237**(9), 1157–1164 (2008)
S. Stepney, Co-designing the computational model and the computing substrate, in *International Conference on Unconventional Computation and Natural Computation (UCNC 2019)*. LNCS, vol. 11493 (Springer, 2019)
S. Stepney, S. Rasmussen, M. Amos (eds.), *Computational Matter* (Springer, 2018)
G. Tanaka, T. Yamane, J.B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, A. Hirose, Recent advances in physical reservoir computing: a review. Neural Netw. **115**, 100–123 (2019)
A. Thompson, An evolved circuit, intrinsic in silicon, entwined with physics, in *Evolvable Systems: From Biology to Hardware (ICES 1996)* (Springer, 1997), pp. 390–405
A.M. Turing, Intelligent machinery, in *Machine Intelligence*, vol. 5, ed. by B. Meltzer, D. Michie (Edinburgh University Press, 1969), pp. 3–23 (published after the author's death)
J. Yperman, T. Becker, Bayesian optimization of hyper-parameters in reservoir computing (2016), arXiv:1611.05193
S. Yuenyong, On the gradient-based sequential tuning of the echo state network reservoir parameters, in *Pacific Rim International Conference on Artificial Intelligence* (Springer, 2016), pp 651–660

Part III
Physical Implementations: Mechanics and Bio-inspired Machines

Physical Reservoir Computing in Robotics

Helmut Hauser

Abstract In recent years, there has been an increasing interest in using the concept of physical reservoir computing in robotics. The idea is to employ the robot's body and its dynamics as a computational resource. On one hand, this has been driven by the introduction of mathematical frameworks showing how complex mechanical structures can be used to build reservoirs. On the other hand, with the recent advances in smart materials, novel additive manufacturing techniques, and the corresponding rise of soft robotics, a new and much richer set of tools for designing and building robots is now available. Despite the increased interest, however, there is still a wide range of unanswered research questions and a rich area of under-explored applications. We will discuss the current state of the art, the implications of using robot bodies as reservoirs, and the great potential and future directions of physical reservoir computing in robotics.

1 Introduction

Reservoir computing as a term has been coined by Schrauwen et al. (2007). Their fundamental insight was that the underlying principles of Echo State Networks (ESN) (Jaeger and Haas 2004) and Liquid State Machines (LSM) (Maass et al. 2002), which have been developed independently from each other, are the same. Both use complex, nonlinear dynamical systems (referred to as *reservoirs*) as a computational resource. While LSMs, inspired by the structure of the brain, are using differential equations describing neurons, ESNs are employing simple, but abstract nodes in form of nonlinear differential equations to achieve the same results. Both use simple, dynamic building blocks to construct a high-dimensional, stable, but nonlinear dynamical system, i.e. the *reservoir*.

H. Hauser (✉)
Department of Engineering Mathematics, University of Bristol, Woodland Road, Bristol BS8 1UB, UK
e-mail: helmut.hauser@bristol.ac.uk

Bristol Robotics Lab, Coldharbour Ln, Bristol BS16 1QY, UK

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_8

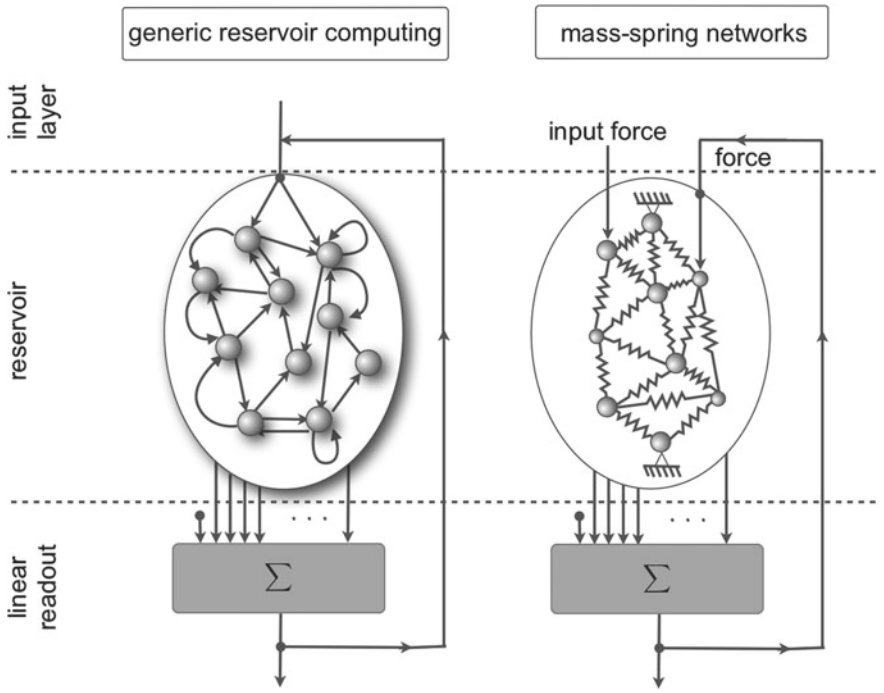


Fig. 1 Comparison of conventional reservoir computing (example is with ESNs nodes) with physical reservoir computing using mass–spring–damper systems as suggested by Hauser et al. (2011). Figure adapted from Hauser et al. (2014)

The role of a reservoir can be described as a *kernel* in the machine learning sense for support vector machines, see, e.g. Vapnik (1998). Low-dimensional input signals (typically in form of time series) are fed into the reservoir and projected nonlinearly into its high-dimensional state space. It is well established that this boosts the computational power of anything that comes after it. For a discussion in more detail in the context of reservoir computing, we refer to Hauser et al. (2011, 2012, 2014). In addition, since a reservoir is built up by smaller dynamical systems (i.e. differential equations), it also integrates information over time. Consequently, loosely speaking, besides the kernel property, the reservoir exhibits also some form of (fading) memory. This leads us to an interesting consequence. Since the reservoir is already able to integrate information and carry out nonlinear mappings, it suffices to add a simple linear, static readout to obtain a powerful computational device. We only have to find a set of static, linear weights to combine the signals from the high-dimensional state space of the reservoir to achieve the desired output (compare Fig. 1).

The resulting setup, i.e. reservoir plus linear readout, is a surprisingly powerful computational device. It can be used to approximate in principle any computation that

can be represented by a Volterra series. In practice, this means we can emulate any smooth (i.e. sufficiently derivable) dynamical system with one equilibrium point.[1]

The remarkable part of this setup however is how we achieve learning. We don't change any parameters of the reservoir (typically it is randomly initialized within some parameter bounds), but we only have to adapt/find the optimal readout weights. Since they are a set of fixed (static) values used to linearly combine the states of the reservoir, we can employ simple linear regression. Note that this means that we can learn to emulate nonlinear dynamical systems with such a setup by using simple, linear regression. One could say the integration and nonlinear combination of information (which are both needed in a nonlinear, dynamical system) are outsourced to the reservoir or the reservoir is exploited as a computational resource.

From a machine learning perspective, reservoir computing is an alternative approach to Recurrent Neural Networks (RNN), which are typically employed to approximate dynamical systems. However, it is well known that the learning process for RNNs, i.e. to find the optimal connectivity weights, is tedious. There exist various approaches, e.g. Backpropagation Through Time (BPTT) and its variations, but they are all prone to get stuck in a local minimum or are known to be slow in their convergence. On the other hand, in reservoir computing, we only have to find a linear, static set of output weights by employing linear regression, which by definition finds always the globally optimal solution[2] and is very fast.

The step from *standard* reservoir computing as described above to *physical* reservoir computing is straightforward when we consider which properties a reservoir has to have to be computationally useful. As pointed out before, the reservoir has (i) the role of mapping nonlinearly low-dimensional inputs into a higher state space and (ii) to integrate information over time. Interestingly, these properties are rather abstract and, as a consequence, a wide range of dynamical systems can be used as a reservoir. They only have to exhibit nonlinear dynamics with a fading memory property (i.e. being exponentially stable with one equilibrium point) and a high-dimensional state space.

For example, in the case of LSMs, a network of spiking neuron models is used to construct such a system, while in ESNs, randomly connected structures of simple nonlinear differential equations are employed. This means any nonlinear, exponentially stable (with one equilibrium point), high-dimensional dynamical system can be potentially used as reservoirs, which naturally includes also real physical systems.

The earliest work demonstrating that this is possible was done by Fernando and Sojakka (2003). They used the water surface in a bucket as a reservoir to carry out vowel classification. The input was sound waves exciting the water and the readout was carried out through the pixelated version of video recordings of the water surface. Since then, a wide range of nonlinear physical systems have been proposed and shown to be useful as reservoirs. Examples include nonlinear effects in lasers

[1] The relation between Volterra series and dynamical systems is discussed in Boyd (1985).

[2] Linear regression minimizes the quadratic error against the target. Since the error landscape is quadratic, there is only one (global) minimum. Note that this doesn't necessarily mean that reservoir computing setups are always performing better than RNN networks.

(Smerieri et al. 2012), body dynamics of soft robots (Nakajima et al. 2018a), and even quantum-mechanical effects (Fujii and Nakajima 2017). Here, in this chapter, we will concentrate on applications in robots.

In the next section, we will re-introduce the underlying theoretical models that have contributed to a better understanding of how the dynamics of robot bodies can be used as reservoirs. In Sect. 3, we discuss the state of the art and give an overview of existing work in this area, which is followed by a critical assessment of the advantages (Sect. 4) and limitations (Sect. 5) of the approach. In Sect. 6, we will highlight the close connection of physical reservoir computing and the recently emerged research field of soft robotics. Finally, in Sect. 7, we discuss the future of physical reservoir computing in robotics and the great potential it holds.

2 Theoretical Models

Physical reservoir computing in robotics has gained a lot of traction recently due to the introduction of mathematical models proving that certain mechanical structures can serve as reservoirs. Particularly, two models proposed by Hauser et al. (2011, 2012) were able to demonstrate that complex, mechanical architectures can serve as powerful reservoirs. They are composed of nonlinear mass–spring–damper systems which are connected to form a mechanical network (see Fig. 1 on the right). The motivation to use this approach was that these structures serve as good models to describe the body of biological systems and soft-bodied robots. Both exhibit rather complex and nonlinear dynamics that can be exploited in a physical reservoir computing setup.[3]

The proposed mathematical frameworks for physical reservoir computing based on mass–spring–damper systems are an extension of the original work by Maass et al. (2002, 2007) which show that LSMs can indeed carry out computations. While they prove the concept using models of spiking neurons and neural connection to construct a computationally useful structure (i.e. a reservoir), Hauser et al. demonstrated the same is possible with mechanical structures.

As with the original work, there has been a distinction between the two main reservoir computing setups. The so-called *feedforward setup*[4] maps an input directly (through the reservoir and readout) to the desired output, while the *feedback setup* also includes feedback loop(s) from the output back into the reservoir. Note that Fig. 1 shows the setup with explicit feedback loops. We will discuss both approaches now in more detail.

[3] Note that this also means that one could use biological bodies directly as reservoirs as well. One just has to find a way to read out the (at least partial) dynamic state of the system.

[4] Note that the term *feedforward* highlights here the fact that there are no explicit feedback loops from the output back into the reservoir. However, the reservoir naturally will still have stable feedback loops inside the reservoir to achieve the fading memory that is a prerequisite for reservoir computing.

2.1 Feedforward Setup

As mentioned before, the mathematical proof for the feedforward setup has been given by Hauser et al. (2011) and is based on the work by Maass et al. (2002). Both are based on a seminal paper by Boyd and Chua (1985), which explored the idea to emulate arbitrary Volterra series. Note that a Volterra series is an elegant way to approximate nonlinear, differential equations that have one equilibrium point.[5]

Boyd and Chua showed that any Volterra series can be approximated by a two-stage process, where the first stage has to be dynamic (include memory), but can be linear, and a second stage that has to be nonlinear, but can be static. The remarkable part is that these two stages can be implemented in any way as long as they fulfil certain properties. Specifically, the first stage has to exhibit the property of fading memory and there has to exist a pool of subsystems that is rich enough so that we can guarantee that we can always find two subsystems to separate any possible input signals (through filtering).[6] The second stage needs to be a universal function approximator, i.e. a system that is able to approximate any smooth enough nonlinear function with arbitrary precision. As one can see, these properties are very general and there are a lot of potential candidates that can embody them. Boyd and Chua gave a number of examples of possible implementations in their original work, mostly from the field of electrical circuits.

To appreciate the importance of their result, it's crucial to understand the concept of a Volterra series. Loosely speaking,[7] this could be understood as a Taylor series expansion that includes time. Instead of only approximating smooth, nonlinear functions, a Volterra series can approximate nonlinear dynamical systems. In the context of reservoir computing, especially physical reservoir computing, this gives us an elegant way to describe analogue computation. Instead of having to rely on a digital interpretation and the Turing machine concept, dynamical systems are a powerful way to describe analogue computational functions in the context of robotics. They can be used to express, for example, complex sensor functions (e.g. filtering) as well nonlinear controllers.

The only[8] limitation of using Volterra series as descriptors is they can only approximate nonlinear dynamical systems that have only one exponentially stable equilibrium point (or approximate only in the neighbourhood of an equilibrium point). However, we have to emphasize that this is still a very powerful and rich set of possible computations that can be represented. It includes nonlinear mappings that use memory. This means the output will not simply depend on the current input (that would be a simple function and could be approximated with a standard Arti-

[5] Or they approximate only the close neighbourhood of one equilibrium point, i.e. local approximation. Note that this is different from linear approximation approaches, e.g. expressed through a Jacobian matrix, since a Volterra series is still capturing nonlinearities.

[6] For example, we can achieve separation through different integration time constants.

[7] For more rigorous and mathematically sound descriptions, please refer to Boyd and Chua (1985).

[8] Note that there are more mathematical details, but for the sake of simplicity, we discuss only the main points. For a more detailed discussion, we refer to Boyd (1985) and Boyd and Chua (1985).

ficial Neural Network), but also on the history of input values, which would need a Recurrent Neural Network (RNN) to approximate it.

The question is now how can we use morphological structures, specifically mass–spring–damper systems, to approximate the Volterra series. The answer can be directly derived from Boyd and Chua's work. They laid out clearly the required properties for potential building blocks to construct the first stage of the process. They have to be (exponentially) stable and be able to separate signals as discussed before. The proof for linear mass–spring–damper systems to have these properties is straightforward (for more details, see Hauser et al. 2011).

For the second stage, the requirement is that it should be made up of a universal function approximator. We can use ANNs, which have exactly this property (see Hornik et al. 1989 for proof). Note that this two-stage setup does not yet resemble a standard reservoir setup, since the readout is still nonlinear and the first stage consists of a set of parallel, independent systems. Nevertheless, one can already say that at least the "memory part" (integration of information) is outsourced to stage one, i.e. the morphological part. However, a set of parallel mass–spring–damper system doesn't look very much like a common robot design,[9] nor does it describe well a biological body.

To get to a reservoir computing setup, we have to push Boyd and Chua's approach to the extreme by considering to outsource nonlinearity also to stage one, leaving a simplified readout that can be static and linear. This leads to networks like the one shown on the right side of Fig. 1. Note that this push "breaks" Boyd and Chua's proof and, so far, there is no mathematical proof for the full network structure (also not in the original work by Maass et al.). However, the step from the two-stage process to the full structure is reasonable and simulation results (as well real-world examples in the case of mechanical structures) demonstrate that it works.

As previously pointed out, the only limitation is that the feedforward setup is restricted to systems that can be represented by a Volterra series. Since this is already a very rich class of possible mappings, the question arises, which systems that might be interesting in robotics don't fall into this category? This will be answered in the next section on feedback setups.

2.2 Feedback Setup

Boyd already hinted at this limitation in his Ph.D. thesis (Boyd 1985). He made a link between the Volterra series and nonlinear dynamical systems. He laid out that the Volterra series can only approximate systems that have only one exponentially stable equilibrium point. Keeping the discussion in the realm of dynamical systems, it is then quite straightforward which dynamical systems fall outside of this definition, but would be still interesting in the context of robotics. First, we can think about multiple

[9] Note that, interestingly, Shim and Husbands (2007) suggested a setup which looks very similar and is used for a feathered flyer and which predates Hauser et al.'s work.

equilibrium points. Depending on the situation/task, different endpoints might be of interest. Another type of dynamical systems that go beyond exponentially stable ones are (nonlinear) limit cycles. These are particularly interesting in the context of robotics as limit cycles are a way to encode repetitive movements, e.g. in locomotion. In addition, other nonlinear effects like bifurcation go beyond a Volterra description as well. This could be potentially helpful if we want to control a qualitative change in the behaviour. For example, the robot could switch smoothly from a locomotion movement of its front leg (i.e. limit cycle) to a reaching movement (i.e. equilibrium point)—all described by one set of differential equations.

Interestingly, it is quite simple to overcome the limitations of the feedforward setup in reservoir computing. The solution is to counteract the energy loss in the reservoir. We simply have to add (linear) feedback loops[10] from the output of the setup. Again, in standard reservoir computing, these signals are abstract streams of information that, through the feedback, will change the state of the reservoir. However, in physical reservoir computing, the feedback is a real physical value and provides energy to the mechanical body, e.g. in form of forces.

Although it seems adding a feedback loop is a rather small change compared to the bigger picture, the previous mathematical models based on Boyd and Chua's work are not applicable anymore. Fortunately, there exists a framework that can deal with feedbacks, i.e. non-fading memory tasks. Maass et al. (2007) employed the theory of feedback linearization from control theory to prove that neural models can be modified via static, but nonlinear feedback and readouts to approximate almost any smooth dynamical system. The limitations are only the smoothness and the degree of freedoms, i.e. a two-dimensional system as a building block can only approximate other two-dimensional systems.[11] One simplified way is to look at the standard model to describe an n-dimensional nonlinear differential system, i.e. $\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})u$ with $\mathbf{x} \in \mathbb{R}^n$. This means we have n integrators which depend on the previous states through the nonlinear mapping $f(\mathbf{x})$ and on the input through $g(\mathbf{x})$. Feedback linearization is able to overwrite both functions, i.e. $f(\mathbf{x})$ and $g(\mathbf{x})$ through applying, loosely speaking, the inverse effect.[12] In addition, we can add to the feedback a nonlinearity that "overwrites" the linearized system with new, desired dynamics in order to emulate the desired target computation, i.e. dynamic input–output mapping. Note that this is similar to finding a controller for a linear system by applying a pole-placement approach. We want the overall system to behave in a certain way (which is reflected by desired eigenvalues of the system matrix) and we try to find the right feedback controller to achieve that, for example, with Ackermann's method.

[10] There can be one or multiple feedback loops.

[11] Note that one can argue that combing multiple such systems would then again allow us to approximate systems of higher order as well, see Maass et al. (2007).

[12] Note that not every nonlinear system is feedback linearizable. However, there is a formal process to check for that by applying Lie derivatives. Note that this is equivalent to constructing the controllability matrix $\mathbb{C} = [B, BA, BA^2, \ldots, BA^n]$ in linear systems. We refer to the reader to Slotine and Lohmiller (2001) for an excellent introduction. For a more in-depth discussion, we refer to Isidori (2001).

Without going into details, the mathematical proof employed in Hauser et al. (2012) is demonstrating that certain types of nonlinear mass–spring–damper systems are feedback linearizable, i.e. can be used as a basic template that can be overwritten as new desired dynamics.

However, the proof presented by Hauser et al. (as well in the original work by Maass et al.) is only for one single system, i.e. one mass–spring–damper system, and it needs static, but still nonlinear feedbacks and readouts. The mathematical proof for more complex systems is nontrivial due to the complexity. Nevertheless, as before with the feedforward setup, the step to full network structures with linear readouts and linear feedbacks (as in Fig. 1) is reasonable. This is supported by numerous simulation results as well as real-world examples that show the usability of the approach.

While the models for the feedforward setup give us some (however) general guidelines, the models for the feedback setup, unfortunately, are not very insightful with respect to how to build better reservoirs. Maybe the only point is that such systems can be rather small in size to be useful. This has been shown and discussed with simulation examples in Hauser et al. (2012).

In the context of physical reservoir computing in robotics, however, we are very interested in understanding how to design and build better reservoirs. Therefore, the following section discusses how abstract concepts in reservoir computing are mapped onto real physical interpretations in physical reservoir computing setups.

2.3 Connecting Theoretical Models to the Real World

While conventional reservoir computing does not care about the physical significance of inputs or outputs, naturally, physical reservoir computing is much more concerned with physical meanings. This is particularly true in the context of robotics as we hope to gain insight into how to design better robot bodies to be exploited as computational resources.

For example, the input is not just an abstract time series, but is a real physical quantity that changes over time. The most natural interpretation in the light of the proposed models, i.e. using mass–spring–damper systems, inputs are forces that work on the body of the robot. This could be through the interaction of the robot with the environment, e.g. by grasping an object or by locomoting on the ground. But this could be also in the context of a sensor setup, where the sensor has to physically interact with the environment to gain information for measurements. Similarly, feedback signals (as in the feedback setup) can be forces obtained through the environment or through internal actuators that can change the state of the system.

Both, input and feedback, don't have to be restricted to forces but can be any physical entity that changes the state of the reservoir sufficiently to be picked up by the readouts. This also means that we don't have to use necessarily mechanical structures for the reservoir, but we could think also of chemical systems, electromagnetic interactions, etc. Of course, a combination of them is valid as well. Smart material and additive manufacturing, which both have gained a lot of traction recently,

will play a crucial role here. We refer the reader to Sects. 6 and 7 for a more in-depth discussion.

As previously pointed out in the introduction, one of the key properties of a reservoir is its exponential stability or in other terms fading memory. While in abstract reservoir computing setups one has to make sure to have stable systems (e.g. it's quite common to re-scale the connectivity matrix in ESNs), stability is a natural property in mechanical structures used for robotics, i.e. stability comes for "free."

Furthermore, there is a clear connection between material properties and the fading memory, i.e. how fast the system forgets its initial state. If we consider linear mass–spring–damper systems, the real part of the eigenvalue of the system matrix describes how fast, e.g. an impulse input is "faded out." More precisely, the exponential hull curve for the response of the system is defined by $e^{-d/m}$ with d being the linear damping factor and m the mass. This can be considered directly in the design process by choosing the right material or even by including mechanisms, e.g. through smart materials, to change the damping to adapt the reservoir and its corresponding fading memory for different tasks.

Another interesting aspect of physical reservoirs is that often there is not a clear border that separates them from the environment. Let's take the example of a fish-inspired underwater robot that we want to control for locomotion. The input (force) will come from some form of actuation in the system. This will change the dynamic state of the fish body. In addition, however, this also introduces changes in the water environment, which can reflect back onto the fish. This means the water is actually part of the reservoir and so are nonlinearities in the physical embodiment of the input (e.g. nonlinear features in the electrical motor) and the readout (e.g. nonlinear properties of the sensors). This points to the idea of embodiment, which states that a close interaction of body and environment is fundamental for the rise of intelligent behaviour (see, e.g. Pfeifer and Bongard 2006).

Another point of difference is that abstract reservoir computing approaches assume that the readout has access to the full state of the reservoir. However, practically, in physical reservoir computing setups in robotics most of the time this is not possible. It turns out that knowing the full state of the reservoir is not necessary and one can achieve working physical reservoir computing setups with a subset of the states.

In addition, it is quite typical in robotics to have a wider range of sensors measuring different quantities instead of measuring directly the state of the system. States are an abstract concept anyway and it's well established in linear control theory that systems can be easily transformed into others with the right matrix transformation. Practically, we can only measure real physical quantities, and the generic reservoir computing does not allow us any additional signal transformation at the readout. Practically, we have measurements that might provide redundant information. For example, two different gyroscopes at different locations on the robot will very likely produce partially redundant information. But also completely different sensors will potentially share information about states. Ideally, we want to reduce the amount of redundant information. The worst-case scenario would be linearly dependent readout

signals.[13] However, some overlap is fine since linear regression by definition will pick and choose (by assigning the right weights) the best signals to get as close as possible to the target.[14]

Another important difference is that physical reservoir computing approaches are naturally leading us to think in dynamical systems terms instead of abstract mathematical frameworks. This provides us with some intuition on how to build intelligent machines based on this concept. For example, Fuechslin et al. (2011) suggested to look at computation in the physical reservoir computing context as mechanical structures that implement attractor landscapes. This could help us to design physical structures with a bias, for example, by having implemented a number of potential attractor points and limit cycles (which would be different behaviours—e.g. various locomotion gaits) and corresponding readouts/feedbacks can exploit them. For a discussion of the potential of this approach, we refer to Hauser and Corucci (2017).

3 Example Cases from Robotics

There are a number of examples from the literature that have demonstrated the usefulness of applying physical reservoir computing in robotics. The implemented computations include abstract mapping (as a proof-of-concept) to applications in sensing and controlling.

We will first present results from simulations and then discuss real-world platforms.

In the original work by Hauser et al. (2011), a number of example computations have been presented that might be interesting in the context of robotics. The results range from learning to emulate a simple Volterra series,[15] a model of a nonlinear pendulum, inverse dynamics of a two-link robot arm, and NARMA systems taken from Atiya and Parlos (2000) that have been known to be hard to emulate with recurrent neural networks due to their long-term dependencies, see for example Hochreiter and Schmidhuber (1997).

An extension of the robot arm example has been introduced in Hauser and Griesbacher (2011), where the mechanical reservoir is directly connected to the two-link robot arm. The setup was able to learn to control itself to move along a desired (figure eight) end-point trajectory. Later, similar results were achieved using L-systems to "grow" the structure around the arm (Bernhardsgrütter et al. 2014).

---

[13] Note that this is directly connected to the degradation of the signal separation property discussed in Sect. 2.1, see Hauser et al. (2014) for a discussion.

[14] Note that linear regression is solving the optimization problem to find a set of optimal weights $\mathbf{w}^*$ that minimize the quadratic error between the target output $y_t(t)$ the produced output $y(t)$.

[15] The reason being that the theoretical model from Hauser et al. (2011) is based on Volterra series and it's a generic way to approximated nonlinear, exponentially stable sets of differential equations.

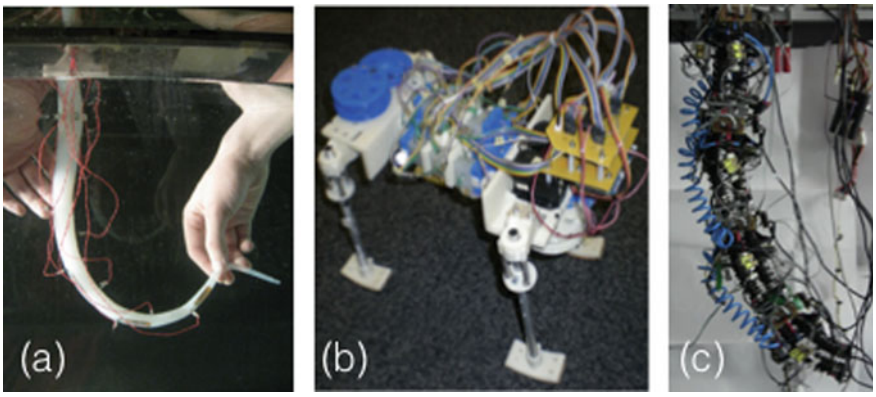


Fig. 2 Example cases with real physical platforms. **a** Octopus arm setup used in Nakajima et al. (2018a) and (Nakajima et al., 2013), **b** Kitty robot from Pfeifer et al. (2013), **c** Pneumatically driven, modular robot arm from Eder et al. (2017)

As pointed out before in Sect. 2, Hauser et al. (2012) also introduced a second theoretical model dealing with physical reservoir approaches that consider linear feedback loops from the output back into the reservoir. In the same work, a number of example cases were presented. They ranged from learning to emulate robustly various nonlinear limits cycles (e.g. like the van der Pol equations and others) to switching between different gaits by changing the readout weights. Finally, they also learned to produce three different limit cycles with the same set of readout weights. The change from one to another was only driven by a change in input. Specifically, the otherwise constant input force was switched to one of three different constant values. This means, depending on how strong the physical reservoir was squeezed, it reacted with a different limit cycle. This example is particularly interesting as it points to the possibility to change behaviour (e.g. gaits) triggered by changes in the environment (through the change in forces acting on the robot, e.g. by putting a heavy weight on it). For a deeper discussion of this results, we refer to Hauser et al. (2014) and Fig. 2.

Caluwaerts et al. (2013) were able to show that the complex dynamics of worm-like structures built based on the tensegrity principles can be used to produce robustly control signals for the locomotion of the robot. They used also learning to optimize the locomotion behaviour. In addition, the work showed that the body can be used as a sensor to classify the ground (flat vs bumpy) by using it as a physical reservoir. Another simulation work showing the capability of classification with the help of mechanical structures was introduced by Johnson et al. (2014). They used the same mass–spring–damper systems as suggested by Hauser et al. (2011) to build a system that can actively discriminate different shapes.

Still in simulation, but with a different approach to physical reservoir computation in robotics has been suggested by various people by using more structured reservoirs. Instead of random networks, bio-inspired morphologies were used. Naturally, they

are more constrained with respect to their potential computational power, but they work surprisingly well. For example, Sumioka et al. (2011) used a mechanical setup that was loosely inspired by the muscle–tendon–bone arrangements in the biological system as a reservoir. Despite the simplicity of the model, they were able to emulate a Volterra series. Nakajima et al. (2013) built a network of nonlinear mass–spring–damper systems simulating an octopus arm. The input was applied at the shoulder and the readout was obtained by measuring the strain throughout the body, i.e. similar to proprio-receptive sensing.

More recently, even more structured setups have been shown to work as a reservoir as well. Yamanaka et al. (2018) demonstrated that a 2D grid structure, representing a model of a soft cloth, can be used as a reservoir as well. In this particular work, the authors investigated which stiffness and damping values lead to better performances for various computational tasks.

While simulation results are interesting in themselves, robotics is a field of research that ultimately wants to develop frameworks and technologies that can be used under real-world conditions. The same is true for physical reservoir computing in robotics. The first step[16] has been made by a series of work by Nakajima et al. They used a silicone-based octopus-inspired robot arm as a reservoir. The input was in form of rotation of the arm introduced at the shoulder via an electric motor (compare Figs. 2a and 3a). The readout was obtained via strain sensors along both sides of the octopus arm. Nakajima et al. were able to demonstrate that this setup was able to learn to emulate timers and various digital functions like delays and parity (demonstrating the memory capacity of the system), see Nakajima et al. (2014). In the same work, it has been even shown that the setup is able to learn to produce a control signal for itself, i.e. the body of the octopus arm is used as a computational resource (i.e. reservoir) to control robustly the movement of the same arm. Interestingly, when they interacted with the arm (touching and grasping it during the movement), the arm started to react to it by naturally looking movements, i.e. it looked as if it tried to wriggle itself free and sometimes it stopped completely until it was released. While this interpretation is clearly anthropomorphic, it nevertheless points to the fact that the feedback loop through the environment plays a crucial role in the behaviour in such a setup.

Nakajima et al. also emulated with the same setup various NARMA systems with different complexities and other nonlinear, dynamical mappings (Nakajima et al. 2018a, b) with the goal to systematically explore the limitations of the computational power, e.g. they quantified the amount of available fading memory by comparing it to standard ESN.

Zhao et al. (2013) applied the idea of physical reservoir computing to locomotion. The physical reservoir was constructed out of a bio-inspired assemble of hard and soft parts (resembling loosely the spinal structure) and randomly distributed pressure sensors (compare Figs. 2b and 3b). The system was able to robustly locomote

[16] Note that there had been previous work on physical reservoir computing, e.g. the previously mentioned "bucket in the water" setup by Fernando and Sojakka (2003), but none in the context of robotics.

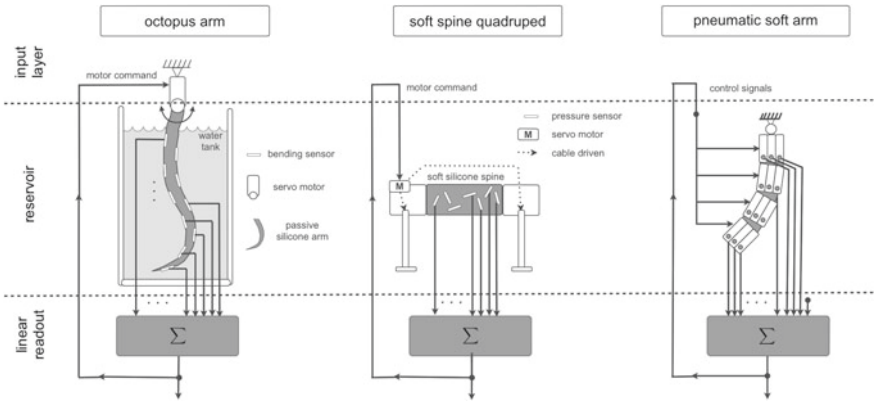


Fig. 3 Schematics setups for real platform shown in Fig. 2 and discussed in the text. Also compare to Fig. 1. Figures adapted from Hauser et al. (2014) and Eder et al. (2017)

and change direction. Another locomotion example on a real-world platform was provided by Caluwaerts et al. (2014) using a mechanical tensegrity structure to learn an oscillator to control it.

Physical reservoir computing has also been shown to work for other tasks as well. For example, Eder et al. (2017) used a pneumatically driven robot arm and its nonlinear dynamics as a reservoir to control the arm (compare Figs. 2c and 3c). While the results were interesting, they also revealed clearly limitations for using this approach on this particular platform.

Besides the mapping of abstract computations and to learn to control a system, real physical reservoir computing setups have also been used now for sensing, classification, and even modelling. For example, Soter et al. (2018) used a physical reservoir computing setup extended by an RNN to predict the movement (actually the pixels of the video of the movement) of an octopus arm solely based on the proprio-receptive information (bending). It learns "imaging" the visual movement of the arm through observing its internal states.

Recent work by Judd et al. (2019) was able to use physical reservoir computing to detect objects in the vicinity of the moving octopus arm without touching them. This included classification, i.e. is the object there or not, as well regression, i.e. estimation of where it is located. The results point to the fact that the environment indeed is part of the reservoir and should play a crucial role when developing intelligent systems (sensing and control) with physical reservoir computing setups in robotics.

In summary, there is a substantial amount of exciting results, however, there is also still a wide range of interesting, yet to be explored ideas. To inspired future researcher working in the field, we discuss in the next section what advantages a physical reservoir computing approach in robotics might have.

4 Advantages of Physical Reservoir Computing in Robotics

The application of physical reservoir computing in robotics has a number of remarkable aspects and advantages.

For example, the readout for conventional reservoir computing is typically the same for every state, since it's an abstract mapping. However, in a physical setup, we need real physical sensors to obtain the information about the state of the system. As previously pointed out, it's not trivial to get the information about the full state in a real robot. However, it turns out that is also not needed. Since the readout weights are found with the help of linear regression, we can provide a variety of sensory signals. Linear regression automatically chooses the "best" ones, i.e. the ones that provide the best information to reduce the error between output and target signal. This also means that linear regression doesn't "care" what kind of signal we provide or where it comes from. This means we can combine sensory information from gyroscopes, pressure sensors, stretch sensors, and even binary switches. We can even include and combine sensors that measure in completely different physical domains, e.g. a force sensor with sensors to measure chemical concentration or temperature. If they provide useful information, they will get high enough readout weights assigned.[17]

Another advantage is the inherent robustness of the approach. Examples in Hauser et al. (2012) demonstrated that the learned limit cycles, although only provided with data from a certain range of the state space, seem to exhibit global stability. We believe this is due to the inherent global stability of the underlying mass–spring–damper systems. Obviously, practically there are limitations with respect to global stability, after all, mechanical structures will eventually break under too high forces. Nevertheless, this points to the underlying property of inherent stability, which can be exploited. Maybe frameworks like contraction theory (Lohmiller and Slotine 1998) or concepts like passivity from control theory, especially more recent extensions like Forni and Sepulchre (2019), might be able to provide us with a general proof of this intuition.

Another interesting aspect of robustness is that smaller changes in the reservoir, like tearing of parts of the robot because of rough use or interaction with the environment, can be easily counteracted by slightly changing the weights of the readout layer. As a result, imprecise design is not a problem. We don't need a perfectly controlled fabrication process as long as the final result is close enough for being useful. This points to a very different way to build robots and it suggests a quite radical break with conventional robotic design approaches (see discussion later in Sect. 7).

Another advantage of reservoir computing in general is that learning through applying linear regression is very fast. This is particularly important for robotics. Also, online learning algorithms are thinkable using the wide range of available tools that implement recursive, online versions of linear regression.

Another particularly interesting advantage is that noise is beneficial for the reservoir computing setup that uses feedback. It's common practice in reservoir computing in general to add noise to learning data when using feedback loops. This helps to

[17] Note that a limitation is that the different sensors should provide linearly independent information.

learn instead of a simple trajectory (e.g. of a limit cycle) an attractor region around this trajectory making it more robust. Interestingly, while in simulations, noise has to be added artificially, it seems the noise present in real physical robots is enough and appropriate.

Finally, looking at robot design from the viewpoint of reservoir computing leads to novel approaches to build robots and might be a way around the implicit assumption in the robotics community that we have to build robots such that we can easily model them. In the context of physical reservoir computing underactuation, i.e. degrees of freedom or states of the system that are not directly controlled by the input are a prerequisite for a computationally powerful body. Hence, using more compliant and soft structures for building robots might be beneficial. A more detailed discussion of this point is carried out in Sect. 6.

Besides the listed advantages, clearly, the physical implementation of reservoirs in robot bodies also implies a range of problems and limitations, which are discussed in the next section.

5 Limitations of Physical Reservoir Computing in Robotics

In general, the implementation of a reservoir in a real, physical body also always means the introduction of physical limitations. While abstract approaches, e.g. like ESNs, in principle, don't have to worry about limitations (assuming we have the right dynamical properties to make a useful reservoir), a real body also means real-world constraints. For example, (stable) mechanical structures always have the effect of a low-pass filter. Consider a soft structure, i.e. a body with low stiffness. A high-frequency input will not get transmitted effectively enough through the body. This means whatever information is provided by this particular input is lost and the reservoir is not useful for this kind of computation. On the other hand, very stiff structures have no problem to transmit high-frequency vibration throughout their bodies. But this also means it has a very low fading memory as every effect is almost immediately damped out.

Another limitation, which is inherent to any reservoir computing setup in general, is that we don't directly control what is happening in the reservoir. The readout uses linear regression to pick and choose the best signals; however, we have very little control over how to improve the performance of the reservoir. There has been some work on improving the performance through changing the structure, but so far almost none of them have been implemented in real physical systems. We discuss the potential of such approaches later in Sects. 6 and 7.

In general, the lack of control over the dynamics of the reservoir is a big point of discussion in a robotics application. Conventional approaches to design and control robots are particularly keen to build systems that are easily modelled and, therefore, easily controlled. However, this reduces the computational power of the corresponding morphology and therefore makes them unfit to serve as reservoirs. This is an interesting tension which is particular to robotics and it's not clear if these two seem-

ingly opposite positions can be (or even should be) reconciled. However, since both have advantages and disadvantages, most likely both will be used as complementary approaches and they will be implementing different tasks and at different levels in future robotic systems.

Connected to that is the problem of safety. If we don't have a good enough model of the dynamics of the reservoir, we can't prove the safety or stability of the learned behaviour. Again, biological systems don't seem to have a problem with that and maybe the previously mentioned inherent (seemingly global) stability (see Sect. 4) might be the key to this problem.

Finally, reservoir computing in principle is based on supervised learning. This means we have to provide the right input–output data set. While this might be possible for sensor applications, this is quite tricky for feedback-based setups, e.g. in locomotion. We often don't know the best mapping that we should learn to emulate. One possible solution is to start with a good enough guess and use online adaptation mechanisms to improve on them, see, e.g. Caluwaerts et al. (2014).

6 Connection to Soft Robotics

As pointed out before, one of the main motivations to use nonlinear mass–spring–damper systems as a basic building block for the models of mechanical reservoirs was the observation that soft-bodied robots, as well as bodies of biological systems, can be described elegantly by a network of such systems.

However, there is a much stronger connection between physical reservoir computing and soft robotics if we have a closer look at the dynamical properties of soft-bodied structures.[18] They typically exhibit complex, nonlinear dynamics, a high-dimensional state space, underactuation, and noise. These properties are all perceived as negative in conventional robotics since they make it more difficult to model and consequently control such systems. However, on the other side, these are the exact properties that are needed to build computationally powerful reservoirs. So, instead of avoiding complexity in the dynamics of robots, through the framework of reservoir computing, we can embrace them and, consequently, exploit them. One could even claim that reservoir computing is a good candidate to solve the control problem in soft robotics, see, e.g. Hauser (2016) for a discussion.

In addition, the rise of soft robotics is strongly coupled with the advancements in material science and additive manufacturing. These two fields of research have been proven over and over again to be able to extend the already rich set of dynamically interesting building blocks. Most of them have the potential to eventually be exploited in the context of physical reservoir computing. Most of the so-called smart materials have highly interesting, nonlinear dynamic behaviours that are potentially

[18] In the context of Sect. 6, we discuss only soft robotics structures. However, all the points are also true for biological systems or hybrid structures, e.g. the mixture of soft artificial structures and biological tissue.

beneficial to boost the computational power of a reservoir. In addition, the input can be "perceived" by the reservoir through mechanisms offered by smart materials. Direct physical interaction with the environment can be translated into changes in the dynamics. For example, the stiffness can change in some parts of the body through physical interaction or a soft robot "bumping" into an object can serve as a reliable input for the setup to switch behaviour, i.e. to switch to a different attractor space.

Another important point of connection between soft robotics and physical reservoir computing is that both research fields emphasize the importance of the embodiment. While conventional robotics tries to reduce the influence of the body of the robots (and the environment) as much as possible, soft robotics embraces the idea to outsource functionality to the body—and so does physical reservoir computing.

Finally, smart materials have the great potential to serve as substrates to implement future technologies that are capable of extending the notion of physical reservoir computing. This includes optimization of the reservoir and learning in material. Some of these future directions are discussed in the following chapter.

7 The Future of Physical Reservoir Computing in Robotics

While there has been quite a lot of excitement around the idea of outsourcing computation to the body of robots in the form of a reservoir, there is still a large underexplored potential pointing to a number of interesting research questions and corresponding applications.

Foremost, reservoir computing, physical or not, naturally raises the question about how to optimize the reservoir. While it's a clear advantage that we don't have to adapt parameters in the reservoir during learning, it also limits our control. Linear regression always finds the optimal solution based on the signal (readouts) it receives. However, we don't know how to get better or more information out of our reservoir. Furthermore, as previously pointed out in Sect. 2, the existing theoretical frameworks don't give us any specific design guidelines. We only get very high level suggestions of what kind of properties we should be looking for in a reservoir—i.e. high-dimensionality, underactuation, and nonlinear dynamics.[19] For a given computational task, e.g. a specific nonlinear controller or specific sensory filter that we want to emulate, the models don't give us a direct mapping to (in some sense) optimal reservoir. We don't know how many mass–spring–damper systems (or systems of equivalent complexity) are needed to achieve a certain performance. Therefore, typically, reservoirs are initialized randomly (in simulation) or pre-existing robotic structures, which had been designed with another functionality in mind, are simply exploited as a reservoir.

Since optimizing the reservoir is a general challenge that people are addressing, we might be able to draw inspiration from their results for specific applications in robotics. For example, in the context of LSMs, some approaches have been sug-

[19] And noise in the case of external feedback loops.

gested. Sussillo and Abbott proposed the First-Order Reduced and Controlled Error (FORCE) algorithm (Sussillo and Abbott 2009) where they adapt synaptic strengths by feeding back the error. Another approach in the context of LSMs, in this case, based on reward-modulated Hebbian learning, has been suggested by Hoerzer et al. (2014). They use a stochastic approach for the optimization of the reservoir. In the case of LSMs, which are inspired by brain structures, there is a clear connection to biological adaptation mechanisms, i.e. neuro-plasticity. However, in the case of physical reservoir computing in robotics, the picture becomes more blurry as there are many different ways to change real physical, morphological features, e.g. ranging from simply changing specific parameters like stiffness to complex growing processes. To the best of our knowledge, so far, there have been only simulation results to address this issue. For example, Hermans et al. (2014, 2015) proposed to use backpropagation through mechanical structures leading to optimal physical reservoirs. While their results are impressive, it would be great to see their approaches implemented in real physical bodies. While this has been out of the reach for a long time, mostly due to practical limitations, with the recent rise of soft robotics and additive manufacturing, we suddenly have a much broader set of tools available to build physically adaptive systems. There is a big potential for a range of interesting research directions with respect to learning directly in materials to improve the reservoir. Also, research in protocells, synthetic biology, and even biology (e.g. stem cells) is all very likely to be able to contribute to this idea. In general, this points clearly to much more sophisticated physical bodies and we might have to let go of the idea that we need to control explicitly every single aspect of a robotic system.

Another aspect connected to adaption in the reservoir is that we need more performance measurements that can help us to guide the adaptation process. For a lot of interesting tasks in robotics, we don't have a clear-cut target function. For example, if we have a given body and we want to exploit it for locomotion, we don't know how the optimal input–output mapping should look like. However, meaningful performance measurements can guide a corresponding adaptation process, either directly in the body/reservoir and/or at the readout. One promising approach is to use abstract measurements based on information-theoretic approaches. For example, for the measurements based on the concept of Predictive Information, see, e.g. work by Martius et al. (2013) and Martius (2014). Another possibility, suggested by Ghazi-Zahedi and Rauh (2015) and Ghazi-Zahedi et al. (2017), could be to measure directly how much computation is outsourced to the morphology (in our case the physical reservoir). These are just a few examples, but of course, there is a very strong body of work on statistical approaches in general. A reinterpretation in the light of physical reservoir computing might provide a number of exciting new approaches.

The idea of changing the body to improve the reservoir has an even wider implication. If we consider morphological structures (e.g. in form of mechanical systems) as being capable of representing computational functionality, then concepts like self-assembly, growing, and self-healing become an entirely new, additional meaning: Changing the body is changing the computational functionality. Instead of simply assembling to a morphological structure, which is interesting in itself, we also build an underlying computational functionality. We assemble also a program. The same

is also true for growing (Hauser 2019). A system (biological or artificial) capable of growing is able to construct functionality through this process. In addition, if such a system has self-healing capability, it wouldn't simply fix a broken leg, but also the underlying computational functionality.

In this context, it is interesting to note that there are many different ways to *build* a reservoir. One can easily image two reservoirs that are morphologically different but have the same computational power. This implies the growing process might have a primary goal, for example, to make a limb for locomotion, while a secondary goal (with less constraints) could be to construct a reservoir structure. As shown in real-world physical reservoir computing setups with existing robotic prototypes, already existing physical bodies can be exploited as reservoir even if during their design and building process this has not been considered at all.

Another point of discussion is how, in robotic applications, can we get away from the more homogeneous structures present in classical reservoir computing approaches. Traditional reservoir computing uses one simple module (e.g. an integrate and fire models in ESN, or spiking neural models in LSM) and then connects them with each other. However, the fundamental building blocks are always the same. In the context of physical reservoir computing and robotics, this seems to be quite an unnatural approach. Intelligent systems are made of a variety and dynamically different body parts. This again can be of great benefit if exploited properly. Unfortunately, so far this has not been really explored in the community.

One way to investigate this idea could be to include buckling mechanisms or hysteresis—both are quite common and naturally occurring dynamical features. Buckling could potentially facilitate the learning of different attractors (e.g. different gaits as in Hauser et al. 2012 or switching between different controllers as in Füchslin et al. 2013). There has also been very interesting work by Kachman et al. (2017) who showed from the viewpoint of thermodynamics that dynamical systems with two (or more) stable states can lead to self-organization under (thermal) noise. Hysteresis could also be useful, for example, to help to detect changes in the direction of the input. Note that real muscles have hysteresis and even more complex memory effects, see, for example Paetsch et al. (2012).

Another field of application of physical reservoir computing in robotics has been so far under-explored, i.e. the implementation of dynamically complex sensors. While there has been some work, see Sect. 3, there are still a lot of interesting open research questions and potential applications. One can think of the implementation of interesting (signal processing) filters, information-based approaches, or adaptive sensing morphologies that change the sensing modality based on different tasks. This is especially interesting if we consider adaptation mechanisms to optimize the physical reservoir (i.e. morphology of the sensor) to improve the sensing performance. Information-theoretic approaches and research on sensing systems in biological systems (especially insects) will play a crucial role in this context. Also, the idea of building more structured morphologies for the reservoir is a valid angle to be explored. For example, the project "Computing with Spiders' Webs" (Hauser and Vollrath 2017) works on spider web-inspired approaches to build physical reservoir computing sensors to measure vibration and flow.

As one can see there is still a great, unexplored potential in the approach of physical reservoir computing in the context of robotics. The field is rich with research opportunities and we are looking very much forward to see novel approaches, more robots, and exciting research results from the community.

Acknowledgements This publication has been written with the support of the Leverhulme Trust Research Project RPG-2016-345.

References

A.F. Atiya, A.G. Parlos, New results on recurrent network training: unifying the algorithms and accelerating convergence. IEEE Trans. Neural Netw. **11**(3), 697–709 (2000)

R. Bernhardsgrütter, C.W. Senn, R.M. Füchslin, C. Jaeger, K. Nakajima, H. Hauser, Employing L-systems to generate mass-spring networks for morphological computing, in *Proceedings of International Symposium on Nonlinear Theory and its Applications (NOLTA2014)*. Research Society of Nonlinear Theory and its Applications, IEICE (2014), pp. 184–187

S. Boyd, Volterra series: engineering fundamentals. PhD thesis, UC Berkeley (1985)

S. Boyd, L. Chua, Fading memory and the problem of approximating nonlinear operators with Volterra series. IEEE Trans. Circuits Syst. **32**(11), 1150–1161 (1985)

K. Caluwaerts, M. D'Haene, D. Verstraeten, B. Schrauwen, Locomotion without a brain: physical reservoir computing in tensegrity structures. Artif. Life **19**, 35–66 (2013)

K. Caluwaerts, J. Despraz, A. Işçen, A.P. Sabelhaus, J. Bruce, B. Schrauwen, V. SunSpiral, Design and control of compliant tensegrity robots through simulation and hardware validation. J. R. Soc. Interface **11**(98), 20140520 (2014)

M. Eder, F. Hisch, H. Hauser, Morphological computation-based control of a modular, pneumatically driven, soft robotic arm. Adv. Robot. **32**(7), 375–385 (2017)

C. Fernando, S. Sojakka, Pattern recognition in a bucket, in *Advances in Artificial Life SE - 63*, vol. 2801, Lecture Notes in Computer Science, ed. by W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich, J.T. Kim (Springer, Berlin, 2003), pp. 588–597

F. Forni, R. Sepulchre, Differential dissipativity theory for dominance analysis. IEEE Trans. Autom. Control. **64**(6), 2340–2351 (2019)

R.M. Füchslin, H. Hauser, R.H. Luchsinger, B. Reller, S. Scheidegger, Morphological computation: applications on different scales exploiting classical and statistical mechanics, in *Proceedings of the 2nd International Conference on Morphological Computation*, September 2011, ed. by R. Pfeifer, S. Hidenobu, R.M. Füchslin, H. Hauser, K. Nakajima, S. Miyashita (2011)

R.M. Füchslin, A. Dzyakanchuk, D. Flumini, H. Hauser, Morphological computation and morphological control: Steps toward a formal theory and applications. Artif. Life **19**, 9–34 (2013)

K. Fujii, K. Nakajima, Harnessing disordered-ensemble quantum dynamics for machine learning. Phys. Rev. Appl. **8** (2017)

K. Ghazi-Zahedi, J. Rauh, Quantifying morphological computation based on an information decomposition of the sensorimotor loop, in *The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)*, vol. 27 (2015), pp. 70–77

K. Ghazi-Zahedi, C. Langer, N. Ay, Morphological computation, synergy of body and brain. Entropy **19**(9) (2017)

H. Hauser, Morphological computation – a potential solution for the control problem in soft robotics, in *Proceedings of the 19th International Conference on CLAWAR 2016* (2016)

H. Hauser, Resilient machines through adaptive morphology. Nat. Mach. Intell. **1**(8), 338–339 (2019)

H. Hauser, G. Griesbacher, Moving a robot arm by exploiting its complex compliant morphology, in *Proceedings of the 2nd International Conference on Morphological Computation*, September 2011, ed. by R. Pfeifer, S. Hidenobu, R.M. Füchslin, H. Hauser, K. Nakajima, S. Miyashita (2011)
H. Hauser, F. Corucci, Morphosis - taking morphological computation to the next level. Biosyst. Biorobot. **17**, 117–122 (2017)
H. Hauser, F. Vollrath, Leverhulme Trust Research Project RPG-2016-345, Computing with Spiders' Webs (2017)
H. Hauser, A.J. Ijspeert, R.M. Füchslin, R. Pfeifer, W. Maass, Towards a theoretical foundation for morphological computation with compliant bodies. Biol. Cybern. **105**, 355–370 (2011)
H. Hauser, A.J. Ijspeert, R.M. Füchslin, R. Pfeifer, W. Maass, The role of feedback in morphological computation with compliant bodies. Biol. Cybern. **106**(10), 595–613 (2012)
H. Hauser, K. Nakajima, R.M. Füchslin, Morphological computation – the body as a computational resource, in *E-book on Opinions and Outlooks on Morphological Computation*, ed. by H. Hauser, R.M. Füchslin, R. Pfeifer (2014), pp. 226–244
M. Hermans, B. Schrauwen, P. Bienstman, J. Dambre, Automated design of complex dynamic systems. PLoS ONE **9**(1) (2014)
M. Hermans, M. Burm, T. Van Vaerenbergh, J. Dambre, P. Bienstman, Trainable hardware for dynamical computing using error backpropagation through physical media. Nat. Commun. (2015)
S. Hochreiter, J. Schmidhuber, Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
G.M. Hoerzer, R. Legenstein, W. Maass, Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning. Cereb. Cortex **24**(3), 677–690 (2014)
K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators. Neural Netw. **2**, 359–366 (1989)
A. Isidori, *Nonlinear Control Systems*, third edn. (Springer GmbH, 2001)
H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)
C. Johnson, A. Philippides, P. Husbands, Active shape discrimination with physical reservoir computers, in *ALIFE 14: The Fourteenth Conference on the Synthesis and Simulation of Living Systems*, vol. 14 (2014), pp. 176–183
T. Kachman, J.A. Owen, J.L. England, Self-organized resonance during search of a diverse chemical space. Phys. Rev. Lett. **119** (2017)
W. Lohmiller, J.-J.E. Slotine, On contraction analysis for non-linear systems. Automatica **34**(6), 683–696 (1998)
W. Maass, T. Natschlaeger, H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. **14**(11), 2531–2560 (2002)
W. Maass, P. Joshi, E.D. Sontag, Computational aspects of feedback in neural circuits. PLoS Comput. Biol. **3**(1), e165 (2007)
G. Martius, R. Der, N. Ay, Information driven self-organization of complex robotic behaviors. PLOS ONE **8**(5), 1–14 (2013)
G. Martius, L. Jahn, H. Hauser, V. Hafner, Self-exploration of the stumpy robot with predictive information maximization, in *From Animals to Animats 13*, ed. by A.P. del Pobil, E. Chinellato, E. Martinez-Martin, J. Hallam, E. Cervera, A. Morales. Lecture Notes in Computer Science, vol. 8575 (Springer International Publishing, 2014), pp. 32–42
K. Nakajima, H. Hauser, R. Kang, E. Guglielmino, D.G. Caldwell, R. Pfeifer, A soft body as a reservoir: case studies in a dynamic model of octopus-inspired soft robotic arm. Front. Comput. Neurosci. **7**(91), 91 (2013). Research Topic: Modularity in Motor Control: From Muscle Synergies to Cognitive Action Representation
K. Nakajima, T. Li, H. Hauser, R. Pfeifer, Exploiting short-term memory in soft body dynamics as a computational resource. J. R. Soc. Interface **11**(100), 20140437 (2014)
K. Nakajima, H. Hauser, T. Li, R. Pfeifer, Exploiting the dynamics of soft materials for machine learning. Soft Robot. **5**(3) (2018a)

K. Nakajima, H. Hauser, T. Li, R. Pfeifer, Exploiting the dynamics of soft materials for machine learning. Soft Robot. (2018b)

C. Paetsch, B.A. Trimmer, A. Dorfmann, A constitutive model for activepassive transition of muscle fibers. Int. J. Non-Linear Mech. **47**(2), 377–387 (2012)

R. Pfeifer, J.C. Bongard, *How the Body Shapes the Way We Think* (The MIT Press, 2006)

R. Pfeifer Q. Zhao, K. Nakajima, H. Sumioka, H. Hauser, Spine dynamics as a computational resource in spine-driven quadruped locomotion, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2013)

B. Schrauwen, D. Verstraeten, J. Van Campenhout, An overview of reservoir computing: theory, applications and implementations, in *Proceedings of the 15th European Symposium on Artificial Neural Networks* (2007), pp. 471–482

Y. Shim, P. Husbands, Feathered flyer: integrating morphological computation and sensory reflexes into a physically simulated flapping-wing robot for robust flight manoeuvre, in *ECAL*, ed. by F. Almeida e Costa et al. (Springer, Berlin/Heidelberg, 2007), pp. 756–765

J.J. Slotine, W. Lohmiller, Modularity, evolution, and the binding problem: a view from stability theory. Neural Netw. **14**(2), 137–145 (2001)

A. Smerieri, F. Duport, Y. Paquot, B. Schrauwen, M. Haelterman, S. Massar, Analog readout for optical reservoir computers, in *Advances in Neural Information Processing Systems*, vol. 25, ed. by F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Curran Associates, Inc., 2012), pp. 944–952

G. Soter, A. Conn, H. Hauser, J. Rossiter, Bodily aware soft robots: Integration of proprioceptive and exteroceptive sensors, in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018 (2018), pp. 2448–2453

G. Soter, A. Conn, H. Hauser, J. Rossiter, Sensing through the body – non-contact object localisation using morphological computation, in *2018 IEEE International Conference on Soft Robotics (RoboSoft)* (2019)

H. Sumioka, H. Hauser, R. Pfeifer, Computation with mechanically coupled springs for compliant robots, in *IEEE International Conference on Intelligent Robots and Systems* (IEEE, 2011), pp. 4168–4173

D. Sussillo, L.F. Abbott, Generating coherent patterns of activity from chaotic neural networks. Neuron **63**(4), 544–57 (2009)

V.N. Vapnik, *Statistical Learning Theory* (Wiley, New York, 1998)

Y. Yamanaka, T. Yaguchi, K. Nakajima, H. Hauser, Mass-spring damper array as a mechanical medium for computation, in *Artificial Neural Networks and Machine Learning – ICANN 2018*, ed. by V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, I. Maglogiannis (Springer International Publishing, Cham, 2018), pp. 781–794

Reservoir Computing in MEMS

Guillaume Dion, Anouar Idrissi-El Oudrhiri, Bruno Barazani, Albert Tessier-Poirier, and Julien Sylvestre

Abstract This chapter explores the use of the Duffing nonlinearity and fast dynamics found in microelectromechanical beam oscillators for reservoir computing applications. General properties of MEMS are discussed, and the Duffing microscale beam characteristics are analyzed through analytical models and simulations. The reservoir computer is then constructed around a single such nonlinear oscillator through temporal multiplexing of the input and self-coupling via delayed feedback. The parameters of the resulting physical system are finally adjusted for optimal performance on computing the parity of a binary input stream, as well as on a spoken digit recognition task.

1 Introduction

Artificial intelligence (AI) and machine learning have progressed tremendously over recent years and are now the focus of an intense interest worldwide within many fields, with applications ranging from self-driving cars (Huval et al. 2015) to health monitoring systems (Witt et al. 2019). This rapid progress has occurred over only a few years and was driven by algorithmic advances and improvements in computing hardware (LeCun et al. 2015) that have resulted in much shorter training and validation times for AI systems. The expectation that better hardware could contribute to further improving AI systems currently fuels a large research effort to find new "computing substrates" for AI. While conventional AI is implemented with software running on general-purpose computers, it is widely accepted that much more efficient hardware implementations of AI must exist; our brains are an existence proof that some computing architectures can far exceed the density and energy efficiency of current microelectronics technology. We have published the first demonstration that microelectromechanical systems (MEMS) were an appropriate substrate for miniature, low energy consumption AI systems (Coulombe et al. 2017; Dion et al.

G. Dion · A. I.-E. Oudrhiri · B. Barazani · A. Tessier-Poirier · J. Sylvestre (✉)
Institut interdisciplinaire d'innovation technologique (3iT), Université de Sherbrooke, Sherbrooke, Canada
e-mail: julien.sylvestre@usherbrooke.ca

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_9

2018). By exploiting the nonlinearity of microfabricated mechanical oscillators, our approach implements the concept of reservoir computing (RC) (Jaeger and Haas 2004) physically in MEMS. As MEMS can be fabricated to small dimensions and therefore have high resonance frequencies (up to the GHz van Beek et al. 2007), our approach has the potential to be used as a highly efficient electrical component to implement reservoir computing.

As MEMS are also the mainstream technology for many modern sensors (Khoshnoud and de Silva 2012), our work further paves the way to the development of a new class of smart sensors with built-in data processing capabilities. As an example, we have demonstrated a MEMS displacement sensor which implements reservoir computing in the mechanical domain (Barazani et al. 2019). As the sensor is moved randomly between two positions separated by 2 μm at 20.8 Hz, it uses the nonlinear dynamics of its resonating mechanical structures to compute at every timestep when the position can change, if it had been in one of the two positions an even or an odd number of times over the last five timesteps. More recently, we have also demonstrated a MEMS accelerometer with similar neuromorphic computing capabilities (Barazani et al. 2020). By using the hardware implementation of reservoir computing in MEMS, these devices offer both sensing and non-trivial computing functions in small, highly integrated structures. We envision a number of applications for MEMS sensors integrating machine learning capabilities through our architecture (Sylvestre et al. 2018), especially in fields where small, energy-efficient systems are required, including the Internet of Things, autonomous systems, as well as mobile and wearable electronic devices.

This chapter provides a general overview of our neuromorphic computing MEMS technology. We start with an introduction to MEMS in Sect. 2, including the unique characteristics of microfabricated devices (relative to conventional devices) which are leveraged to implement computing functionalities. We discuss the modeling and analysis of nonlinear MEMS resonators (Sect. 3), leading to an example of a silicon beam design which has proven to be useful in experiments. Measurements of computing performances are presented in Sect. 4, together with observations on the tuning of the system parameters to optimize performance on different benchmark tasks.

2 Microelectromechanical Systems

Microelectromechanical systems (MEMS) are miniaturized machines able to sense or produce displacements at the micrometer and sub-micrometer scales, typically in the range of 0.1 μm to 100 μm. MEMS devices comprise structures such as beams or membranes that are able to move relative to the substrate, providing actuation (MEMS actuators, e.g., micropumps) or detection capabilities (MEMS sensors, e.g., pressure or force meters). However, the design of miniaturized actuators and sensors requires some modifications if compared to the design of conventional machines. At the scale of MEMS structures, surface forces (such as electrostatic and adhe-

sion forces) are dominant compared to volumetric forces (such as gravitational and inertial forces). For instance, water surface tension forces can completely suppress MEMS mobility and are sometimes very difficult to avoid (Van Spengen et al. 2003). On the other hand, MEMS μm-dimensions allow them to be batch produced and assembled in the same chip as the electronic circuits, resulting in cheaper (lower cost per unit), faster, and more compact monolithic devices. Furthermore, MEMS tend to demonstrate higher sensitivity, faster response, and lower energy consumption than conventional mechanisms (Ananthasuresh 2012). MEMS applications can be quite diverse and include for example printers ink-jet nozzles, airbag sensors, mirror arrays in video projectors, focusing systems in smartphone cameras, and accelerometers in smartphones or personal fitness trackers.

2.1 MEMS Fabrication

In order to manufacture MEMS, traditional fabrication methods such as milling and extrusion are replaced by processes with increased precision and resolution, such as photolithography, chemical etching, and plasma etching. MEMS fabrication utilizes processes adapted from the microelectronics industry, which were mainly developed for the handling and processing of silicon substrates (Madou 1997; Liu 2006). This sort of manufacturing consists of multiple steps of deposition and etching of structural (usually silicon) and sacrificial (usually oxide) thin films. At the end of the process, the sacrificial material is removed to enable the structural parts to move relative to the substrate. One simple MEMS fabrication method is the direct etching of silicon on insulator (SOI) wafers. SOI wafers are standardized stacks composed of a device structural layer on the top, an oxide sacrificial layer in the middle, and a handle substrate layer at the bottom. The SOI MEMS fabrication process, illustrated in Fig. 1, can be roughly summarized into two main steps: (1) etching of the device layer, after it is patterned using photolithography; and (2) partial removal of the oxide layer granting motion to the structural parts, which remain connected to the substrate through the oxide that is not etched away (the anchors). The addition of electrical contacts to the fabrication flow allows the induction of motion by the application of electrical voltages. Likewise, measurements of voltage changes can be used to gage MEMS motion.

2.2 Sensing and Driving Methods

There are several techniques used to provide or detect microscale displacements in MEMS. The most common operating principles include electrostatic, electrothermal, piezoelectric, and piezoresistive (Liu 2006). In the great majority of MEMS devices, energy conversion involves an input or an output electrical signal, typically a voltage difference. The electrostatic and electrothermal phenomena, which produce

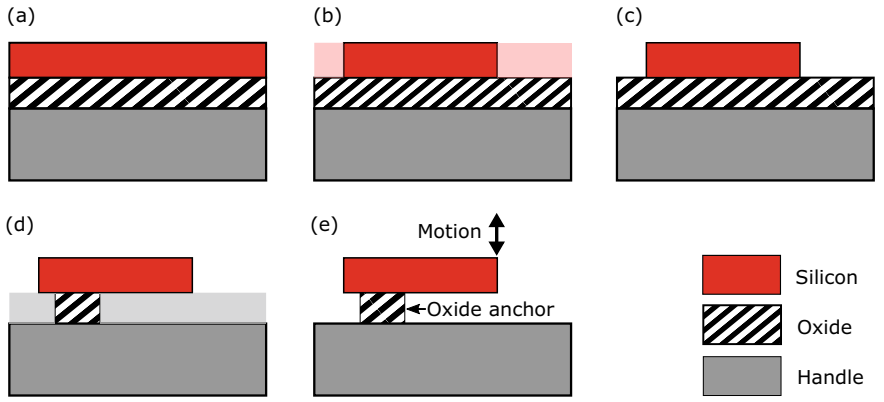


Fig. 1 **a** Initial SOI wafer. **b** Parts of the device layer are selected to be etched away after photolithography. **c** SOI wafer after the etching of the device layer. **d** Oxide areas to be removed in a selective etching procedure. **e** Final device after the oxide removal; the remaining silicon structure is anchored to the handle and is free to flex or move relative to the handle

forces that are usually negligible in conventionally sized mechanisms, are the most traditional configurations for driving and sensing in MEMS. Electrothermal MEMS, for example, produce motion through the thermal expansion of structures (usually beams) caused by Joule heating due to the application of voltage (Lai et al. 2004). In the case of electrostatic MEMS, motion is induced by electrostatic forces between microelectrodes separated by a small gap (Batra et al. 2007). Alternatively, changes in the gaps caused by an external force can be measured by the capacitance change between the electrodes.

MEMS accelerometers, some of the most commercially successful MEMS devices, may present a large variety of design types and working principles (Yazdi et al. 1998). Typically, external inertial forces displace an inertial mass that is suspended by compliant springs. This motion is then converted to an electrical signal that is proportional to the magnitude of this displacement. The transduction principle is usually capacitive (electrostatic) or piezoresistive (changes in the electrical resistance due to mechanical deformations). MEMS accelerometers can detect in-plane or out-of-plane forces depending on their design configurations (Fig. 2). Planar accelerometers commonly use an interdigitated configuration in order to increase the total capacitance and therefore the electrostatic sensitivity of the sensor. Higher sensitivity can also be achieved by reducing the accelerometer's natural frequency, which could be done by diminishing the suspension's stiffness. However, this also reduces the frequency response (bandwidth) of the sensor. Another practice to increase the sensitivity is to increase the signal-to-noise ratio by reducing the system's damping. This is usually done by etching holes along the proof mass or by operating the device under vacuum.

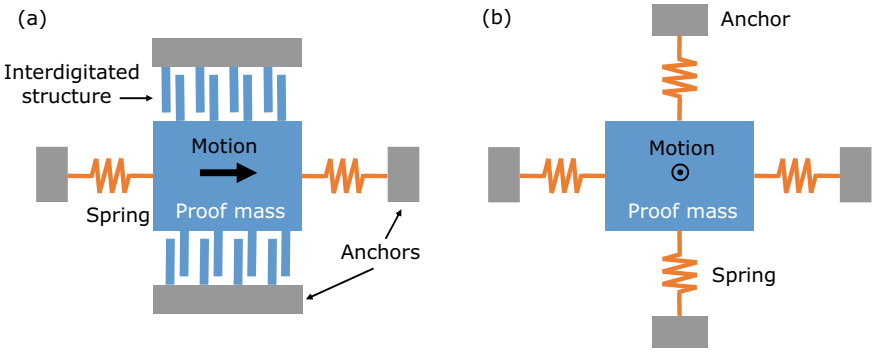


Fig. 2 Schematic illustration of two possible configurations of MEMS accelerometers: **a** the proof mass moves laterally, enabling the detection of in-plane accelerations, and **b** the proof mass moves up and down allowing the device to sense out-of-plane accelerations

2.3 MEMS Dynamics and Nonlinearity

MEMS devices are frequently designed to work in their dynamic regime, as it happens for example in MEMS resonators. As vibrating structures, MEMS exhibit much higher resonance frequencies compared to non-miniaturized mechanisms. This is because of their much higher k/m ratio, where k is the device elastic constant and m is its total mass. In MEMS resonators, shifts in the resonance frequency can be used to detect changes of different physical quantities, enabling the manufacturing of a variety of sensors such as pressure, force, and temperature sensors (Tilmans et al. 1992). The resonance frequency of MEMS tends to be very well defined (small bandwidth) due to their typically large quality factor (Q), which is a measure of the energy dissipation of oscillating structures. High values of Q indicate low energy dissipation, which leads to lower energy consumption, higher sensitivity, and lower noise. Energy dissipation can be classified as intrinsic or extrinsic (Ekinci and Roukes 2005). The former is associated with losses due to the material microstructure while the latter is mainly related to losses induced by the media surrounding the device. Extrinsic damping effects such as drag forces or squeezed films (when structures are too close) are usually the dominant sources of energy dissipation.

Another observed characteristic of MEMS resonators is their nonlinearity. Micromechanical oscillating structures demonstrate nonlinear behavior when driven above a certain critical amplitude (Husain et al. 2003; Ekinci and Roukes 2005). Frequently, the Duffing equation for nonlinear oscillators is used to describe the motion of MEMS resonators. Essentially, when oscillating at very large amplitudes (above critical), changes in the structure's stiffness result in nonlinear shifts of the resonance frequency. In the case of a clamped–clamped microbeam (i. e. both ends anchored) vibrating in its flexural mode, large driving amplitudes generate tensile forces that increase the beam stiffness resulting in an increase of its resonance frequency (Tilmans et al. 1992). The onset of nonlinearity in microstructures has been

explored elsewhere (Buks and Yurke 2006; Tadokoro et al. 2018). In this study, the nonlinearity of a clamped–clamped microbeam is used to set up a reservoir computing system able to perform non-trivial computing tasks.

3 Driven Oscillators with Duffing Nonlinearities

The Duffing model was first introduced to describe the hardening spring effect observed in mechanical systems (Duffing 1918). It is considered as one of the most common models used to describe the jump phenomenon observed in highly deformed mechanical resonators, where a slight change of forcing frequency leads to an abrupt discontinuous change in the steady-state amplitude (Guckenheimer and Holmes 2002; Kalmar-Nagy and Balachandran 2011). It keeps a simple mathematical form and accepts, under some approximations, analytical solutions (Ali 1995; Worden 1996).

3.1 Duffing Oscillator

Several micromechanical structures behave as nonlinear systems for high levels of excitation (Ekinci and Roukes 2005; Zaitsev et al. 2012). The Duffing equation with damping and external harmonic forcing is

$$\ddot{x} + \frac{\omega_0}{Q}\dot{x} + \omega_0^2 x + \beta x^3 = A\cos(\Omega t), \tag{1}$$

where x, t, ω_0, Q, A, Ω, and β are the displacement, time, undamped angular frequency, quality factor, excitation amplitude, angular excitation frequency, and cubic stiffness parameter, respectively. Dots denote derivatives with respect to time. As can be seen, Eq. (1) reduces to the forced damped linear oscillator when the anharmonic term is ignored ($\beta = 0$). An approximative solution for the position $x(t)$ can be obtained for small ω_0/Q, β, and A values and assuming the forcing is close to resonance, with $\Omega - \omega_0$ also small. Equation (1) can then be viewed as a perturbation of the autonomous harmonic oscillator. The perturbation technique known as "averaging" gives an approximative steady-state solution $x(t) = r\cos(\Omega t + \phi)$ where r is the oscillation amplitude and ϕ is the phase (see Guckenheimer and Holmes 2002 or Jan 2007 for details). Averaging gives a frequency response curve (Jan 2007),

$$\left(-2\omega_0\left(\Omega - \omega_0\right)r + \tfrac{3}{4}\beta r^3\right)^2 + 4\left(\omega_0^2/Q\right)^2 r^2 - A^2 = 0, \tag{2}$$

which can be solved for r.

Figure 3 shows the frequency response curve for $\beta = 0$ (from the exact solution of the linear problem) and curves from averaging for $\beta = \pm 0.05\text{Hz}^2/\text{m}^2$. The introduc-

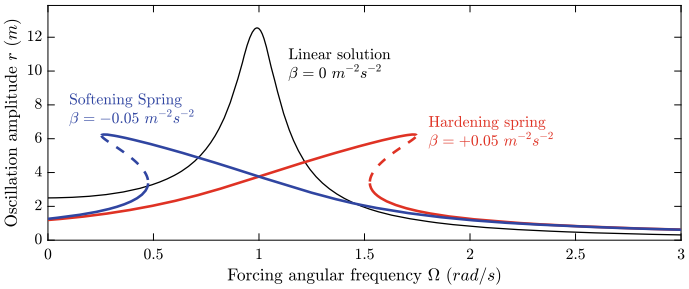


Fig. 3 Amplitude–frequency response curves for the linear system ($\beta = 0$ Hz^2/m^2) from the exact solution and by averaging for stiffness parameter $\beta = \pm 0.05$ Hz^2/m^2. Stable and unstable solutions are denoted as solid and dashed lines, respectively. The parameters used to construct these curves were $A = 2.5$m/s^2, $Q = 5$ and $\omega_0 = 1$ rad/s

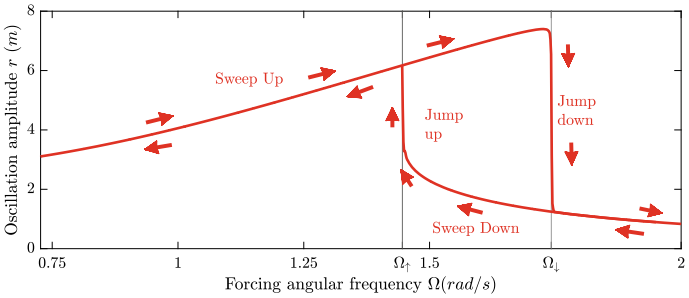


Fig. 4 Amplitude–frequency response curve obtained numerically by a sweep up followed by a sweep down of Ω; the jump and the hysteresis are apparent. The parameter values used to construct these response curves were $A = 2.5$ m/s^2, $Q = 5$, β =0.05m^{-2} s^{-2}, and $\omega_0 = 1$ rad/s

tion of the cubic nonlinearity tilts the curve to the right for $\beta > 0$ (hardening spring) and to the left for $\beta < 0$ (softening spring). Furthermore, close to the peak, there are three possible solutions for a given Ω (two stable ones and an unstable one, denoted as a dashed line). Figure 4 shows numerical solutions to Eq. 1 for β =0.05m^{-2} s^{-2}, as the forcing angular frequency Ω is swept up and down. Once Ω is increased above the angular frequency of the peak $\Omega_\downarrow$, the oscillation amplitude abruptly jumps to the lower branch, which is the only remaining solution. As Ω is reduced again, the oscillation amplitude follows the lower stable branch and jumps back to the upper branch once it reaches the unstable solution, at $\Omega_\uparrow$. Since $\Omega_\downarrow > \Omega_\uparrow$, the nonlinear system exhibits hysteresis.

Figure 5 shows the phase-space plot of three distinct motion regimes. For low forcing amplitudes or when the anharmonic term is not taken into account in the Duffing equation (1), the motion of the resonator resembles a linear harmonic device where the response in phase-space is an ellipse. At intermediate forcing, the system can have more complex dynamics due to the stiffening characteristic of the resonator: there can be more than one harmonic component in the oscillator motion, as studied

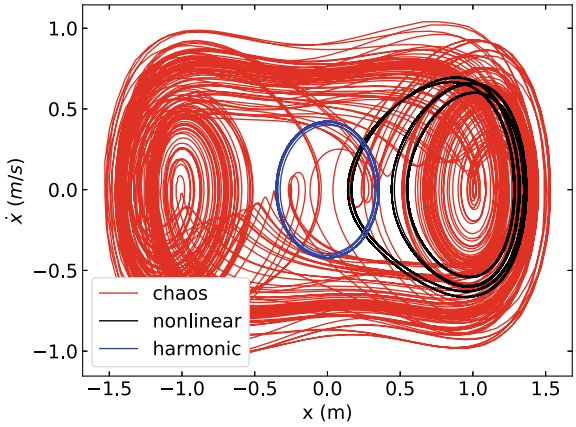


Fig. 5 Phase-space plots obtained from Eq. (1) for three motion conditions: harmonic oscillations with weak forcing $A = 0.2\text{m/s}^2$ and cubic stiffness corresponding to zero (blue line), moderate forcing $A = 0.29\ \text{m/s}^2$ with $\beta = 1\ \text{Hz}^2/\text{m}^2$ (black line), and chaotic oscillations at high forcing level $A = 0.5\ \text{m/s}^2$, $\beta = 1\ \text{Hz}^2/\text{m}^2$ (red line). The other parameters used to construct these curves were $Q = 3.33$, $\omega_0 = 1$ rad/s, and $\Omega = 1.2$ rad/s

in Kalmar-Nagy and Balachandran (2011). Large forcing amplitudes lead to a chaotic motion and the system becomes very sensitive to the initial conditions.

For nonlinear Duffing systems, sudden jumps in the resonance response are observed, as in Fig. 6. The jump frequency depends on the direction of the frequency sweep and the type of nonlinearity (softening or stiffening) (Malatkar and Nayfeh 2002). For lightly damped Duffing oscillator, Brennan et al. presented a simple approximated non-dimensional expression which gives the maximum oscillation amplitude $r_{\max}$ at the jump frequency $\Omega_{\downarrow}$ (Brennan et al. 2008). The relationship between the jump-down frequency and the cubic stiffness can be written in a dimensional form as (Tang et al. 2016)

$$\Omega_{\downarrow}^2 = \frac{3}{4}\beta r_{max}^2 + \omega_0^2. \tag{3}$$

Solving for r_{max} gives the so-called "backbone curve" presented by the dashed line in Fig. 6. It can be used to predict the frequency response of the system (Cammarano et al. 2014; Arroyo and Zanette 2016).

3.2 *Clamped–Clamped Beams*

A clamped–clamped beam is an oscillator exhibiting an anharmonic behavior at higher excitation amplitudes. Multiple studies have demonstrated that the Duffing

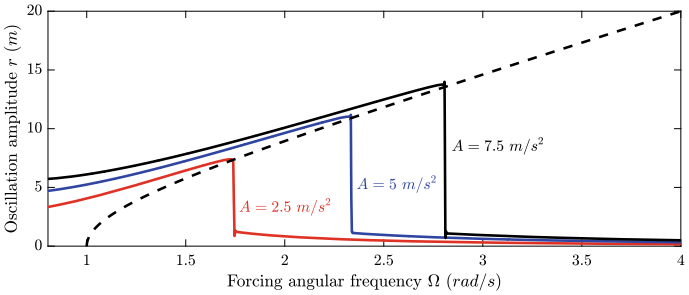


Fig. 6 Example of backbone curve (dashed line) for a Duffing oscillator swept up in forcing frequency. The parameter values used to construct these response curves were $Q = 5$, $\beta = 0.05\ \mathrm{Hz}^2/\mathrm{m}^2$, and $\omega_0 = 1$ rad/s

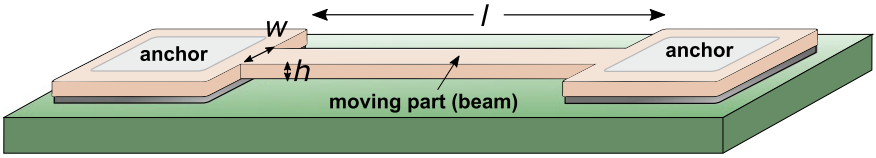


Fig. 7 Schematic description of a clamped–clamped beam: l, w, and h correspond to the length, width, and thickness of the beam

model may describe the nonlinear behaviors observed in the beam dynamics (Verbridge et al. 2006; Antonio et al. 2012; Abdolvand et al. 2016).

Figure 7 depicts a simplified schematic of a clamped–clamped structure.

3.2.1 Linear Analysis

The mass–damper–spring system represents the simplest model used to describe the linear resonator motions. It corresponds to Eq. (1) for which the nonlinear term β is null. The damper is associated here with energy losses in the system. The fundamental frequencies of excited clamped–clamped beam can be determined by solving the differential equation from Euler–Bernoulli beam theory. We assume that the beam deflection follows the fundamental mode vibration. The expression of the undamped resonance frequency for a clamped–clamped beam subjected to a lateral surface excitation can then be written as (Tilmans et al. 1992; Bao 2005)

$$\omega_0 = \frac{\lambda^2}{l^2}\sqrt{\frac{EI}{\rho wh}}, \tag{4}$$

where I, E, ρ, l, w, and h are quadratic moment, Young's modulus, mass density, length, width, and thickness of the beam, respectively. λ is a constant satisfying $\cosh(\lambda)\cos(\lambda) = 1$. Equation (4) indicates that the resonance frequency is closely

related to the mechanical structure geometry. It corresponds, for instance, to 389 kHz for a 300 μm silicon beam with a width and a thickness of 4 μm and 10 μm, respectively ($\lambda = 4.73$ in that case).

3.2.2 Nonlinearity Effects

In a clamped–clamped beam, the nonlinear parameter caused by the elongation of the beam can be approximated from (Postma et al. 2005)

$$\beta = \frac{E}{18\rho}\left(\frac{2\pi}{l}\right)^4. \tag{5}$$

For example, the calculated nonlinear coefficient is equal to $7.75\text{x}10^{23}(\text{Hz/m})^2$ using Eq. (5) for a 300 μm silicon beam.

To better understand the nonlinear dynamics of a clamped–clamped beam, a finite element modeling using the ANSYS software (Theory Reference for the Mechanical 2017) was developed. Figure 8a) presents a deformed silicon beam in its fundamental mode. The anchors, substrate, and gages are also considered in the simulation. An initial modal simulation is used to identify the resonance modes of the beam. Using an explicit time analysis, the system is then excited in the proximity of a resonant peak by a time-varying lateral force applied in the middle of the beam. This analysis takes into account the nonlinear phenomena induced by large geometrical deformations and the mechanical dissipation that occurs during the structure motion.

The simulation results are depicted in Fig. 8b). We first note that the "hardening" phenomenon, characteristic of Duffing oscillator, is present. Unlike the symmetric response in the linear case, the peak amplitudes shift to the higher frequencies when the excitation force increases. The jumps are also observed. The cubic stiffness

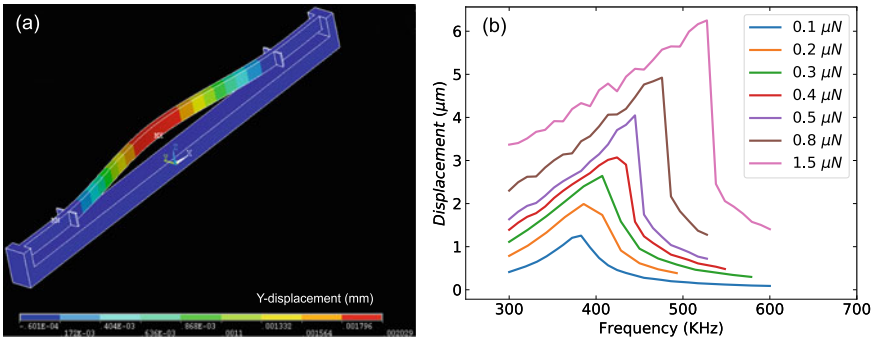


Fig. 8 **a** Displacement mapping of 300 μm clamped–clamped silicon beam obtained by ANSYS modal analysis. **b** Results of the analysis of transient finite elements on the clamped–clamped beam for different force amplitudes. The width w and thickness h of the beam were 4 μm and 10 μm, respectively

parameter can be determined from a fit to Eq. (3) and is equal to (1.87 ± 0.26) x 10^{23} $(\text{Hz/m})^2$. This result is similar to the one obtained theoretically (Eq. (5)).

3.2.3 Damping Effects

The energy dissipation mechanisms of the mechanical system are associated with damping effects. The parameter indicating the damping and the efficiency of the resonator systems, the so-called quality factor Q, can be defined as the ratio of dissipated energy per period, Δ, to the energy stored in the oscillator (here, $kr^2/2$) (Tilmans et al. 1992; Bao and Yang 2007)

$$Q = 2\pi \times \frac{kr^2/2}{\Delta}. \tag{6}$$

Figure 9 depicts the amplitude–frequency curve for three damping conditions using numerical Duffing solutions (Eq. (1)). The larger damping effect corresponds to the smaller factor (black line) while the peak amplitude is higher for smaller damping (blue line). Note that the peak amplitude would be infinite in the absence of damping.

There are several sources of damping in mechanical structures. A quality factor Q_i can be attributed to each dissipation mechanism. The total quality factor Q can be written as Matthiessen's rule (Matthiessen and Vogt 1864; Naeli and Brand 2009)

$$\frac{1}{Q} = \sum_i \frac{1}{Q_i}. \tag{7}$$

The extrinsic damping caused by the surrounding air can often be ignored for conventional mechanical systems. However, as air damping is related to the surface area of the resonator, viscous air damping can be significant for micromechanical

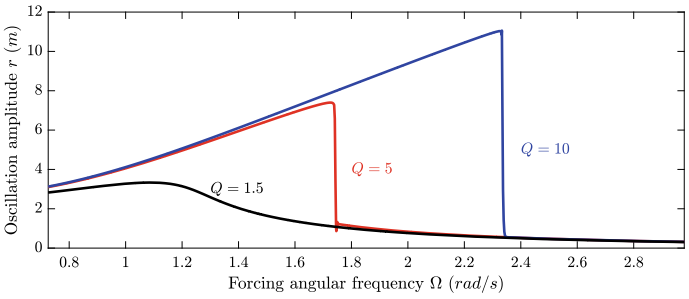


Fig. 9 Effect of damping on the amplitude–frequency response curve: small damping (blue line), intermediate damping (red line), and high damping (black line). The parameter values used to construct these response curves were $A = 2.5\text{m}/rms^2$, $\beta = 0.05$ Hz^2/m^2, and $\omega_0 = 1$ rad/s

devices. The first damping mechanism highlighted is the drag force. It represents the effect caused by the surrounding gas on the resonator when the beam is far away from any surrounding object. From Naeli and Brand (2009), the quality factor describing gas dissipation in microbeams is

$$Q_d = \frac{\rho h w \omega_0}{3\pi \left(\mu + w\sqrt{\rho_a \mu \omega_0/16}\right)}, \tag{8}$$

where μ is the air dynamic viscosity and ρ_a is the air mass densities. This factor can be reduced experimentally by placing mechanical devices under vacuum (Tilmans and Legtenberg 1994; Gui et al. 1995).

A driving electrode must be close to the beam in order to electrostatically drive the mechanical resonator. If the gap d between the beam and the electrode is small compared to the beam thickness h, the main damping mechanism is the "squeezed-film effect" due to the incompressible character of the gas. This is all the more important when the gap is reduced. The corresponding analytical expression of squeezed-film damping is (Starr 1990; Bao 2005)

$$Q_s = \frac{\rho w d^3 \omega_0}{\mu h^2}. \tag{9}$$

For a silicon beam with $(w, h, l) = (4, 10, 300)$ μm, where the gap d corresponds to 6 μm, one has $Q_d = 529$ and $Q_s = 2740$. From Eq. (7), the combined quality factor Q is then 457. For additional effects comprising, for instance, the thermoelastic mechanism, we refer the reader to Verbridge et al. (2006), Naeli and Brand (2009), and Younis (2010). Note that the anchors in the clamped–clamped beams can also have a significant effect on the dynamics of the resonator (Lee et al. 2008; Naeli and Brand 2009).

4 Reservoir Computing in a MEMS

As highlighted in the previous sections, MEMS technology can reliably produce small and energy-efficient devices exhibiting rich dynamical behaviors often not accessible for mechanical structures at larger scales. Exploiting these dynamics for neuromorphic hardware thus seems a promising alternative to computing using conventional electronics, which keep struggling with power dissipation issues. As a result, the following section explores the use of a micromachined clamped–clamped silicon beam as the single dynamical node of a delay-coupled reservoir computer trained to perform simple classification tasks.

4.1 The MEMS Nonlinear Node

Construction of a hardware reservoir computer (RC) begins with the choice of a suitable physical node, which should have a nonlinear activation function in order to be able to model nonlinear processes. The stiffening Duffing behavior of a clamped–clamped silicon beam oscillating at large amplitudes can provide the nonlinearity in MEMS RC. An order of magnitude for the minimum oscillation amplitude to obtain sufficient nonlinear behavior is the amplitude r_c associated with the onset of bistability (Lifshitz and Cross 2010):

$$r_c = \left(\frac{4}{3}\right)^{3/4} \sqrt{\frac{\omega_0^2}{Q\beta}}. \tag{10}$$

For the beam studied in this section, the onset of the nonlinearity is around $r_c = 150\,\text{nm}$.

The beam shown in Fig. 10 was microfabricated on a (100) silicon on insulator (SOI) substrate with a nominal resistivity of $(0.003 \pm 0.002)\ \Omega$ m and a sacrificial oxide thickness of $1.5\,\mu\text{m}$. It has a length of $L = 500\ \mu\text{m}$, a width of $w = 10\,\mu\text{m}$, corresponding to the SOI device layer thickness, and an in-plane thickness (normal to its displacement) of $h = 4\ \mu\text{m}$. The device was wirebonded to a chip carrier and placed in a Faraday cage for the experiments, but was otherwise unpackaged. This lack of proper packaging makes the beams sensitive to dust in their environment, which has the undesirable effect of modifying their resonant frequency over time. For instance, one beam has had its natural frequency lowered by as much as 20% over the course of one year. The experimental quality factor of the MEMS was 167 $\pm$ 2. This value, which is independent of the oscillation amplitude, is comparable to the analytical value of 204 obtained using Eqs. 7–9 for the nominal dimensions of the beam. Fabrication tolerances could account for this gap between the two values, as

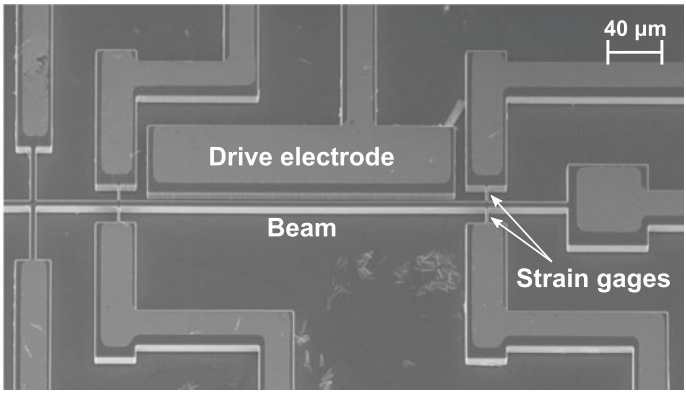


Fig. 10 SEM image of the MEMS

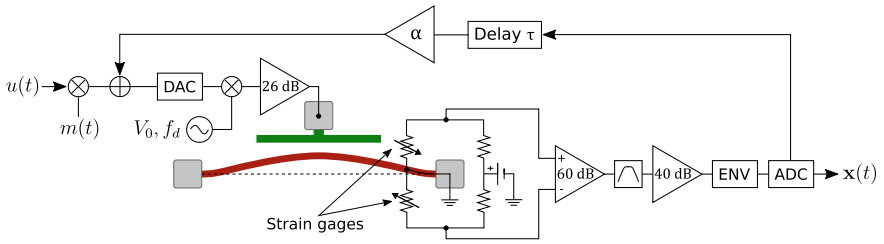


Fig. 11 Experimental setup for the reservoir computer. The masking procedure as well as the delayed feedback loop are implemented in the digital domain, while the post-processing (extraction of the displacement amplitude) is carried out through custom analog electronics

well as other dissipation mechanisms such as anchor loss and the proximity of the substrate. In the linear regime, the beam naturally oscillated at $f_0 = 155\,\text{kHz}$, compared to a calculated value of 144.2 kHz (Eq. 4), although the maximum of the frequency response shifted to higher frequencies as the drive amplitude was increased, a behavior which corresponds to a stiffening Duffing oscillator. The Duffing parameter for the beam shown in Fig. 10 was estimated to $1.9 \times 10^{23}\ \text{Hz}^2\text{m}^{-2}$ by adjusting Eq. 3 to experimental data of the beam's response. Equation 5 yields a comparable value of $1.1 \times 10^{23}\ \text{Hz}^2\text{m}^{-2}$.

Among the plethora of possible transduction methods presented in Sect. 2.2, an appropriate choice for RC MEMS is to drive the beam electrostatically and sense its displacement piezoresistively. By polarizing a 300 µm long drive electrode placed 6 µm away from the beam in Fig. 10 with a voltage signal of the form $V_d(t) = V_0 \cos(2\pi f_d t)$, a force $F_d \propto V_d^2(t) = \frac{V_0^2}{2}(1 + \cos(4\pi f_d t))$ can be applied between the beam and the fixed electrode such that vibrations of the beam are solicited at twice the input voltage frequency f_d. The piezoresistive transduction of the beam motion to an electrical signal, carried out through 12 µm long by 1.2 µm wide piezoresistive strain gages patterned on the device, was chosen for its linearity (to ensure that nonlinear mapping comes exclusively from the beam's displacement) and sensitivity (transduction coefficient of $\sim 10^2$ V/m). Two external resistors were combined with the two piezoresistive gages, as illustrated in Fig. 11, to form a Wheatstone bridge, allowing for a differential measurement of the beam's motion. Compared to a single-ended measurement, the differential configuration has the advantage of reducing the system sensitivity to noise in the DC voltage source polarizing the Wheatstone bridge, but more importantly, it also cancels the feedthrough drive signal at the readout. This unwanted signal is symmetrically coupled to both readout points (ends of the piezoresistive gages) through parasitic capacitors (much larger than the ~10 fF capacitor formed by the beam and drive electrode) present in the device, while the displacement signal is of opposite sign in each branch (one gage stretches when the other gets compressed), so only the latter gets amplified by the instrumentation amplifier. The differential input stage is followed by a bandpass filter with a bandwidth of 80 kHz to further reduce the noise contribution, and a second amplification stage brings the displacement signal, initially of a few tens of μV, to a

level suitable for the envelope detection stage that follows. This last step produces an appropriate output by extracting the amplitude of the beam displacement signal, yielding a signal-to-noise ratio (SNR) of 35 dB, essentially limited by the Johnson noise generated by the resistor bridge.

4.2 Training with Delayed Feedback

The use of a single physical node (Appeltant et al. 2011) greatly simplifies hardware implementation of an RC by drastically reducing the number of structures to couple physically, drive, and measure, with the main drawbacks of requiring a more refined preprocessing scheme and a serialization of the network (and thus of the computation). Indeed, since a single physical node is available, the reservoir consists of a virtual network created by time-division multiplexing of the input signal. While a space-coupled network would possess a multitude of physical nodes (typically $\sim 10^2$) coupled in space and use the ring-down time of the oscillators as a form of memory (the behavior of the oscillators depends on their history), a delay-coupled reservoir instead uses this decay time to couple adjacent virtual nodes in the time domain: the input signal is masked by a function of period τ, which in the simplest case is a function alternating randomly between two values after each time interval θ. τ is an integer multiple of θ which defines the number of virtual nodes ($N = \tau/\theta$). By choosing $\theta < T$, where $T = Q/(\pi f_0) = 330\ \mu$s is the decay time of the oscillator, the beam response during a given interval θ depends on its response during previous intervals. Since the oscillator decay time T is much shorter than the characteristic time τ of the input, the reservoir activation does not persist between two timesteps of the input signal, and the virtual network requires an additional feedback loop in order to have access to some form of memory. A feedback signal is thus added, with a delay τ and gain α, to the input for the next timestep. As a result, a given virtual node is driven by a superposition of the (masked) input and of its response to the input from the previous timestep:

$$V_d(t) = V_0\left[u(t)m(t) + \alpha x(t-\tau) + 1\right]\cos\left(2\pi f_d t\right), \tag{11}$$

where $x(t)$ is the displacement amplitude signal at time t, $m(t)$ is the temporal mask, and $u(t)$ is the input signal.

The nonlinear nature of the beam's amplitude response (Dion et al. 2018) guided the choice of amplitude modulation of the sinusoidal pump for the RC input. In the case of a Duffing oscillator, the nonlinearity can be tuned to a certain extent by adjusting the drive frequency. The resulting system is schematized in Fig. 11. The input $u(t)$ is first scaled so that it is restricted to the empirically determined range [0.60, 0.75], then it is sampled and held for a time τ and multiplied by the temporal binary mask of period τ and characteristic time θ. For the MEMS RC, optimization of the mask with respect to the RC success rate yielded mask values of 0.45 and 0.70. The result, $u(t) \times m(t)$, is used to modulate the amplitude of the sinusoidal pump

(Sect. 4.4.2 discusses adjusting the pump in more detail). Sampling the envelope (ENV) of the displacement signal at a rate θ^{-1} with an analog to digital converter (ADC) yields a vector $\mathbf{x}(t)$, containing the N virtual node states at timestep t. These values are then combined linearly to produce a scalar output:

$$y(t) = \mathbf{w}^T \mathbf{x}(t). \tag{12}$$

The goal of the training phase is to compute the appropriate vector $\mathbf{w}$ of weights by adjusting them so that the response of the RC to a series of training examples approximates as well as possible a known target $y'(t)$. If the task for which the RC is trained is to process a signal which changes at every time period τ, for instance, then a series of M training periods can be presented to the system, each with an input value $u_k = u(k\tau)$ for $k = 1, \ldots, M$, resulting in M outputs $y_k = y(k\tau)$ which can be compared to the desired outputs $y'_k = y'(k\tau)$ with the mean squared error

$$\frac{1}{M} \sum_{k=1}^{M} (y_k - y'_k)^2. \tag{13}$$

A similar mean squared error can be defined for the classification of input sequences of different lengths (with $y(t)$ sampled at the end of each input sequence).

The training process is done offline and consists in computing the vector $\mathbf{w}$ minimizing the mean squared error between $y(t)$ and $y'(t)$. The result is

$$\mathbf{w} = \mathbf{y}' \mathrm{X}^T \left(\mathrm{X}\mathrm{X}^T + \gamma \mathrm{I} \right)^{-1}, \tag{14}$$

where $\mathbf{y}'$ is the vector of desired outputs and X is a matrix with each row corresponding to the state $\mathbf{x}$ of the virtual nodes after one of the inputs u_k from the training set has been processed. γ is a regularization parameter that increases numerical stability and prevents overfitting. A value of $\gamma = 10^{-4}\ \mathrm{V}^2$ proved adequate for both benchmarks investigated below.

4.3 Performance Metrics

Following the training phase, it is customary to test the performance of the RC with inputs that were not part of the training set, so that the generalization capability of the RC can be assessed. In order to highlight its universal character, the MEMS RC discussed above was tested on two different benchmarks with the same set of hyperparameters: a network of $N = 400$ virtual nodes sampled every $\theta = 0.1$ ms with a feedback gain $\alpha = 1.1$ and a beam driven at $f_d = 80.3\,\mathrm{kHz}$, $V_0 = 72.5$ V, with the piezoresistive gages biased at 2.5 V.

4.3.1 Parity Benchmark

The parity benchmark is a conceptually simple task that can be nonlinear and requires memory. As such, it is well suited for a first evaluation of the system's performance. It consists of computing the parity of $n \geq 1$ successive input bits after an initial delay $\delta \geq 0$:

$$P_{n,\delta}(t) = \prod_{i=0}^{n-1} u\left(t - (i+\delta)\tau\right). \tag{15}$$

$P_{1,0}$ is linear and does not require memory, but for $\delta > 0$ or $n > 1$, the target depends on the history of the input signal, so the system must be able to store a transformed version of the input for a finite time. In this chapter, we will only report results with no delay, i.e., for $P_n = P_{n,0}$. For this task, the input $u(t)$ is a binary sequence randomly alternating between -1 and +1 at each time $t = k\tau$. It is thus first shifted and scaled to [0.60, 0.75] before being fed to the RC, as discussed in Sect. 4.2.

Figure 12a shows the RC output for this task overlaid on the target after a training phase of 2000 samples. The performance is quantified by comparing the signs of the prediction and of the target over the whole 2000 samples of the testing set. The accuracy of the classification is the same for P_2 to P_4 since the raw RC output is thresholded, but the trace is more noisy for P_4. By increasing n or δ, the complexity of the task is increased and this translates to a decrease in the prediction success rate. This performance drop can be counterbalanced up to a degree by increasing the number of nodes or the number of training samples, as evidenced by Fig. 14, or by a finer tuning of the nonlinearity (see Fig. 15). For the network of $N = 400$ nodes used to produce Fig. 12a, the mask period is $\tau = N\theta = 40$ ms, such that the bitstream is processed at a rate of 25 bits/s. On the other hand, a network of 10 virtual nodes is sufficient to process P_2 with less than 1% error, which leads to a classification rate of 10^3 bits/s. This means that for a given physical node with immutable characteristics, processing speed can be optimized for a specific task by adjusting the number of virtual nodes.

4.3.2 Spoken Digit Classification

With the same set of hyperparameters, the MEMS RC was also trained to classify the digits zero to nine spoken by sixteen different speakers, male and female, using the TI-46 dataset (Lieberman 1993). Since sounds have an inherent temporal dependence, this task seems well adapted to the RC approach, as evidenced by its predominance as a RC benchmark (Appeltant et al. 2011; Brunner et al. 2013; Coulombe et al. 2017; Dion et al. 2018; Duport et al. 2012; Larger et al. 2012, 2017; Martinenghi et al. 2012; Paquot et al. 2012; Soriano et al. 2015; Torrejon et al. 2017; Verstraeten 2005). Whether it is obtained through RNNs or by using other means, state-of-the-art performance for this task is usually accompanied by spectral preprocessing to model the human ear, such as the Mel-Frequency Cepstral Coefficients (MFCC) or

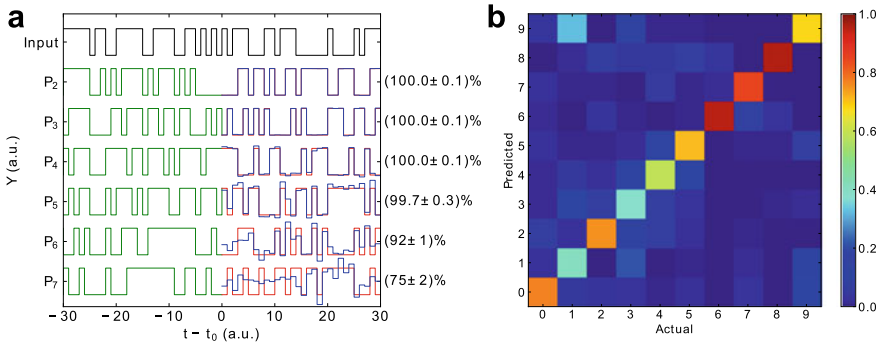


Fig. 12 **a** Performance for the parity benchmark. After a training phase (green), the RC response (blue) to the input (black) is compared to the target (red). **b** Confusion matrix for the spoken digit classification task. Colors indicate the probability that an input digit (columns) is assigned to a given class (rows) by the RC

the Lyon Passive Ear model (Lyon 1982). For this study, the preprocessing was kept minimal in anticipation of eventually interfacing the MEMS RC directly with sound pressure, as opposed to feeding samples from recorded waveforms. Each randomly selected utterance is first lowpass filtered 30 Hz and resampled at 60 samples/s, then it is normalized and scaled so that the complete sequence of waveforms is restricted to the range [0.60,0.75]. In order to save processing time, silences before and after the utterance are cropped, which results in an average of $\bar{\eta} = 29$ samples per word. After being masked as described in Sect. 4.2, those samples are then fed sequentially to the reservoir without any pause between them. The output of a given virtual node for a given utterance is then the mean of its responses over the whole utterance (i.e., $x_i = (1/\eta) \sum_{j=0}^{\eta-1} x\,(i\theta + j\tau)$ for node i). Ten output layers are trained for the same reservoir activation: one boolean classifier is used for each individual digit. Since there are ten different possible classes for this task, the length M of the training sequence was increased to 6000 utterances so that the RC is trained on a sufficient number of examples for each digit.

Figure 12b shows that the confusion matrix for this task is almost diagonal, although some phonetically similar digits such as “1” and “9” or “4” and “5” are more often misclassified by the RC. The global success rate is (70 ± 2) %, and slightly better performance (Dion et al. 2018) could be obtained by optimizing the hyperparameters with respect to this particular task. Despite the fact that the training procedure lasts a few hours, the trained 400 node RC processes words at a rate of 1 per second, fast enough so that one could envision using such a system for real-time speech processing.

4.4 Hyperparameter Optimization

Finding optimal parameters for successful reservoir computing can be a tedious task, as RC performance typically depends on the appropriate combination of multiple hyperparameter values. Moreover, these parameters cannot be tuned independently: modifying one of them can shift the optimal value of other parameters. Choosing a random set of parameters will most often result in no computational success at all, and the accuracy landscape may display multiple local minima, making gradient descent optimization impractical. A gridsearch may seem like a foolproof optimization method, but without any indication of the location of the success region, the search space is vast and of high dimensionality. Besides, the region of non-zero success can be limited to a rather narrow region, as will become apparent later in this section, so that if the gridsearch is too coarse, the optimal parameter set can be missed altogether. Expert knowledge is thus necessary to set bounds for the different parameters of the gridsearch in a principled way or to perform a manual search in order to find a starting point with non-trivial success for optimization. To circumvent this obstacle, different methods are investigated in the RC literature (Bala et al. 2018), such as using genetic algorithms (Dale et al. 2016; Ferreira and Ludermir 2009, 2011), particle swarm optimization (Zhou 2010; Sergio and Ludermir 2012; Jubayer Alam Rabin et al. 2013; Salah et al. 2017), differential evolution (Zhang et al. 2013; Rigamonti et al. 2018; Wang et al. 2018), or hybrid variants thereof which combine different metaheuristics.

Temporal traces of reservoir activation such as those presented in Fig. 13 can also guide the initial optimization. By detuning a single parameter such as the drive frequency f_d, the feedback gain α, or the virtual node duration θ, the traces for healthy and unhealthy reservoirs can be compared and a few empirical criteria for successful RC can be deducted. Such criteria include the dynamic range and saturation of the response and its correlation with the input signal.

The optimization of hyperparameters shown below was performed using the parity benchmark, as the total training and testing time is much lower than the spoken word recognition benchmark: a training example for parity is composed of a single sample, while a spoken digit utterance contains tens of samples to feed to the RC. Nevertheless, the resulting parameter set can be used as a starting point for optimization with respect to a different task.

4.4.1 Number of Training and Testing Samples, Reservoir Size

The number of examples used for testing is one parameter that can be chosen in a principled way. Its only effect is on the uncertainty of the performance measurement. Considering that for all the benchmarks investigated here the testing phase is a series of Bernoulli trials (i.e., is the sample correctly classified?), the precision of the obtained success rates can be quantified using a binomial proportion confidence interval, such as the Agresti–Coull interval (Agresti and Coull 1998). In this specific

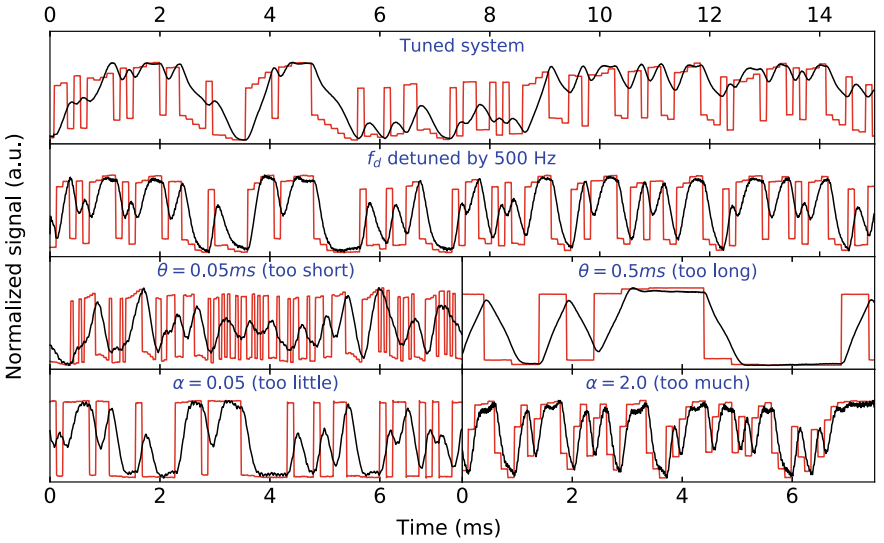


Fig. 13 Normalized drive signal envelope (red) and beam displacement amplitude (black) for a well-performing RC (top panel) and for various detuned configurations (lower panels). Note that the beam response is oversampled compared to normal operation where it is only sampled when the mask value is updated

case, the measurement error decreases as the number of trials and success rate are increased. A longer testing phase thus increases the measurement accuracy, but it also increases the acquisition time, making the results more susceptible to the effects of parameter drifts in the MEMS. This is where cross-validation becomes relevant: the training data can be reused for testing (and testing data for training), and thus not increase acquisition time but still get more measurement accuracy. A testing set of 2000 samples was deemed sufficient for the results presented here, as it is a good compromise between acquisition speed (∼3 min for one complete training and testing experiment) and accuracy (<2%).

Figure 14 shows the P_3 to P_6 success rate for different pairs of (N, M) values. For this task, the minimum length of the training set (M) insuring optimal performance increases with the number of virtual nodes (N) in the explored region, and the number of nodes needs to be increased as the complexity of the task is increased from P_3 to P_6 in order to keep a constant success rate. A narrow region, centered around $M = N$, seems to prohibit adequate results. This could be due to overfitting, since this region does not respect the rule of thumb stating that N should not exceed $M/10$ to $M/2$ (Jaeger 2002). Training another output layer on the same data with $\gamma = 10^{-2}$ V^2 (to reduce overfitting by increasing regularization) increases performance for $M = N$ but considerably degrades performance otherwise. Good performance is also possible in a region where $N > M$, although unless the training set is of limited size, it is advisable to choose $N < M$ as the speed and energy cost of increasing the number of nodes is generally higher than using a longer training phase.

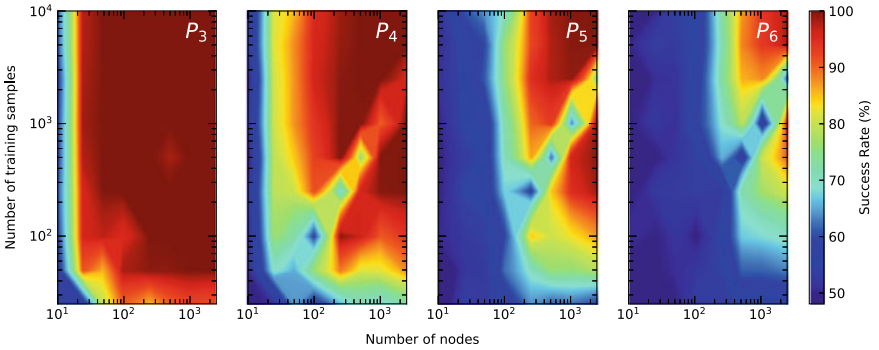


Fig. 14 Interpolated success rate in the number of nodes (N)—number of training samples (M) plane for P_3 to P_6 (left to right)

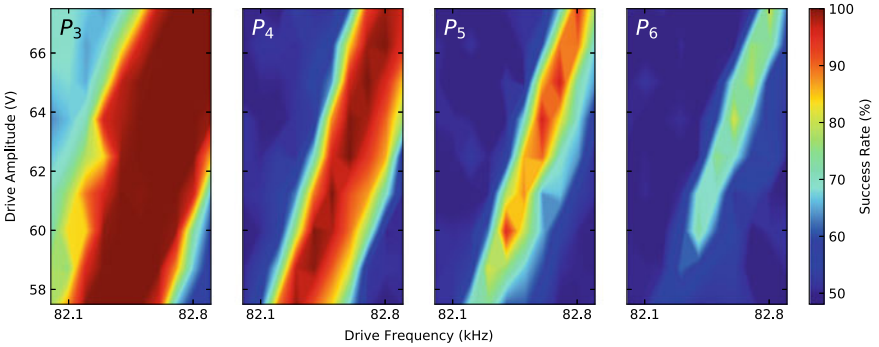


Fig. 15 Success rate in the drive frequency—drive amplitude plane for the P_3 to P_6 tasks. Note that this figure was produced earlier than the other figures when the oscillator had a slightly higher natural frequency. This slow drift of f_0 merely translates the features in this figure horizontally

4.4.2 Tuning the Nonlinearity

Figure 15 shows that good performance for P_3 to P_6 is limited to a rather narrow, tilted band in the drive frequency—drive amplitude plane. The more nonlinear task P_6 requires higher drive amplitudes for optimal success, corresponding to higher beam oscillation amplitudes and thus a more pronounced impact of the cubic term in the Duffing equation (Eq. 1). Figure 13 shows the effect of operating the system with the wrong combination of drive amplitude and frequency. At 500 Hz below the proper operating frequency, the dynamic range of the readout signal is reduced and its shape more closely resembles the input due to the more linear behavior of the beam. Such detuning can occur for example during the MEMS life if a large enough foreign particle gets attached to (or detached from) the beam, shifting its natural frequency.

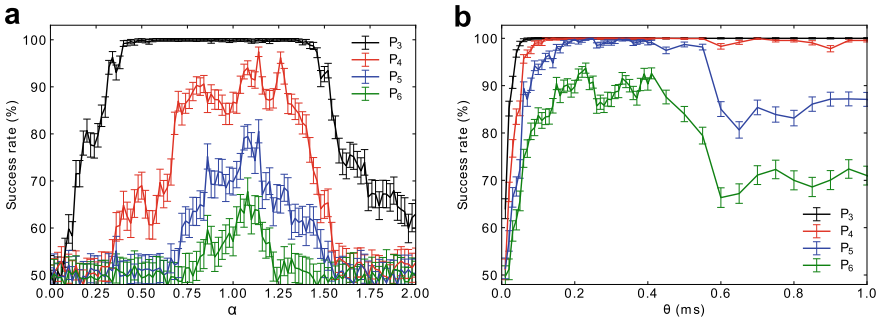


Fig. 16 The success rate for the parity function strongly depends on both the feedback gain α (**a**) and the mask update rate θ (**b**)

4.4.3 Feedback Strength

By plotting the success rate for the parity benchmark against the feedback strength α as in Fig. 16a, it can be seen that there is an intermediate value of α providing optimal results for all the investigated tasks. Below this value of $\alpha \simeq 1.1$, the system has less memory and success eventually vanishes at $\alpha = 0$. For values of α which are too large, the RC may not exhibit the fading memory property (Jaeger 2001) (or it may fade too slowly), and the system also tends to saturate (see bottom panel of Fig. 13), negatively impacting performances.

4.4.4 Coupling Strength

Figure 13 shows the effect of increasing or decreasing θ on the dynamics of the system. For $\theta = 0.05\,\text{ms} \ll T$, the dynamic range is limited: the beam cannot respond quickly enough to the rapidly alternating low and high mask bits, and only behaves appropriately when there is a succession of identical mask values. This translates into a lower correlation coefficient of 0.05 between the input and output amplitudes, compared to a correlation coefficient of 0.44 for the optimized RC. For the case $\theta = 0.5\,\text{ms} \gg T$, the response saturates as soon as there are two or more successive identical mask values, such that the readout (points sampled at the end of each period θ) essentially only visits two points of the transfer function (low level and high level saturation). The correlation coefficient is 0.60 and feedback has little effect, as the signal is less dynamical and more closely tied to the input due to the weak coupling between adjacent virtual nodes. The weak coupling regime ($\theta \lesssim T$), where a given virtual node state is only dependent on the state of its neighbor, is analogous to a linear chain of space-coupled oscillators.

Figure 16b shows the success rate for P_3 to P_6 as a function of θ, which essentially controls the connectivity matrix of the reservoir. While using a value of $\theta = 0.2$ ms gives slightly better results, a virtual node duration of $\theta = 0.1\,\text{ms} \simeq T/3$ was used

for the results presented here as the computation is two times faster ($\sim$2 min). For higher values of θ, the longer acquisition time increases the effect of medium-term drifts in the system on the results: optimal weights may evolve over time but our offline training method doesn't allow adapting them through the acquisition.

5 Conclusion

MEMS devices form the basis of many of today's sensor technologies and are expected to play an important role in the development of new technologies related to artificial intelligence and machine learning, in the context of producing "big data" from autonomous systems (e.g., self-driving cars) or distributed sensor systems (e.g., the Internet of Things). We have presented in this chapter key concepts for using MEMS to construct neuromorphic computing devices, as well as key experimental results showing that reservoir computing can be implemented efficiently and robustly in MEMS. As MEMS can be small, energy-efficient, and function at high speeds, they could constitute a very attractive hardware substrate for unconventional AI computing. When used as "pure" computing devices (with an analog electrical input and an analog electrical output), they could implement AI functionalities with performance levels exceeding those of conventional electronics (Coulombe et al. 2017). Perhaps more interestingly, our MEMS devices can implement both neuromorphic computing *and* sensing functionalities in the same device. This is a fairly new idea, which could bring significant gains in system size and energy consumption through integration: instead of building mechatronic systems with a discrete sensor coupled to separate signal processing electronics, one could envision building a trainable sensor which exploits the nonlinearity of its sensing mechanism to implement computing functions on the measured data. We are developing this idea in MEMS, but similar ideas might also be relevant for optical sensors and RC systems, for instance.

Deep learning, as the most productive line of research for artificial intelligence today, relies on training complex systems (artificial neural networks) using large amounts of data. The separation between data generation and data processing has traditionally been very clear in such deep neural networks. One might however consider the example of biological brains, which actually integrate the sensing and computing functionalities in some sensory neurons (Pitkow 2015), perhaps as a strategy to increase efficiency, robustness, or adaptiveness. Nature might have discovered long ago that such integration was an effective way to build faster, smaller, and more energy-efficient intelligent biological systems, which are able to respond efficiently to sensory inputs collected from their environment (i.e., systems which are sophisticated integrated sensing *and* computing devices).

References

R. Abdolvand, B. Bahreyni, J. Lee, F. Nabki, Micromachined resonators, a review. Micromachines **7**(9), 160 (2016)

A. Agresti, B.A. Coull, Approximate is better than "Exact" for interval estimation of binomial proportions. Am. Stat. **52**(2), 119 (1998)

G. Ananthasuresh, *Micro and Smart Systems: Technology and Modeling* (Wiley, Hoboken, 2012)

D. Antonio, D.H. Zanette, D. Lopez, Frequency stabilization in nonlinear micromechanical oscillators. Nat. Commun. **3**(1) (2012)

L. Appeltant, M.C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C.R. Mirasso, I. Fischer, Information processing using a single dynamical node as complex system. Nat. Commun. **2**, 468 (2011)

S.I. Arroyo, D.H. Zanette, Duffing revisited: phase-shift control and internal resonance in self-sustained oscillators. Eur. Phys. J. B **89**(1) (2016)

A. Bala, I. Ismail, R. Ibrahim, S.M. Sait, Applications of metaheuristics in reservoir computing techniques: a review. IEEE Access **6**, 58012–58029 (2018)

M. Bao, *Analysis and Design Principles of MEMS Devices*, 1st edn. (Elsevier, Amsterdam, 2005). OCLC: 254583926

M. Bao, H. Yang, Squeeze film air damping in MEMS. Sens. Actuators A **136**(1), 3–27 (2007)

B. Barazani, G. Dion, A. Idrissi-El Oudrhiri, F. Ghaffari, J. Sylvestre, Micromachined neuro-processing accelerometer. To appear in *27th Canadian congres of Applied Mechanics*, vol. 3 (2019)

B. Barazani, G. Dion, J.-F. Morissette, L. Beaudoin, J. Sylvestre, M. Neuroaccelerometer, Integrating sensing and reservoir computing in MEMS. J. Microelectromech. Syst. **29**(3), 338–347 (2020)

R.C. Batra, M. Porfiri, D. Spinello, Review of modeling electrostatically actuated microelectromechanical systems. Smart Mater. Struct. **16**(6), R23–R31 (2007)

M.J. Brennan, I. Kovacic, A. Carrella, T.P. Waters, On the jump-up and jump-down frequencies of the Duffing oscillator. J. Sound Vib. **318**(4–5), 1250–1261 (2008)

D. Brunner, M.C. Soriano, C.R. Mirasso, I. Fischer, Parallel photonic information processing at gigabyte per second data rates using transient states. Nat. Commun. **4**, 1364 (2013)

E. Buks, B. Yurke, Mass detection with a nonlinear nanomechanical resonator. Phys. Rev. E **74**(4) (2006)

A. Cammarano, T.L. Hill, S.A. Neild, D.J. Wagg, Bifurcations of backbone curves for systems of coupled nonlinear two mass oscillator. Nonlinear Dyn. **77**(1), 311–320 (2014)

J.C. Coulombe, M.C.A. York, J. Sylvestre, Computing with networks of nonlinear mechanical oscillators. PLOS ONE **12**(6), e0178663 (2017)

M. Dale, S. Stepney, J.F. Miller, M. Trefzer, Reservoir computing in materio: an evaluation of configuration through evolution, in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, December 2016, Athens, Greece (IEEE, 2016), pp. 1–8

G. Dion, S. Mejaouri, J. Sylvestre, Reservoir computing with a single delay-coupled non-linear mechanical oscillator. J. Appl. Phys. **124**(15) (2018)

G. Duffing, Erzwungene Schwingungen bei veranderlicher Eigenfrequenz und ihre technische Bedeutung. Brunswick (1918)

F. Duport, B. Schneider, A. Smerieri, M. Haelterman, S. Massar, All-optical reservoir computing. Opt. Express **20**(20), 22783 (2012)

K.L. Ekinci, M.L. Roukes, Nanoelectromechanical systems. Rev. Sci. Instrum. **76**(6) (2005)

A.A. Ferreira, T.B. Ludermir, Genetic algorithm for reservoir computing optimization, in 2009 International Joint Conference on Neural Networks, June 2009, Atlanta, Ga, USA (IEEE, 2009), pp. 811–815

A.A. Ferreira, T.B. Ludermir, Comparing evolutionary methods for reservoir computing pre-training, in *The 2011 International Joint Conference on Neural Networks*, July 2011, San Jose, CA, USA (IEEE, 2011), pp. 283–290

J. Guckenheimer, P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields* (Springer, New York, 2002). OCLC: 51506830

C. Gui, R. Legtenberg, M. Elwenspoek, J.H. Fluitman, Q-factor dependence of one-port encapsulated polysilicon resonator on reactive sealing pressure. J. Micromech. Microeng. **5**(2), 183–185 (1995)

A.C. Harrie, Tilmans and Rob Legtenberg, Electrostatically driven vacuum-encapsulated polysilicon resonators: Part II. Theory and performance. Sens. Actuators A: Phys. **45**(1), 67–84 (1994)

A. Husain, J. Hone, H.W.Ch. Postma, X.M.H. Huang, T. Drake, M. Barbic, A. Scherer, M.L. Roukes, Nanowire-based very-high-frequency electromechanical resonator. Appl. Phys. Lett. **83**(6), 1240–1242 (2003)

B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. Mujica, A. Coates, A.Y. Ng, An empirical evaluation of deep learning on highway driving, April 2015 (2015), arXiv:1504.01716 [cs]

H. Jaeger, The echo state approach to analysing and training recurrent neural networks-with an erratum note. GMD Technical Report **148**(34), 13, German National Research Center for Information Technology, Bonn, Germany (2001)

H. Jaeger, A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach (2002), p. 46

H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)

Md. Jubayer Alam Rabin, Md. Safayet Hossain, Md. Solaiman Ahsan, Md. Abu Shahab Mollah, Md. Tawabur Rahman, Sensitivity learning oriented nonmonotonic multi reservoir echo state network for short-term load forecasting, in *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*, May 2013, Dhaka, Bangladesh (IEEE, 2013), pp. 1–6

T. Kalmar-Nagy, B. Balachandran, Forced harmonic vibration of a duffing oscillator with linear viscous damping, in *The Duffing Equation*, ed. by I. Kovacic, M.J. Brennan (John Wiley & Sons, Ltd., Chichester, 2011), pp. 139–174

F. Khoshnoud, C.W. de Silva, Recent advances in MEMS sensor technology-mechanical applications. IEEE Instrum. Meas. Mag. **15**(2), 14–24 (2012)

Y. Lai, J. McDonald, M. Kujath, T. Hubbard, Force, deflection and power measurements of toggled microthermal actuators. J. Micromech. Microeng. **14**(1), 49–56 (2004)

L. Larger, M.C. Soriano, D. Brunner, L. Appeltant, J.M. Gutierrez, L. Pesquera, C.R. Mirasso, I. Fischer, Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing. Opt. Express **20**(3), 3241 (2012)

L. Larger, A. Baylon-Fuentes, R. Martinenghi, V.S. Udaltsov, Y.K. Chembo, M. Jacquot, High-speed photonic reservoir computing using a time-delay-based architecture: million words per second classification. Phys. Rev. X **7**(1) (2017)

Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature **521**(7553), 436–444 (2015)

J.E.-Y. Lee, Y. Zhu, A.A. Seshia, A bulk acoustic mode single-crystal silicon microresonator with a high-quality factor. J. Micromech. Microeng. **18**(6) (2008)

D.A. Lieberman, *Learning: Behavior and Cognition*, 2nd edn. (Thomson Brooks/Cole Publishing Co., Belmont, CA, US, 1993)

R. Lifshitz, M.C. Cross, Nonlinear dynamics of nanomechanical resonators, in *Nonlinear Dynamics of Nanosystems*, ed. by G. Radons, B. Rumpf, H.G. Schuster (Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, Germany, 2010), pp. 221–266

H. Lin Wang, X.-Y.A. Huanling, H. Liu, Effective electricity energy consumption forecasting using echo state network improved by differential evolution algorithm. Energy **153**, 801–815 (2018)

C. Liu, *Foundations of MEMS* (Pearson Prentice Hall, Upper Saddle River, 2006)

R.F. Lyon, A computational model of filtering, detection, and compression in the cochlea **7**, 1282–1285 (1982)

M. Madou, *Fundamentals of Microfabrication* (CRC Press, Boca Raton, 1997)

P. Malatkar, A.H. Nayfeh, Calculation of the jump frequencies in the response of S.D.O.F. Nonlinear systems. J. Sound Vib. **254**(5), 1005–1011 (2002)

R. Martinenghi, S. Rybalko, M. Jacquot, Y.K. Chembo, L. Larger, Photonic nonlinear transient computing with multiple-delay wavelength dynamics. Phys. Rev. Lett. **108**(24) (2012)
A. Matthiessen, C. Vogt, On the influence of temperature on the electric conducting-power of alloys. Philos. Trans. R. Soc. Lond. 167–200 (1864)
K. Naeli, O. Brand, Dimensional considerations in achieving large quality factors for resonant silicon cantilevers in air. J. Appl. Phys. **105**(1) (2009)
A.H. Nayfeh, D.T. Mook, *Nonlinear Oscillations* (Wiley, 1995)
Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, S. Massar, Optoelectronic reservoir computing. Sci. Rep. **2**(1) (2012)
X. Pitkow, S. Liu, D.E. Angelaki, G.C. DeAngelis, A. Pouget, How can single sensory neurons predict behavior? Neuron **87**(2), 411–423 (2015)
H.W.Ch. Postma, I. Kozinsky, A. Husain, M.L. Roukes, Dynamic range of nanotube- and nanowire-based electromechanical systems. Appl. Phys. Lett. **86**(22) (2005)
M. Rigamonti, P. Baraldi, E. Zio, I. Roychoudhury, K. Goebel, S. Poll, Ensemble of optimized echo state networks for remaining useful life prediction. Neurocomputing **281**, 121–138 (2018)
S.B. Salah, I. Fliss, M. Tagina, Echo state network and particle swarm optimization for prognostics of a complex system, in *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, October 2017, Hammamet (IEEE, 2017), pp. 1027–1034
J.A. Sanders, F. Verhulst, J. Murdock, *Averaging Methods in Nonlinear Dynamical Systems*, 2nd edn., Applied Mathematical Sciences (Springer, New York, 2007)
A.T. Sergio, T.B. Ludermir, PSO for reservoir computing optimization, in *Artificial Neural Networks and Machine Learning - ICANN 2012*, vol. 7552, ed. by D. Hutchison, T. Kanade, J. Kittler, J.M. Kleinberg, F. Mattern, J.C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M.Y. Vardi, G. Weikum, A.E.P. Villa, W. Duch, P. Erdi, F. Masulli, G. Palm (Springer, Berlin, 2012), pp. 685–692
M.C. Soriano, S. Ortin, L. Keuninckx, L. Appeltant, J. Danckaert, L. Pesquera, G. van der Sande, Delay-based reservoir computing: noise effects in a combined analog and digital implementation. IEEE Trans. Neural Netw. Learn. Syst. **26**(2), 388–393 (2015)
J.B. Starr, Squeeze-film damping in solid-state accelerometers, in *IEEE 4th Technical Digest on Solid-State Sensor and Actuator Workshop* (1990), pp. 44–47
J. Sylvestre, G. Dion, B. Barazani, Provisional US patent application 62/780,589 (2018)
Y. Tadokoro, H. Tanaka, M.I. Dykman, Driven nonlinear nanomechanical resonators as digital signal detectors. Sci. Rep. **8**(1) (2018)
B. Tang, Mj. Brennan, V. Lopes, S. da Silva, R. Ramlan, Using nonlinear jumps to estimate cubic stiffness nonlinearity: an experimental study. Proc. Inst. Mech. Eng. Part C: J. Mech. Eng. Sci. **230**(19), 3575–3581 (2016)
Theory Reference for the Mechanical APDL and Mechanical Applications (2017)
H.A.C. Tilmans, M. Elwenspoek, J.H.J. Fluitman, Micro resonant force gauges. Sens. Actuators A: Phys. **30**(1), 35–53 (1992)
J. Torrejon, M. Riou, F.A. Araujo, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, K. Yakushiji, A. Fukushima, H. Kubota, S. Yuasa, M.D. Stiles, J. Grollier, Neuromorphic computing with nanoscale spintronic oscillators. Nature **547**(7664), 428–431 (2017)
J.T.M. van Beek, G.J.A.M. Verheijden, G.E.J. Koops, K.L. Phan, C. van der Avoort, J. van Wingerden, D. Ernur Badaroglu, J.J.M. Bontemps, Scalable 1.1 GHz fundamental mode piezo-resistive silicon MEMS resonator, in *2007 IEEE International Electron Devices Meeting*, December 2007, Washington, DC (IEEE, 2007), pp. 411–414
W.M. Van Spengen, R. Puers, I. De Wolf, On the physics of stiction and its impact on the reliability of microstructures. J. Adhes. Sci. Technol. **17**(4), 563–582 (2003)
S.S. Verbridge, J.M. Parpia, R.B. Reichenbach, L.M. Bellan, H.G. Craighead, High quality factor resonance at room temperature with nanostrings under high tensile stress. J. Appl. Phys. **99**(12), 124304 (2006)

D. Verstraeten, B. Schrauwen, D. Stroobandt, Isolated word recognition using a liquid state machine, in *Proceedings of the 13th European Symposium on Artificial Neural Networks (ESANN)* (2005), pp. 435–440

D. Witt, R. Kellogg, M. Snyder, J. Dunn, Windows into human health through wearables data analytics. Curr. Opin. Biomed. Eng. (2019)

K. Worden, On jump frequencies in the response of the duffing oscillator. J. Sound Vib. (1996)

N. Yazdi, F. Ayazi, K. Najafi, Micromachined inertial sensors. Proc. IEEE **86**(8), 20 (1998)

M.I. Younis, *MEMS Linear and Nonlinear Statics and Dynamics*, vol. 20, Microsystems (Springer, New York, 2010). OCLC: ocn495781913

S. Zaitsev, O. Shtempluck, E. Buks, O. Gottlieb, Nonlinear damping in a micromechanical oscillator. Nonlinear Dyn. **67**(1), 859–883 (2012)

Y. Zhang, Y. Yu, D. Liu, The application of modified ESN in chaotic time series prediction, in *2013 25th Chinese Control and Decision Conference (CCDC)*, May 2013, Guiyang, China (IEEE, 2013), pp. 2213–2218

H. Zhou, Y. Wang, K. Xing, Modeling of McKibben pneumatic artificial muscles using optimized echo state networks, in *2010 8th World Congress on Intelligent Control and Automation*, July 2010, Jinan, China (IEEE, 2010), pp. 1723–1728

Part IV
Physical Implementations: Neuromorphic Devices and Nanotechnology

Neuromorphic Electronic Systems for Reservoir Computing

Fatemeh Hadaeghi

Abstract This chapter provides a comprehensive survey of the researches and motivations for hardware implementation of reservoir computing (RC) on neuromorphic electronic systems. Due to its computational efficiency and the fact that training amounts to a simple linear regression, both spiking and non-spiking implementations of reservoir computing on neuromorphic hardware have been developed. Here, a review of these experimental studies is provided to illustrate the progress in this area and to address the technical challenges which arise from this specific hardware implementation. Moreover, to deal with the challenges of computation on such unconventional substrates, several lines of potential solutions are presented based on advances in other computational approaches in machine learning.

1 Introduction

The term "neuromorphic computing" refers to a variety of brain-inspired computers, architectures, devices, and models that are used in the endeavor to mimic biological neural networks (Mead and Ismail 2012). In contrast to von Neumann architectures, biologically inspired neuromorphic computing systems are promising for being highly connected and parallel, incorporating learning and adaptation, collocating memory and processing, and requiring low power. By creating parallel arrays of connected synthetic neurons that are asynchronous, real-time, and data- or event-driven, neuromorphic devices offer an expedient substrate to model neuroscience theories as well as implementing computational paradigms to solve challenging machine learning problems.

The accustomed growth rates of digital computing performance levels (Moore's Law) are showing signs of flattening out (Kish 2002). Furthermore, the explod-

F. Hadaeghi (✉)
Department of Computer Science and Electrical Engineering, Jacobs University Bremen, 28759 Bremen, Germany

Institute of Computational Neuroscience, University Medical Center Hamburg-Eppendorf (UKE), 20251 Hamburg, Germany
e-mail: f.hadaeghi@uke.de

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_10

ing energy demand of digital computing devices and algorithms is approaching the limits of what is socially and environmentally tolerable (Dreslinski et al. 2010). Neuromorphic technologies suggest escape routes from both predicaments due to their potentials for unclocked parallelism and minimal energy consumption of spiking dynamics. Moreover, neuromorphic systems have also received increased attention due to their scalability and small device footprint (Schuman et al. 2017).

Significant keys to such advancements are remarkable progress in material science and nanotechnology, low-voltage analog CMOS design techniques, and theoretical and computational neuroscience. At the device level, using the new materials and nanotechnologies for building extremely compact and low-power solid-state nanoscale devices has paved the way toward on-chip synapses with characteristic properties observed in biological synapses. For instance, "memory resistor" or the memristor, a nonlinear nanoscale electronic element with volatile and non-volatile modes, is ubiquitously used in neuromorphic circuits to store multiple bits of information and to emulate dynamic weights with intrinsic plasticity features (e.g., spike-time-dependent plasticity (STDP)) (Yang et al. 2013; Moon et al. 2019). It has been argued that a hybrid memristor–CMOS neuromorphic circuit may represent a proper building block for implementing biological-inspired probabilistic/stochastic/approximate computing paradigms that are robust to memristor device variability and fault-tolerant by design (Indiveri et al. 2013; Adam et al. 2017; Jo et al. 2010). Similarly, conductive-bridging RAM (CBRAM) (Suri et al. 2012a), which is a non-volatile memory technology, atomic switches (Aono and Hasegawa 2010)—nanodevices implemented using metal-oxide-based memristors or memristive materials—and tin oxide nanoparticles (Phuong et al. 2020) have also been fabricated to implement both short-term plasticity (STP) and long-term plasticity (LTP) in neuromorphic systems (Avizienis et al. 2012; Sillin et al. 2013). Both atomic switches and CBRAM have nano dimensions, are fast, and consume low energy (Schuman et al. 2017). Spike- time-dependent-depression and -potentiation observed in biological synapses have also been emulated by phase-change memory (PCM) elements in hybrid neuromorphic architectures where CMOS "neuronal" circuits are integrated with nanoscale "synaptic" devices. PCM elements are commonly used to achieve high density and scalability; they are compatible with CMOS circuits and show good endurance (Suri et al. 2012b; Ambrogio et al. 2016). Programmable metallization cells (PMCs) (Yu and Philip Wong 2010) and oxide-resistive memory (OXRAM) (Yu et al. 2011) are another types of resistive memory technologies that have been demonstrated to present STDP-like characteristics. Beyond the CMOS technologies, spintronic devices and optical (photonic) components have also been considered for neuromorphic implementation (Schuman et al. 2017; Tanaka et al. 2019).

At the hardware level, a transition away from purely digital systems to mixed analog/digital implementation to purely analog, unclocked, spiking neuromorphic microchips has led to the emergence of more biological-like models of neurons and synapses together with a collection of more biologically plausible adaptation and learning mechanisms. Digital systems are usually synchronous or clock-based, rely on Boolean logic-based gates and discrete values for computation, and tend to need

more power. Analog systems, in contrast, tend to rely more on the continuous values and inherent physical characteristics of electronic devices for computation and more closely resemble the biological brain, which takes the advantages of physical properties rather than Boolean logic for computation (Indiveri and Liu 2015). Analog systems, however, are significantly vulnerable to various types of thermal and electrical noise and artifacts. It is, therefore, argued that only computational paradigms that are robust to noise and faults may be proper candidates for analog implementation. Hence, among the wide diversity of computational models, a leading role is emerging for mixed analog/digital or purely analog implementation of artificial neural networks (ANNs). Two main trends in this arena are the exploit of low-energy, spiking neural dynamics for "deep learning" solutions (Merolla et al. 2014), and "reservoir computing" methods (Jaeger and Haas 2004).

At the network and system level, new discoveries in neuroscience research are being gradually incorporated into hardware models, and mathematical theory frameworks are being developed to guide the algorithms and hardware developments. A constructive paradigm shift has also occurred from strict structural replication of neuronal systems toward the hardware implementation of systemic/functional models of the biological brain (James et al. 2017). Consequently, over the past decade, a wide variety of model types have been implemented in electronic multi-neuron computing platforms to solve pattern recognition and machine learning tasks (Indiveri and Liu 2015; Misra and Saha 2010).

As mentioned above, among these computational frameworks, reservoir computing (RC) has been variously considered as a strategy to implement useful computations on such unconventional hardware platforms. An RC architecture comprises three major parts: the *input layer* feeds the input signal into a random, large, fixed recurrent neural network that constitutes the *reservoir*, from which the neurons in the *output layer* read out the desired output signal. In contrast to traditional (and "deep") recurrent neural networks (RNNs) training methods, the input-to-reservoir and the recurrent reservoir-to-reservoir weights in an RC system are left unchanged after a random initialization, and only the reservoir-to-output weights are optimized during training. Within computational neuroscience, RC is best known as *liquid state machines* (LSMs) (Maass et al. 2002), whereas the approach is known as *echo state networks* (ESNs) (Jaeger and Haas 2004) in machine learning. Reflecting the different objectives in these fields, LSM models are typically built around more or less detailed, spiking neuron models with biologically plausible parametrizations, while ESNs mostly use highly abstracted rate models for their neurons. Due to its computational efficiency, simplicity, and lenient requirements, both spiking and non-spiking implementations of reservoir computing on neuromorphic hardware exist. However, proper solutions are still lacking to address a variety of technological and information processing problems. For instance, regarding the choice of hardware, variations of stochasticity due to device mismatch, temporal drift, and aging must be taken into account; in the case of exploiting the spiking neurons, an appropriate encoding scheme needs to be developed to transform the input signal into spike trains; the speed of physical processes may require further adjustment to achieve online learn-

ing and real-time information processing; depending on the available technology, compatible local learning rules must be developed for on-chip learning.

Here, a review of recent experimental studies is provided to illustrate the progress in neuromorphic electronic systems for RC and to address the above-mentioned technical challenges which arise from such hardware implementations. Moreover, to deal with challenges of computation on such unconventional substrate, several lines of potential solutions are presented based on advances in other computational approaches in machine learning. In the remaining part of this chapter, we present an overview of the current approaches to implement reservoir computing model on digital neuromorphic processors, purely analog neuromorphic microchips, and mixed digital/analog neuromorphic systems. Since neuromorphic computing attempts to implement more biological-like models of neurons and synapses, spiking implementations of RC will be highlighted.

2 RC on Digital Neuromorphic Processors

Field programmable gate arrays (FPGAs) and application-specific integrated circuit (ASIC) chips—two categories of digital systems—have been very commonly utilized for neuromorphic implementations. To facilitate the application of computational frameworks in both embedded and stand-alone systems, FPGAs and ASICs offer considerable flexibility, reconfigurability, robustness, and fast prototyping (Wang et al. 2017).

As illustrated in Fig. 1, an FPGA-based reservoir computer consists of neuronal units, a neuronal arithmetic section, input/output encoding/decoding components, an on-chip learning algorithm, memory control, numerous memory blocks such as RAMs and/or memristors wherein the neuronal and synaptic information is stored, and a Read/Write interface to access to these memory blocks. Realizing spiking reservoir computing on such substrates entails tackling a number of critical issues related to spike coding, memory organization, parallel processing, on-chip learning, and tradeoffs between area, precision, and power overheads. Moreover, in real-world applications such as speech recognition and biosignal processing, spiking dynamics might be inherently faster than real-time performance. In this section, an overview of experimental studies is provided to illustrate how these challenges have so far been addressed in the literature.

In the first digital implementation of spiking RC, Schrauwen et al. (2007) suggested storing the neuronal information in RAM, exploiting non-plastic exponential synapses, and performing neuronal and synaptic operations serially. In this clock-based simulation, each time step consists of several operations such as adding weights to the membrane or synapse accumulators, adding the synapse accumulators to the membrane, decaying these accumulators, threshold detection, and membrane reset. On a real-time speech processing task, this study shows that in contrast to "serial processing, parallel arithmetic" (Upegui et al. 2005) and "parallel processing, serial arithmetic" (Schrauwen and Van Campenhout 2006), serial implementations of both

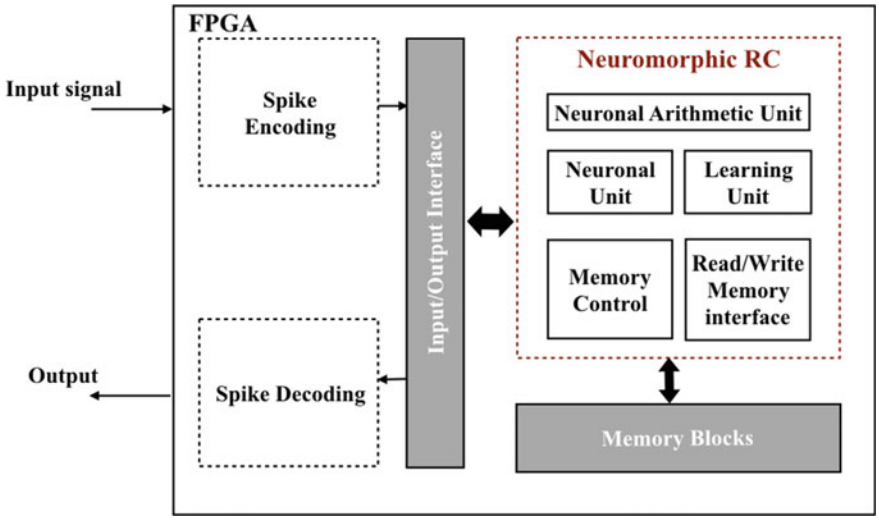


Fig. 1 A top-level schematic of an FPGA-based neuromporphic reservoir computer

arithmetic operations that define the dynamics of the neuron model and processing operations associated with synaptic dynamics yield to slower operations and low hardware costs. Although this architecture can be considered as the first compact digital neuromorphic RC system to solve a speech recognition task, the advantage of distributed computing is not explored in this framework. Moreover, for larger networks and in more sophisticated tasks, a sequential processor—which calculates every neuronal unit separately to simulate a single time step of the network—does not seem to be efficient. Stochastic arithmetic was, therefore, exploited to parallelize the calculations and obtain a considerable speedup (Verstraeten et al. 2005).

More biologically plausible models of spiking neurons (e.g., Hodgkin–Huxley, Morris–Lecar, and Izhikevich models (Gerstner and Kistler 2002) are too sophisticated to be efficiently implemented on hardware and have many parameters which need to be tuned. On the other hand, the simple hardware-friendly spiking neuron models, such as leaky-integrate-and-fire (LIF) (Gerstner and Kistler 2002), often have hard thresholding that makes supervised training difficult in spiking neural networks (SNNs). In spiking reservoir computing, however, the training of the readout mechanism amounts only to solving a linear regression problem, where the target output is a trainable linear combination of the neural signals within the reservoir. Linear regression is easily solved by standard linear algebra algorithms when arbitrary real-valued combination weights are admitted. However, for on-chip learning, the weights will be physically realized by states of electronic synapses, which currently can be reliably set only to a very small number of discrete values. It has been recently proved that computing optimal discrete readout weights in reservoir computing is NP-hard, and approximate (or heuristic) methods must be exploited to obtain high-quality solutions in a reasonable time for practical uses (Hadaeghi and

Jaeger 2019). The spike-based learning algorithm proposed by Zhang et al. (2015) is an example of such approximate solutions for FPGA implementations. In contrast to offline learning methods, the proposed online learning rule avoids any intermediate data storage. Besides, through this abstract learning process, each synaptic weight is adjusted based on only the firing activities of the corresponding pre- and post-synaptic neurons, independent of the global communications across the neural network. In a speech recognition task, it has been shown that due to this locality, the overhead of hardware implementation (e.g., synaptic and membrane voltage precision) can be reduced without drastic effects on its performance (Zhang et al. 2015). This biological-inspired supervised learning algorithm was later exploited in developing an RC-based general-purpose neuromorphic architecture where training the task-specific output neurons is integrated into the reconfigurable FPGA platform (Wang et al. 2015). The effects of catastrophic failures such as broken synapses, dead neurons, random errors as well as errors in arithmetic operations (e.g., comparison, adding and shifting) in similar architecture were addressed in (Jin and Li 2017) where the reservoir consists of excitatory and inhibitory LIF neurons that are randomly connected with non-plastic synapses governed by second-order response functions. The simulation results suggest that at least an 8-bit resolution is needed for the efficacy of plastic synapses if the spike-based learning algorithm proposed by Zhang et al. (2015) is applied for training the readout weights. The recognition performance only degrades slightly when the precision of fixed synaptic weights and membrane voltage for reservoir neurons are reduced down to 6 bits.

To realize the on-chip learning on digital systems with extremely low-bit precision (binary values), Jin et al. (2016) suggested an online pruning algorithm based on variances of firing activities to sparsify the readout connections. Cores to this reconfiguration scheme are the STDP learning rule, firing activity monitoring, and variance estimation. The readout synapses projected from low-variance reservoir neurons are then powered off to save energy. To reduce the energy consumption of the hardware in FPGA, the reservoir computer developed by Wang et al. (2016, 2017) utilizes firing activity-based power gating by turning off the neurons that seldom fire for a particular benchmark and applies approximate arithmetic computing (Shao and Li 2015) to speed up the runtime in a speech recognition task.

Exploiting the spatial locality in a reservoir consisting of excitatory and inhibitory LIF neurons, a fully parallel design approach was also presented for real-time biomedical signal processing in Polepalli et al. (2016a, b). In this design, the memory blocks were replaced by distributed memory to circumvent the long access time due to wire delay. Being inspired by new discoveries in neuroscience, Smith et al. (2017) developed a spiking temporal processing unit (STPU) to efficiently implement more biologically plausible synaptic response functions in digital architectures. STPU offers a local temporal memory buffer including an arbitrary number of memory cells to model delayed synaptic transmission between neurons. This allows multiple connections between neurons with different synaptic latencies. Utilizing excitatory and inhibitory LIF neurons with different time scales and exploiting the second-order response function to model synaptic transmission, the effects of the input signal in a spoken digit recognition task will only slowly "wash out" over time,

enabling the reservoir to provide sufficient dynamical short-term memory capacity. In order to create a similar short-term memory on a reconfigurable digital platform with extremely low-bit resolutions, an STDP mechanism was proposed by Jin et al. (2016) for on-chip reservoir tuning. Applying this local learning rule to the reservoir synapses together with a data-driven binary quantization algorithm creates sparse connectivities within the reservoir in a self-organized fashion, leading to significant energy reduction. Stochastic activity-based STDP approach (Jin and Li 2016), structural plasticity-based mechanism (Roy and Basu 2016), and correlation-based neuron gating rule (Liu et al. 2018) have also been introduced for efficient low-resolution tuning of the reservoir in hardware. Neural processes, however, require a large number of memory resources for storing various parameters, such as synaptic weights and internal neural states, and lead to a heavy clock distribution loading and a significant power dissipation. To tackle this problem, Liu et al. (2018) suggested partitioning the memory elements inside each neuron where separate elements are activated at different phases of neural processing. This leads to an activity-based clock gating mechanism with a granularity of a partitioned memory group inside each neuron.

Another technique for power reduction is incorporating memristive crossbars into digital neuromorphic hardware to perform synaptic operations in on-chip/online learning and to store the synaptic weights in offline learning (Soures et al. 2017; Moon et al. 2019; Midya et al. 2019). In general, a two-terminal memristor device can be programmed by applying a large enough potential difference across its terminals, where the state change of the device is dependent on the magnitude, duration, and polarity of the potential difference. The device resistance states, representing synaptic weights, will vary between a high-resistance state (HRS) and a low-resistance state (LRS), depending on the polarity of the voltage applied. At a system level, various sources of noise (e.g., random telegraph, thermal noise, and 1/f noise) arise from non-ideal behavior in memristive devices and distort the process of reading from the synapses and updating the synaptic weights. Given theoretical models for these stochastic noise processes, the effects of different manifestations of memristor read and write noise on the accuracy of neuromorphic RC in a classification task were investigated in Soures et al. (2017). Other hardware feasible RC structures with memristor double crossbar array have also been proposed in Hassan et al. (2017), Moon et al. (2019), Midya et al. (2019), and Wlaźlak et al. (2020) and were tested on a real-time time series prediction/classification tasks. These studies not only confirm that reservoir computing can properly cope with low-precision environments and noisy data, but they also experimentally demonstrate how RC benefits from memristor device variation to secure a more random heterogeneous weight distribution leading to a more diverse response for similar inputs. Although memristive crossbars are area- and energy-efficient, the digital circuitry to control the read/write logic to the crossbars is extremely power-hungry, thus preventing the use of memristors in large-scale memory systems.

In RC literature, it has also been shown that in time series prediction tasks, a cyclic reservoir—where neurons are distributed on a ring and connected with the same synaptic weights to produce the cyclic rotations of the input vector—is able to operate with an efficiency comparable to the standard RC models (Rodan and Tino

2011; Appeltant et al. 2011). In order to minimize the required hardware resources, therefore, single cyclic reservoir systems with stochastic spiking neurons were implemented for real-time time series prediction and classification tasks (Alomar et al. 2016a; Wlaźlak et al. 2020). The architectures show a high level of scalability and prediction performance comparable to the software simulations. In non-spiking reservoir computing implementations on reconfigurable digital systems, the reservoir has sometimes been configured according to a ring topology, and the readout weights are trained through gradient descent algorithm or ridge regression (Antonik et al. 2015; Yi et al. 2016; Alomar et al. 2016b).

3 RC on Analog Neuromorphic Microchips

Digital platforms, tools, and simulators offer robust and practical solutions to a wide range of engineering problems and provide convenient approaches to explore the quantitative behavior of neural networks. However, they do not seem to be ideal substrates for simulating the detailed large-scale models of neural systems where density, energy efficiency, and resilience are important. Besides, the observation that the brain operates on analog principles and relies on the inherent physical characteristics of the neuronal system for computation motivated the investigations in the field of neuromorphic engineering. Following the pioneering work conducted by Carver Mead (Mead and Ismail 2012)—therein biologically inspired electronic sensors were integrated with analog circuits, and an address-event-based asynchronous continuous-time communications protocol was introduced—over the last decade, mixed analog/digital and purely analog very large scale integration (VLSI) circuits have been fabricated to emulate the electrophysiological behavior of biological neurons and synapses and to offer a medium in which neuronal networks can be presented directly in hardware rather than simply simulated on general-purpose computers (Indiveri et al. 2011). However, the fact that such platforms provide only a qualitative approximation to the exact performance of digitally simulated neural systems may preclude them from being the ideal substrate for solving engineering problems and achieving machine learning tasks where detailed quantitative investigations are essential. In addition to the low resolution, realizing global asynchrony and dealing with noisy and unreliable components seem to be formidable technical hurdles. An architectural solution can be partly provided by reservoir computing due to its bio-inspired principle, robustness to the imperfect substrate, and the fact that only readout weights need to be trained (Schürmann et al. 2005). Similar to other analog and digital computing chips, transistors are the building blocks of analog neuromorphic devices. It has been experimentally shown that in the "subthreshold" region of operation, the current–voltage characteristic curve of the transistor is exponential and analogous to the exponential dependence of active populations of voltage-gated ionic channels as a function of the potential across the membrane of a neuron. This similarity has paved the way toward the fabrication of compact analog circuits that implement electronic models of voltage-sensitive conductance-based neurons and synapses as well

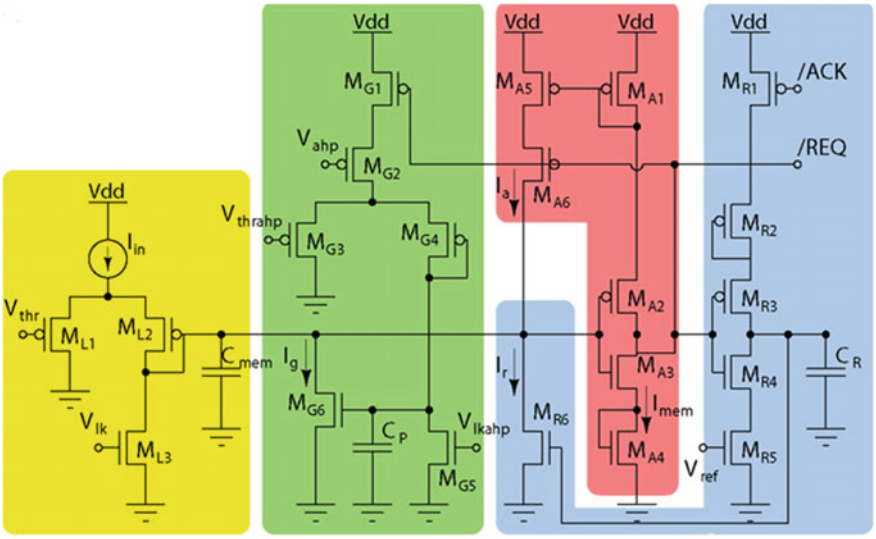


Fig. 2 A schematic of (DPI) neuron circuit (reproduced from Indiveri et al. (2011))

as computational circuitry to perform logarithmic functions, amplification, thresholding, multiplication, inhibition, and winner-take-all selection (Liu and Delbruck 2010; Donahue et al. 2015). An example of biophysically realistic neural electronic circuits is depicted in Fig. 2. This differential pair integrator (DPI) neuron circuit consists of four components: (1) the input DPI filter (M_{L1}–M_{L3}) including the integrating membrane capacitor C_{mem} models the neuron's leak conductance which produces exponential subthreshold dynamics in response to constant input currents, (2) a spike event generating amplifier (M_{A1}–M_{A6}) together with a positive-feedback circuit represent both sodium channel activation and inactivation dynamics, (3) a spike reset circuit with address-event representation (AER) handshaking signals and refractory period functionality (M_{R1}–M_{R6}) emulates the potassium conductance functionality, and (4) a spike-frequency adaptation mechanism implemented by an additional DPI filter (M_{G1}–M_{G6}) produces an after hyper-polarizing current proportional to the neuron's mean firing rate. See Indiveri et al. (2011) for an insightful review of different design methodologies used for silicon neuron fabrication.

In conjunction with circuitry that operates in subthreshold mode, exploiting memristive devices to model synaptic dynamics is a common approach in analog neuromorphic systems for power efficiency and performance boosting purposes (Indiveri and Liu 2015; Avizienis et al. 2012). In connection with the fully analog implementation of RC, it has been shown that by creating a functional reservoir consisting of numerous neurons connected through atomic nano-switches, the functional characteristics required for implementing biologically inspired computational methodologies in a synthetic experimental system can be displayed (Avizienis et al. 2012). A memristor crossbar structure has also been fabricated to connect the analog non-

spiking reservoir neurons—which are distributed on a ring topology—to the output node (Donahue et al. 2015; Merkel et al. 2014). In this CMOS-compatible crossbar, multiple weight states are achieved using multiple bi-stable memristors (with only two addressable resistance states). The structural simplicity of this architecture paves the way to independent control of each synaptic element. However, the discrete nature of memristive weights places a limit on the accuracy of reservoir computing; thus the number of weight states per synapse, that are required for satisfactory accuracy, must be determined beforehand. Besides, appropriate learning algorithms for on-chip training purpose have not yet been implemented for fully analog reservoir computing. Another significant design challenge associated with memristor devices is cycle-to-cycle variability in the resistance values even within a single memristor. With the same reservoir structure, therefore, Yang et al. (2016) showed by connecting memristor devices in series or parallel, a staircase memristor model could be constructed which not only has a delayed-switching effect between several somewhat stable resistance levels, but also can provide more reliable state values if a specific resistance level is required. This model of synaptic delay is particularly relevant for time delay reservoir methodologies (Appeltant et al. 2011).

From the information processing point of view, the ultimate aim of neuromorphic systems is to carry out neural computation in an energy-efficient way. However, firstly, the quantities relevant to the computation have to be expressed in terms of the spikes that spiking neurons communicate with. One of the key components in both digital and analog neuromorphic reservoir computers is, therefore, a neural encoder which transforms the input signals into spike trains. Although the nature of the neural code (or neural codes) is an unresolved topic of research in neuroscience, based on what is known from biology, a number of neural information encodings have been proposed (Grüning and Bohte 2014). Among them, hardware prototypes of rate encoding, temporal encoding, inter-spike interval encoding, and latency (or rank order) encoding schemes have been implemented, mostly on digital platforms (Yi et al. 2016; Yang et al. 2016; Bichler et al. 2012; Zhao et al. 2016). Digital configurations, however, require large hardware overheads associated with analog-to-digital converters, operational amplifiers, digital buffers, and electronic synchronizers and will increase the cost of implementation. Particularly, in analog reservoir computers, fabricating a fully analog encoding spike generator is of crucial importance both to speed the process up and to optimize the required energy and hardware costs. Examples of such generators have been mainly proposed for analog implementation of delayed feedback reservoirs (Zhao et al. 2016; Li et al. 2017).

4 RC on Mixed Digital/Analog Neuromorphic Systems

The majority of analog neuromorphic systems designed to solve machine learning/signal processing problems tend to rely on digital components, for instance, for pre-processing, encoding spike generation, storing the synaptic weights values, and on-chip programmability/learning (Murray and Smith 1988; Azghadi et al. 2013;

Briiderle et al. 2010). Inter-chip and chip-to-chip communications are also primarily digital in some analog neuromorphic platforms (Murray and Smith 1988; Aamir et al. 2016; Chicca et al. 2006). Mixed analog/digital architectures are, therefore, very common in neuromorphic systems. These electronic chips often provide a universal substrate with an extensive configuration space to emulate different types of neurons, synapses, and network typologies. On the hardware developed in Pfeil et al. (2013), for instance, together with five other neural network models, a spiking reservoir model consisting of excitatory and inhibitory populations and a readout tempotron neuron was implemented and trained to classify spike train segments in a continuous data stream. The hardware offers on-chip spike-based learning mechanisms to compensate for fixed-pattern noise.

In the context of hardware reservoir computing and inspired by the nonlinear properties of dendrites in biological neurons, Roy et al. (2014) proposed a readout learning mechanism which returns binary values for synaptic weights such that the "choice" of connectivity can be implemented in a mixed analog/digital platform with the address-event representation (AER) protocols where the connection matrix is stored in digital memory. Relying on the same communication protocol and exploiting the reconfigurable online learning spiking neuromorphic processor (ROLLS chip) introduced in Qiao et al. (2015), a reservoir computer was later designed and fabricated to detect spike-patterns in bio-potential and local field potential (LFP) recordings (Corradi and Indiveri 2015). The ROLLS neuromorphic processor (Fig. 3) contains 256 adaptive exponential integrate-and-fire (spiking) neurons implemented with mixed-signal analog/digital circuits. The neurons are connected to an array of 256×256 learning synapse circuits for modeling long-term plasticity mechanisms, an array of 256×256 programmable synapses with short-term plasticity circuits, and 256×2 row of "virtual synapses" for modeling excitatory and inhibitory synapses that have shared synaptic weights and time constants. The reservoir in this model comprises 128 randomly connected spiking neurons that receive spike event inputs from a neural recording system and aim to enhance the temporal properties of input patterns. The readout layer is afterward trained by applying a spike-based Hebbian-like learning rule previously proposed in Brader et al. (2007). The results of this study suggest that the device mismatch and the limited precision of the analog circuits result in a diversity of neuronal responses that might be beneficial in a population coding scheme within the reservoir computing framework.

Later, in another implementation of reservoir computing on mixed analog/digital substrates endowed with learning abilities, the neuronal circuits were chosen to be analog for power and area efficiency considerations, and an on-chip learning mechanism based on recursive least squares (RLS) algorithm was implemented on an FPGA platform (Yi et al. 2016). The proposed architecture was then applied to modeling a multiple-input multiple-output orthogonal frequency division multiplexing (MIMO-OFDM) system.

A completely on-chip feasible design for RC has also been proposed in Kudithipudi et al. (2016), where non-spiking reservoir neurons are distributed in a hybrid topology and memristive nano synapses are used in the output or regression layer. The proposed hybrid topology consists of a ring RC and a center neuron connected to

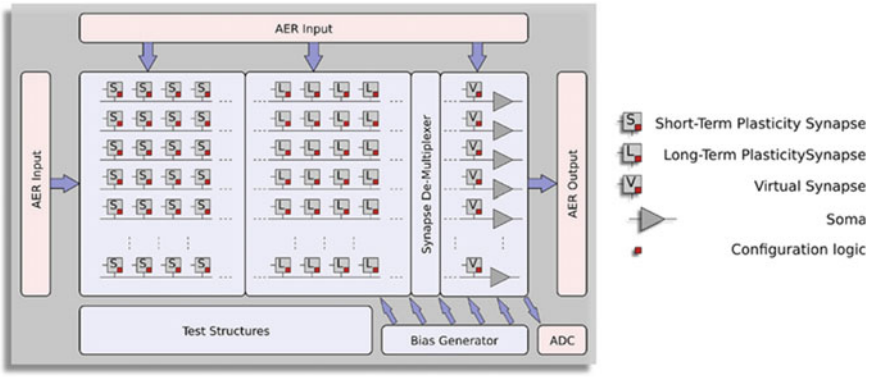


Fig. 3 Architecture of ROLLS neuromorphic processor (reproduced from Qiao et al. (2015))

all the reservoir neurons which transmit the information about the state of the whole reservoir to each neuron. To fabricate non-spiking silicon neurons, current-mode differential amplifiers operating in their subthreshold regime are used to emulate the hyperbolic tangent rate model behavior. In practice, it has been shown that a random distribution of weights in input-to-reservoir and reservoir synapses can be obtained by employing mismatches in transistors threshold voltages to design subthreshold bipolar synapses.

In order to create a functional spiking reservoir on the Dynap-se board (Moradi et al. 2018), a "Reservoir Transfer Method" was also proposed to transfer the functional ESN-type reservoir into a reservoir with analog unclocked spiking neurons (He et al. 2019). The Dynap-se board is a multicore neuromorphic processor chip that employs hybrid analog/digital circuits for emulating synapse and neuron dynamics together with asynchronous digital circuits for managing the address-event traffic. It offers 4000 adaptive exponential integrate-and-fire (spiking) neurons implemented with mixed-signal analog/digital circuits with $64/4k$ fan-in/out. From the computational point of view, implementing efficient algorithms on this neuromorphic system encounters general problems such as bit resolution, device mismatch, uncharacterized neural models, inaccessible state variables, and physical system noise. Realizing the reservoir computing on this substrate, additionally, leads to an important problem associated with fine-tuning the reservoir to obtain the memory span required for a specific machine learning problem. The reservoir transfer method proposes a theoretical framework to learn ternary weights in a reservoir of spiking neurons by transferring features from a well-tuned echo state network simulated on software (He et al. 2019). Empirical results from an ECG signal monitoring task showed that this reservoir with ternary weights is able to not only integrate information over a time span longer than the timescale of individual neurons but also function as an information processing medium with performance close to a standard, high precision, deterministic, non-spiking ESN (He et al. 2019).

5 Conclusion

Reservoir computing appears to be a particularly widespread and versatile approach to harness unconventional nonlinear physical phenomena into useful computations. Spike-based neuromorphic microchips, on the other hand, promise one or two orders of magnitude less energy consumption than traditional digital microchips. Implementation of reservoir computing methodologies on neuromorphic hardware, therefore, has been an attractive practice in neuromorphic system design. Here, a review of experimental studies was provided to illustrate the progress in this area and to address the computational bottlenecks which arise from specific hardware implementations. Moreover, to deal with the challenges of computation on such unconventional substrates, several lines of potential solutions were presented based on advances in other computational approaches in machine learning.

Acknowledgements This work was supported by the European H2020 collaborative project NeuRAM3 [grant number 687299]. I would also like to thank Herbert Jaeger, who provided insight and expertise that greatly assisted this research.

References

S.A. Aamir, P. Müller, A. Hartel, J. Schemmel, K. Meier, A highly tunable 65-nm CMOS LIF neuron for a large scale neuromorphic system, in *ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference* (IEEE, 2016), pp. 71–74

G.C. Adam, B.D. Hoskins, M. Prezioso, F. Merrikh-Bayat, B. Chakrabarti, D.B. Strukov, 3-d memristor crossbars for analog and neuromorphic computing applications. IEEE Trans. Electron Dev. **64**(1), 312–318 (2017)

M.L. Alomar, V. Canals, A. Morro, A. Oliver, J.L. Rossello, Stochastic hardware implementation of liquid state machines, in *2016 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2016a), pp. 1128–1133

M.L. Alomar, V. Canals, N. Perez-Mora, V. Martínez-Moll, J.L. Rosselló, FPGA-based stochastic echo state networks for time-series forecasting. Comput. Intell. Neurosci. **2016**, 15 (2016b)

S. Ambrogio, N. Ciocchini, M. Laudato, V. Milo, A. Pirovano, P. Fantini, D. Ielmini, Unsupervised learning by spike timing dependent plasticity in phase change memory (PCM) synapses. Front. Neurosci. **10**, 56 (2016)

P. Antonik, A. Smerieri, F. Duport, M. Haelterman, S. Massar, FPGA implementation of reservoir computing with online learning, in *24th Belgian-Dutch Conference on Machine Learning* (2015)

M. Aono, T. Hasegawa, The atomic switch. Proc. IEEE **98**(12), 2228–2236 (2010)

L. Appeltant, M.C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C.R. Mirasso, I. Fischer, Information processing using a single dynamical node as complex system. Nat. Commun. **2**, 468 (2011)

A.V. Avizienis, H.O. Sillin, C. Martin-Olmos, H.H. Shieh, M. Aono, A.Z. Stieg, J.K. Gimzewski, Neuromorphic atomic switch networks. PloS One **7**(8), e42772 (2012)

M.R. Azghadi, S. Moradi, G. Indiveri, Programmable neuromorphic circuits for spike-based neural dynamics, in *2013 IEEE 11th International New Circuits and Systems Conference (NEWCAS)* (IEEE, 2013), pp. 1–4

O. Bichler, D. Querlioz, S.J. Thorpe, J.-P. Bourgoin, C. Gamrat, Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity. Neural Netw. **32**, 339–348 (2012)

J.M. Brader, W. Senn, S. Fusi, Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. Neural Comput. **19**(11), 2881–2912 (2007)

D. Briiderle, J. Bill, B. Kaplan, J. Kremkow, K. Meier, E. Müller, J. Schemmel, Simulator-like exploration of cortical network architectures with a mixed-signal VLSI system, in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems* (IEEE, 2010), pp. 2784–8787

E. Chicca, P. Lichtsteiner, T. Delbruck, G. Indiveri, R.J. Douglas, Modeling orientation selectivity using a neuromorphic multi-chip system, in *2006 IEEE International Symposium on Circuits and Systems* (IEEE, 2006), pp. 1235–1238

F. Corradi, G. Indiveri, A neuromorphic event-based neural recording system for smart brain-machine-interfaces. IEEE Trans. Biomed. Circuits Syst. **9**(5), 699–709 (2015)

C. Donahue, C. Merkel, Q. Saleh, L. Dolgovs, Y.K. Ooi, D. Kudithipudi, B. Wysocki, Design and analysis of neuromemristive echo state networks with limited-precision synapses, in *2015 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)* (IEEE, 2015), pp. 1–6

R.G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, T. Mudge, Near-threshold computing: reclaiming Moore's law through energy efficient integrated circuits. Proc. IEEE **98**(2), 253–266 (2010)

W. Gerstner, W.M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity* (Cambridge University Press, Cambridge, 2002)

A. Grüning, S.M. Bohte, Spiking neural networks: principles and challenges, in *ESANN* (2014)

F. Hadaeghi, H. Jaeger, Computing optimal discrete readout weights in reservoir computing is np-hard. Neurocomputing **338**, 233–236 (2019)

A.M. Hassan, H.H. Li, Y. Chen, Hardware implementation of echo state networks using memristor double crossbar arrays, in *2017 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2017), pp. 2171–2177

X. He, T. Liu, F. Hadaeghi, H. Jaeger, Reservoir transfer on analog neuromorphic hardware, in *9th International IEEE EMBS Neural Engineering Conference* (2019)

G. Indiveri, S.-C. Liu, Memory and information processing in neuromorphic systems. Proc. IEEE **103**(8), 1379–1397 (2015)

G. Indiveri, B. Linares-Barranco, T.J. Hamilton, A. Van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud et al., Neuromorphic silicon neuron circuits. Front. Neurosci. **5**, 73 (2011)

G. Indiveri, B. Linares-Barranco, R. Legenstein, G. Deligeorgis, T. Prodromakis, Integration of nanoscale memristor synapses in neuromorphic computing architectures. Nanotechnology **24**(38) (2013)

H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)

C.D. James, J.B. Aimone, N.E. Miner, C.M. Vineyard, F.H. Rothganger, K.D. Carlson, S.A. Mulder, T.J. Draelos, A. Faust, M.J. Marinella et al., A historical survey of algorithms and hardware architectures for neural-inspired and neuromorphic computing applications. Biol. Inspired Cogn. Archit. **19**, 49–64 (2017)

Y. Jin, P. Li, AP-STDP: a novel self-organizing mechanism for efficient reservoir computing, in *2016 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2016), pp. 1158–1165

Y. Jin, P. Li, Performance and robustness of bio-inspired digital liquid state machines: a case study of speech recognition. Neurocomputing **226**, 145–160 (2017)

Y. Jin, Y. Liu, P. Li, Sso-lsm: a sparse and self-organizing architecture for liquid state machine based neural processors, in *2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)* (IEEE, 2016), pp. 55–60

S.H. Jo, T. Chang, I. Ebong, B.B. Bhadviya, P. Mazumder, W. Lu, Nanoscale memristor device as synapse in neuromorphic systems. Nano Lett. **10**(4), 1297–1301 (2010)

L.B. Kish, End of Moore's law: thermal (noise) death of integration in micro and nano electronics. Phys. Lett. A **305**(3–4), 144–149 (2002)

Dhireesha Kudithipudi, Qutaiba Saleh, Cory Merkel, James Thesing, Bryant Wysocki, Design and analysis of a neuromemristive reservoir computing architecture for biosignal processing. Front. Neurosci. **9**, 502 (2016)

J. Li, C. Zhao, K. Hamedani, Y. Yi, Analog hardware implementation of spike-based delayed feedback reservoir computing system, in *2017 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2017), pp. 3439–3446

S.-C. Liu, T. Delbruck, Neuromorphic sensory systems. Curr. Opin. Neurobiol. **20**(3), 288–295 (2010)

Y. Liu, Y. Jin, P. Li, Online adaptation and energy minimization for hardware recurrent spiking neural networks. ACM J. Emerg. Technol. Comput. Syst. (JETC) **14**(1), 11 (2018)

W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. **14**(11), 2531–2560 (2002)

C. Mead, M. Ismail, *Analog VLSI Implementation of Neural Systems*, vol. 80 (Springer Science & Business Media, 2012)

C. Merkel, Q. Saleh, C. Donahue, D. Kudithipudi, Memristive reservoir computing architecture for epileptic seizure detection. Proc. Comput. Sci. **41**, 249–254 (2014)

P.A. Merolla, J.V. Arthur, R. Alvarez-Icaza, A.S. Cassidy, J. Sawada, F. Akopyan, B.L. Jackson, N. Imam, C. Guo, Y. Nakamura et al., A million spiking-neuron integrated circuit with a scalable communication network and interface. Science **345**(6197), 668–673 (2014)

R. Midya, Z. Wang, S. Asapu, X. Zhang, M. Rao, W. Song, Y. Zhuo, N. Upadhyay, Q. Xia, J.J. Yang, Reservoir computing using diffusive memristors. Adv. Intell. Syst. **1**(7), 1900084 (2019)

J. Misra, I. Saha, Artificial neural networks in hardware: a survey of two decades of progress. Neurocomputing **74**(1–3), 239–255 (2010)

J. Moon, W. Ma, J.H. Shin, F. Cai, C. Du, S.H. Lee, W.D. Lu, Temporal data classification and forecasting using a memristor-based reservoir computing system. Nat. Electron. **2**(10), 480–487 (2019)

S. Moradi, N. Qiao, F. Stefanini, G. Indiveri, A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). IEEE Trans. Biomed. Circuits Syst. **12**(1), 106–122 (2018)

A.F. Murray, A.V.W. Smith, Asynchronous VLSI neural networks using pulse-stream arithmetic. IEEE J. Solid-State Circuits **23**(3), 688–697 (1988)

T. Pfeil, A. Grübl, S. Jeltsch, E. Müller, P. Müller, M.A. Petrovici, M. Schmuker, D. Brüderle, J. Schemmel, K. Meier, Six networks on a universal neuromorphic computing substrate. Front. Neurosci. **7**, 11 (2013)

Phuong Y Le, Billy J Murdoch, Anders J Barlow, Anthony S Holland, Dougal G McCulloch, Chris F McConville, and Jim G Partridge. Electroformed, self-connected tin oxide nanoparticle networks for electronic reservoir computing. *Advanced Electronic Materials*, page 2000081, 2020

A. Polepalli, N. Soures, D. Kudithipudi, Digital neuromorphic design of a liquid state machine for real-time processing, in *2016 IEEE International Conference on Rebooting Computing (ICRC)* (IEEE, 2016a), pp. 1–8

A. Polepalli, N. Soures, D. Kudithipudi, Reconfigurable digital design of a liquid state machine for spatio-temporal data, in *Proceedings of the 3rd ACM International Conference on Nanoscale Computing and Communication* (ACM, 2016b), p. 15

Ning Qiao, Hesham Mostafa, Federico Corradi, Marc Osswald, Fabio Stefanini, Dora Sumislawska, Giacomo Indiveri, A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. Front. Neurosci. **9**, 141 (2015)

A. Rodan, P. Tino, Minimum complexity echo state network. IEEE Trans. Neural Netw. **22**(1), 131–144 (2011)

S. Roy, A. Basu, An online structural plasticity rule for generating better reservoirs. Neural Comput. **28**(11), 2557–2584 (2016)

S. Roy, A. Banerjee, A. Basu, Liquid state machine with dendritically enhanced readout for low-power, neuromorphic VLSI implementations. IEEE Trans. Biomed. Circuits Syst. **8**(5), 681–695 (2014)

B. Schrauwen, J. Van Campenhout, Parallel hardware implementation of a broad class of spiking neurons using serial arithmetic, in *Proceedings of the 14th European Symposium on Artificial Neural Networks* (d-side publications, 2006), pp. 623–628

B. Schrauwen, M. D'Haene, D. Verstraeten, J. Van Campenhout, Compact hardware for real-time speech recognition using a liquid state machine, in *2007 International Joint Conference on Neural Networks* (IEEE, 2007), pp. 1097–1102

C.D. Schuman, T.E. Potok, R.M. Patton, J.D. Birdwell, M.E. Dean, G.S. Rose, J.S. Plank, A survey of neuromorphic computing and neural networks in hardware (2017), arXiv:1705.06963

F. Schürmann, K. Meier, J. Schemmel, Edge of chaos computation in mixed-mode VLSI-a hard liquid, in *Advances in Neural Information Processing Systems* (2005), pp. 1201–1208

B. Shao, P. Li, Array-based approximate arithmetic computing: a general model and applications to multiplier and squarer design. IEEE Trans. Circuits Syst. I Regul. Pap. **62**(4), 1081–1090 (2015)

H.O. Sillin, R. Aguilera, H.-H. Shieh, A.V. Avizienis, M. Aono, A.Z. Stieg, J.K. Gimzewski, A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing. Nanotechnology **24**(38), 384004 (2013)

M.R. Smith, A.J. Hill, K.D. Carlson, C.M. Vineyard, J. Donaldson, D.R. Follett, P.L. Follett, J.H. Naegle, C.D. James, J.B. Aimone, A novel digital neuromorphic architecture efficiently facilitating complex synaptic response functions applied to liquid state machines, in *2017 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2017), pp. 2421–2428

N. Soures, L. Hays, D. Kudithipudi, Robustness of a memristor based liquid state machine, in *2017 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2017), pp. 2414–2420

M. Suri, O. Bichler, D. Querlioz, G. Palma, E. Vianello, D. Vuillaume, C. Gamrat, B. DeSalvo, Cbram devices as binary synapses for low-power stochastic neuromorphic systems: auditory (cochlea) and visual (retina) cognitive processing applications, in *2012 International Electron Devices Meeting* (IEEE, 2012a), pp. 10–13

M. Suri, O. Bichler, D. Querlioz, B. Traoré, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume, C. Gamrat, B. DeSalvo, Physical aspects of low power synapses based on phase change memory devices. J. Appl. Phys. **112**(5) (2012b)

G. Tanaka, T. Yamane, J.B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, A. Hirose, Recent advances in physical reservoir computing: a review. Neural Netw. (2019)

A. Upegui, C.A. Pena-Reyes, E. Sanchez, An FPGA platform for on-line topology exploration of spiking neural networks. Microprocess. Microsyst. **29**(5), 211–223 (2005)

D. Verstraeten, B. Schrauwen, D. Stroobandt, Reservoir computing with stochastic bitstream neurons, in *Proceedings of the 16th Annual Prorisc Workshop* (2005), pp. 454–459

Q. Wang, Y. Jin, P. Li, General-purpose lsm learning processor architecture and theoretically guided design space exploration, in *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)* (IEEE, 2015), pp. 1–4

Q. Wang, Y. Li, P. Li, Liquid state machine based pattern recognition on fpga with firing-activity dependent power gating and approximate computing, in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE, 2016), pp. 361–364

Q. Wang, Y. Li, B. Shao, S. Dey, P. Li, Energy efficient parallel neuromorphic architectures with approximate arithmetic on FPGA. Neurocomputing **221**, 146–158 (2017)

E. Wlaźlak, P. Zawal, K. Szaciłowski, Neuromorphic applications of a multivalued [Sni4 {(C6H5) 2SO} 2] memristor incorporated in the echo state machine. ACS Appl. Electron. Mater. **2**(2), 329–338 (2020)

J.J. Yang, D.B. Strukov, D.R. Stewart, Memristive devices for computing. Nat. Nanotechnol. **8**(1), 13 (2013)

X. Yang, W. Chen, F.Z. Wang, Investigations of the staircase memristor model and applications of memristor-based local connection. Analog. Integr. Circuits Signal Process. **87**(2), 263–273 (2016)

Y. Yi, Y. Liao, F. Bin Wang, F.S. Xin, H. Hou, L. Liu, Fpga based spike-time dependent encoder and reservoir design in neuromorphic computing processors. Microprocess. Microsyst. **46**, 175–183 (2016)

S. Yu, H.-S. Philip Wong, Modeling the switching dynamics of programmable-metallization-cell (PMC) memory and its application as synapse device for a neuromorphic computation system, in *2010 International Electron Devices Meeting* (IEEE, 2010), pp. 22–23

S. Yu, Y. Wu, R. Jeyasingh, D. Kuzum, H.-S. Philip Wong, An electronic synapse device based on metal oxide resistive switching memory for neuromorphic computation. IEEE Trans. Electron Dev. **58**(8), 2729–2737 (2011)

Y. Zhang, P. Li, Y. Jin, Y. Choe, A digital liquid state machine with biologically inspired learning and its application to speech recognition. IEEE Trans. Neural Netw. Learn. Syst. **26**(11), 2635–2649 (2015)

C. Zhao, B.T. Wysocki, C.D. Thiem, N.R. McDonald, J. Li, L. Liu, Y. Yi, Energy efficient spiking temporal encoder design for neuromorphic computing systems. IEEE Trans. Multi-Scale Comput. Syst. **2**(4), 265–276 (2016)

Reservoir Computing Using Autonomous Boolean Networks Realized on Field-Programmable Gate Arrays

Stefan Apostel, Nicholas D. Haynes, Eckehard Schöll, Otti D'Huys, and Daniel J. Gauthier

Abstract In this chapter, we consider realizing a reservoir computer on an electronic chip that allows for many tens of network nodes whose connection topology can be quickly reconfigured. The reservoir computer displays analog-like behavior and has the potential to perform computations beyond that of a classic Turning machine. In detail, we present our preliminary results of using a physical reservoir computer for performing the task of identifying written digits. The reservoir is realized on a commercially available electronic device known as a field-programmable gate array on which we create an autonomous Boolean network for information processing. Even though the network nodes are Boolean logic elements, they display analog behavior because there is no master clock that controls the nodes. In addition, the electronic signals related to the written-digit images are injected into the reservoir at high speed, leading to the possibility of full-image classification on the nanosecond time scale. We explore the dynamics of the autonomous Boolean networks in response to injected signals and, based on these results, investigate the performance of the reservoir computer on the written-digit task. For a wide range of reservoir structures, we obtain a typical performance of ∼90% for correctly identifying a written digit, which exceeds that obtained by a linear classifier. This work paves the way for achieving low-power, high-speed reservoir computing on readily available field-programmable gate arrays, which are well matched to existing computing infrastructure.

S. Apostel · E. Schöll
Institut für Theoretische Physik, Technische Universität Berlin, 10623 Berlin, Germany
e-mail: schoell@physik.tu-berlin.de

N. D. Haynes
Department of Physics, Duke University, Box 90305, Durham, NC 27706, USA

O. D'Huys
Department of Mathematics, Aston University, B4 7ET Birmingham, UK
e-mail: o.dhuys@aston.ac.uk

D. J. Gauthier (✉)
Department of Physics, The Ohio State University, 191 W. Woodruff Ave, Columbus, OH 43210, USA
e-mail: gauthier.51@osu.edu

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_11

1 Introduction

As demonstrated by the breadth of contributions in this volume, there is great interest in reservoir computing (Jaeger and Haas 2004), from fundamental studies of their properties to using them for a wide range of applications. While many studies use a standard Turing-von Neumann computer to simulate a reservoir computer (RC), physical RCs have the potential to demonstrate beyond-Turing computing and may increase the information processing speed. One challenge in building a physical RC is fabricating a large enough reservoir, which may require 100s to 1,000s of nonlinear input-output nodes, and a large number of interconnects (links) between the nodes.

A highly successful alternative that is capable of operating at high speeds is to realize a RC using a single nonlinear node with time-delay feedback with a large number of 'virtual' nodes determined by the ratio of the delay time to the characteristic time scale of the node. Such platforms have been used for a wide variety of machine learning tasks, such as classifying spoken words (Appeltant et al. 2011) at high speed (Larger et al. 2017), nonlinear channel equalization (Paquot et al. 2012), and classifying serial digital data (Haynes et al. 2015). Unfortunately, this approach requires complex temporal time delaying and multiplexing of the input information (masking) and injection into the loop (Appeltant et al. 2014; Röhm and Lüdge 2018). In principle, this masking could be done in real time using dedicated hardware as has been demonstrated recently (Penkovsky et al. 2018), but most demonstrations to date perform this preprocessing offline thereby slowing down the overall operation rate of the RC.

A different, potentially scalable, approach is to use nonlinear optical micro-ring resonators fabricated on a planar photonic chip (Denis-Le Coarer et al. 2018; Mesaritakis et al. 2015), or arrays of linear micro-ring resonators or swirls where the nonlinearity is provided by the square-law detectors (that is, detectors that respond to the intensity of the light rather than the field) in the read-out layer (Katumba et al. 2018; Vandoorne et al. 2014; Vinckier et al. 2015; Zhang et al. 2014). This technology is currently limited to a small ($\sim$16) number of resonators because of optical loss and the network connectivity is limited by the planar geometry. But this platform may find future application as the quality of photonic chips improves.

Here, we focus on an approach using a commercially available electronic device known as a field-programmable gate array (FPGA), which greatly simplifies the creation of a reservoir in comparison to the photonic approaches discussed above. FPGAs are also much easier to reconfigure in comparison to custom-built boards with discrete logic (Mason et al. 2004; Zhang et al. 2009) (that is, devices using a collection of single-chip logic gates soldered on the board). Rosin (2015) pioneered the application of FPGAs to realize autonomous time-delay Boolean networks (Ghil and Mullhaupt 1985). Here, the FPGA is configured to realize a reservoir, where the FPGA logic elements serve as the network nodes that nominally perform a Boolean operation on the inputs. The term autonomous indicates that the logic elements are not gated by a master clock; they process new edges as soon as they arrive on the inputs to the logic elements. Furthermore, because the network is autonomous, signals

traveling along the links have a finite propagation speed so that the link delays must be accounted for. The autonomous nature of the complex network allows for the possibility of beyond-Turing computation (Ghil et al. 2008). Autonomous Boolean networks on FPGAs have been used previously for reservoir computing and applied to high-speed pattern recognition of digital serial data patterns (Haynes et al. 2015) and high-speed forecasting of chaotic dynamics (Canaday et al. 2018). The FPGA can also operate in the clocked mode, thus realizing a finite state machine and offering the possibility of accelerating the software simulation of RCs. This approach has been applied to channel equalization (Antonik 2018; Antonik et al. 2015; Skibinsky-Gitlin et al. 2018; Yi et al. 2016) and speech recognition (Alomar et al. 2015; Penkovsky et al. 2018).

In this chapter, we describe our preliminary studies on using an FPGA-based RC for performing a classification task: identifying images containing human-written digits (the MNIST task). While RCs are most suited for processing time-dependent signals, performing a classification task on images presents interesting challenges, such as identifying efficient methods for injecting data into the reservoir. We also take this opportunity to study the dynamics of autonomous Boolean networks (ABNs) in response to a perturbation, such as a phase transition as the node in degree varies. This study helps guide the choice of metaparameters for the RC.

In the next section, we present the reservoir concept with an associated mathematical model, describe our experimental studies of the dynamics of ABNs as the network parameters vary in Sect. 3, realize an RC on an FPGA and use it for the written digit (MNIST) task in Sect. 4, and discuss future directions and conclude in Sect. 5 (Apostel 2017). We briefly discuss using FPGAs and describe our experimental workflow in the Appendix.

2 Reservoir Computers Based on Autonomous Boolean Networks

One aspect that distinguishes RCs from other artificial neural networks is that each node is itself a dynamical system. Glass and colleagues (Edwards and Glass 2006) have studied extensively the dynamics of autonomous Boolean networks using a model where node i is described by a continuous variable x_i whose behavior is governed by a first-order differential equation with decay rate γ_i and driven by a Boolean function f_i of the node inputs. The function f_i is defined via a look-up table as described below and can take on essentially any Boolean function consistent with the number of inputs to the node. In the context of our work, the decay rate γ_i models the finite rise-time of the FPGA logic elements due to their input capacitance and inductance and are nominally identical for all nodes with $\gamma_i \sim 2\pi/(0.41\,\text{ns})$. When signals propagating from node j to i experience delay $\tau_{i,j}$, the so-called Glass model can be extended to a set of delay differential equations (Edwards et al. 2007). Delays appear in our FPGA-based system because the signals propagate at a finite velocity

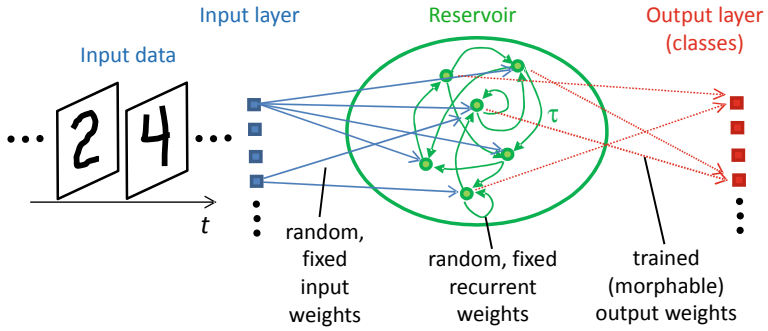


Fig. 1 Illustration of the reservoir computer architecture

along the link wires, where a typical delay time between nearby logic elements is only a few 10's of picoseconds. To purposefully add more delay, we use pairs of series-connected NOT gates along the network links, where we obtain a delay of ∼0.52 ns per pair of gates (Lohmann et al. 2017). We use special care in writing the hardware description language used to configure the FPGA (described in greater detail in the Appendix) to make sure that the logic gates used to create delay are not removed from our design.

Adopting this theoretical approach for reservoir computing with time-delay ABNs, illustrated in Fig. 1, the dynamics of the RC are given by Gauthier (2018)

$$\frac{dx_i}{dt} = -\gamma_i x_i + \gamma_i f_i \left(\sum_{j=1}^{J} W_{i,j}^{\text{in}} u_j(t) + \sum_{n=1}^{N} W_{i,n} x_n(t - \tau_{i,n}) \right), \quad i = 1, \ldots, N \tag{1}$$

$$y_m(t) = \sum_{n=1}^{N} W_{m,n}^{\text{out}} x_n, \quad m = 1, \ldots, M. \tag{2}$$

Here, $u_j(t)$ are the J signals input to the RC, which are connected to the N reservoir nodes with random fixed weights $W_{i,j}^{\text{in}}$, $W_{i,n}$ are the random fixed internal weights of the reservoir, $y_m(t)$ are the M outputs of the RC with trained (morphable) weights $W_{m,n}$. Often, the W's are sparse so that the typical node in-degree (number of inputs) K_i is small. The reservoir embeds the input data to a higher-dimension phase space when $N > J$ because of the nonlinear response of the node, known as dimension expansion and often a key requirement for effective information processing. Because of the time delays, the phase-space dimension of the reservoir is infinite, which can be seen by considering that the initial conditions of the time-delay differential equations must be specified over the real interval of the maximum link delay.

Evaluating f_i is particularly simple when the arguments u_j and x_n are Boolean, but the W's are real. Here, the multiplications and additions appearing in the argument of f_i can be done a single time after the W's are chosen, thereby defining the Boolean look-up table (LUT). Thereafter, only the LUT is used and no further operations

are required. This procedure maps perfectly onto the structure of the FPGA logic elements, which are based on LUTs, as discussed in the Appendix.

Models similar to Eq. (1) have been used to understand the dynamics of time-delay ABNs representing simple genetic regulatory networks are realized on an FPGA, for example, and can capture the extremely long transient behavior observed experimentally (D'Huys et al. 2016; Haynes et al. 2015; Lohmann et al. 2017). While we do not consider solutions to Eq. 1 here, we include it for completeness because it is a good starting point for a theoretical analysis of FPGA-based RCs.

For the classification task considered here, we adjust $W^{\text{out}}_{k,n}$ using a finite-size training data set so that the resulting output properly classifies each input in a least-square sense, known as supervised learning. Specifically, the output weight matrix $\mathbf{W}^{\text{out}}$ is determined by injecting a finite-length input training data set $\mathbf{U}(t)$ and recording the network dynamics $\mathbf{X}(t)$. Here, we assume that the state of the network is sampled at equal discrete time intervals Δt so that the matrices are finite dimensional. Based on these observations, we modify $\mathbf{W}^{\text{out}}$ to minimize the error of the output $\mathbf{Y}$ (the classes) to the expected output $\mathbf{Y}^{\text{expected}}$, resulting in

$$\mathbf{W}^{\text{out}} = \mathbf{Y}^{\text{expected}}\mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T + \beta^2\mathbf{I}\right)^{-1}, \tag{3}$$

where β is a regularization parameter, $\mathbf{I}$ is the identity matrix, and T indicates the transpose. A nonzero value of β ensures that the norm of $\mathbf{W}^{\text{out}}$ does not become large, prevents sensitivity of the training to noise, and improves the generalizability of the RC to different inputs. We stress that $\mathbf{X}(t)$ is a concatenation of the network dynamics over all input data, which is a collection of image data described below in Sect. 4.1. We can also find a solution to Eq. (3) using gradient-descent methods, which are helpful when the matrix dimensions are large. Gradient-descent routines are readily found in modern toolkits developed by the deep learning community, and they can operate at high speed using graphical processing units. Another approach is to use recursive least-squares.

3 Dynamics of Random Autonomous Boolean Networks

Before discussing the performance of the RC applied to the MNIST classification task, we explore the dynamics of time-delay ABNs realized on an FPGA for different network parameters. We seek to identify a phase transition from ordered to disordered (chaotic) behavior and compare our observation to the prediction for clocked Boolean networks. Identifying the location of the phase transition is important because the performance of the RC is expected to be highest near the border of the transition from order to chaos (Yildiz et al. 2012).

3.1 Dynamics of Clocked Random Boolean Networks

There exists a considerable literature on the dynamics of clocked random Boolean networks, and we briefly summarize some of the findings here. A clocked random Boolean network (RBN) is typically taken to consist of N nodes each of which receives exactly K inputs. The Boolean node function f_i is defined in terms of a random LUT, where the probability of an entry in the LUT taking on the value 0 (1) is p $(1 - p)$ and is often called the node bias (Derrida and Pomeau 1986; Flyvbjerg 1988; Luque and Solé 2000).

For a clocked RBN with random and fixed (quenched) functions and connections, the network is a deterministic system with a finite number of states (a finite-state machine) and must show periodic behavior with a maximum period of 2^N. Different network behaviors are observed as K and p varies. One behavior is called *ordered* when the dynamics go to a fixed point or to a low-period periodic orbit. Another behavior, often called *chaotic*, is when the period approaches its maximum value and the specific periodic orbit followed is sensitive to a change in the initial value of a single network node. Of course, chaos cannot exist in an RBN because all behaviors are periodic, but the term is suggested because of the sensitivity of the dynamics on the initial conditions. Using an annealed network approach (Derrida and Pomeau 1986), the order-chaos transition is given by

$$K_c = \frac{1}{2p(1-p)} \tag{4}$$

in the limit $N \to \infty$ and shown in Fig. 2.

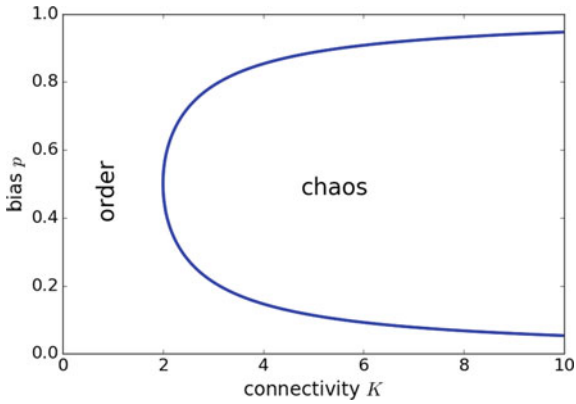


Fig. 2 Order-chaos phase transition in a clocked RBN

3.2 *Dynamics of ABNs on an FPGA*

Guided by the results on clocked RBNs, we investigate experimentally the dynamics of random ABNs on an FPGA. Here, we measure the Booleanized state of each node at discrete (clocked) time intervals using the finite-state machine described in the Appendix. We choose a subset of networks from the full range of possibilities described by Eq. 1 so we can make a comparison to the previous work on clocked networks. In particular, we consider networks with $N = 64$ each having exactly K inputs and outputs so that $W_{i,n}$ appearing in Eq. 1 is quite sparse. Here, we consider only $K \leq 4$. Each node is assigned a random and fixed Boolean function f_i with bias p under the restriction that $f_i = 0$ when all the inputs are zero so that the network is in the quiescent state (no self-oscillation) when the FPGA is first turned on or reset to this value. This restriction limits the bias to $p_{\max} = 1 - 1/2^K$, but is advantageous for realizing an RC because it results in a well-defined initial state of the reservoir. In addition, to slow down the network dynamics (Canaday et al. 2018) to better match the rate at which we can inject data into the RC (see the next section), and to more closely match the clock RNB theory, we use a nominally constant link delay $\tau_{i,n} = 12.8 \pm 0.5$ ns. Here, the link time delay is set by using pairs of series-connected NOT gates, where each pair of gates causes a delay of ~0.52 ns (Lohmann et al. 2017). For this study, we investigate the dynamics of 6,000 randomly chosen networks.

The initial condition of the network is set using an OR logic gate at the output of each node as illustrated in Fig. 3. Initially, the state of all nodes is set to 0 to reset the network, and then the desired initial condition is set during a 5-ns-long window and passed to the link delay lines. This is accomplished using the OR gate, where $u_j(t)$ appears at the output while the node is in the state 0. Once signals appear at the input of the node from other network nodes (after a delay time $\tau_{j,s}$), any signals related to $u_j(t)$ still being injected into the node will be 'blended' with the incoming node signals via the OR gate.

In general, setting the initial conditions to a random binary value will destabilize the initial stable network state where all nodes are equal to 0, giving rise to a transient that can last for a time beyond our data-collection window for $p \sim 0.5$. After initializing the network, we measure the state of each node every 5 ns for 200 samples (1 μs total record length), and transfer the data to a computer for analysis. The network is

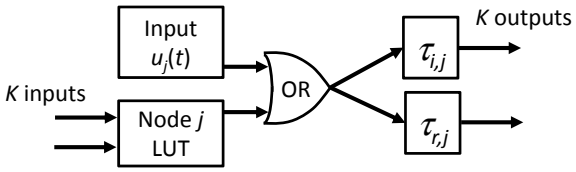


Fig. 3 Using an OR gate on the output of node j to inject the input data for the node, illustrated here for $K = 2$. The signal emanating from the OR gate passes to the delay-line links to nodes i and r

reset to the 0 state at the end of data collection for initial condition. For each network realization, we use 1,000 randomly chosen initial conditions and repeat the experiment 50 times for each initial condition. We do not attempt to study the duration of the transient beyond 1 μs because it is not relevant for reservoir computing although we note the extremely long (second to minutes) chaotic time transient times have been observed in simple Boolean networks with similar bias (Haynes et al. 2015).

One issue that arises when using an FPGA is that a randomly assigned node LUT may have the same output (0 or 1) regardless of the inputs. In this case, the optimizer within the compiler that converts the *Verilog* hardware description language to a bitstream to configure the FPGA removes such nodes from the network. This is more pronounced for low and high biases and the actual network synthesized on the chip has fewer than 64 nodes when the network is pruned by the optimizer (typically, only a few nodes are pruned). In the figures presented below, we use different color symbols to indicate networks that have been pruned by the compiler. This behavior does not change the network dynamics or time scales because such a node would have been inactive. Nodes with high bias ($p \sim 1$) may also be inactive, but we already exclude nodes with high bias as discussed above.

We observe a wide range of network dynamics as we vary p and K. Again motivated by the research on clocked RBNs, we initially search for the network to settle down to a fixed-point behavior where all network nodes take on a constant value (frozen dynamics) after a transient time. While we only measure the network dynamics at discrete times, it is unlikely that fast oscillations in between the measurement times would escape unnoticed. We search for fixed-point behavior by measuring the Hamming distance between the state of the network at time t_0 with its values at all later times up to the limit of our 200 samples. Here, the Hamming distance between two Boolean vectors $\mathbf{A}$ and $\mathbf{B}$ is defined as

$$H(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^{N} A_i \oplus B_i. \tag{5}$$

Figure 4 shows the number of time steps required for the dynamics of the ABN to reach a fixed point (network realizations where a fixed point is never observed are not shown). We also show the probability for a network to reach a fixed point. Qualitatively consistent with the predictions shown in Fig. 2 for a clocked RBN with the same network structure as our ANB, we observe longer transients as p approaches 0.5 from above and below, and the range of biases that do not show fixed-point behavior widens as K increases. Of course, we are only considering fixed-point behavior and so these results only suggest the location of the order-chaos boundary, although we find that chaotic behavior tends to be observed when the probability for observing a fixed point drops near zero around $p = 0.5$

Over the range of p where the probability of observing a fixed point is nearly zero (the region around $p \sim 0.5$ where the green line is near zero), we observe fast, complex behavior after an initial short transient that depends sensitively on the initial conditions, likely a manifestation of chaos in the ABN (Zhang et al. 2009) as

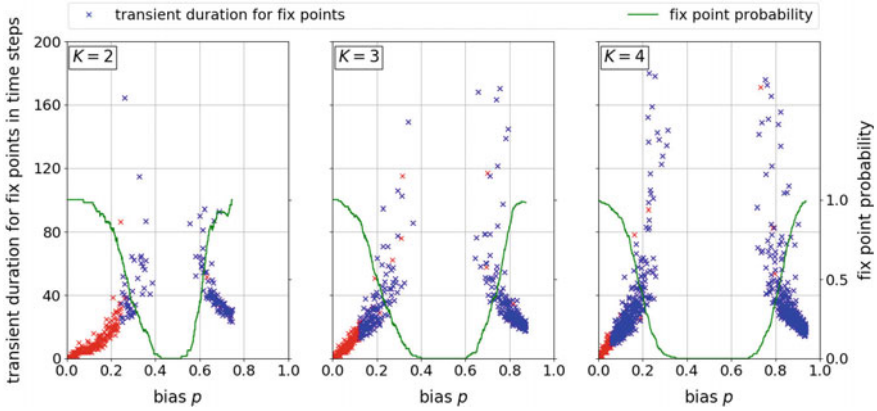


Fig. 4 Length of transient for cases when the ABN reaches a stable fixed point for different network parameters. Blue symbols indicate unmodified networks and red symbols indicate optimizer-pruned networks. The green solid line shows the probability that a network reaches a stable fixed point

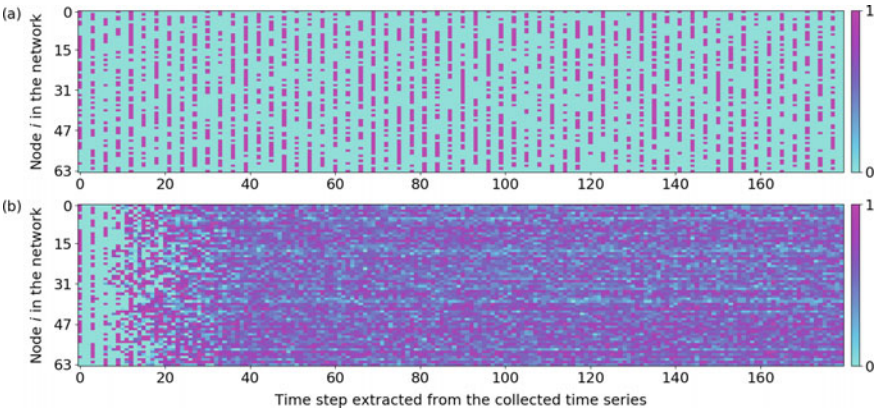


Fig. 5 Dynamics of a Boolean networks with $N = 64$, $K = 4$, and $p = 0.55$ for a clocked network simulated numerically and b an autonomous network on the FPGA averaged over 50 experimental runs with the same initial conditions for each. Boolean 0 (1) is colored cyan (magenta). This network realization has no pruned nodes

shown in Fig. 5b. Here, the dynamics is so fast that our finite sampling rate is too low to faithfully record the dynamics. As seen in Fig. 5a, the behavior predicted by a discrete-time model using the same LUT and initial conditions is similar to our observations of the ABN on the first few time steps, but disagrees thereafter. This disagreement is perhaps not too surprising because the discrete map is a finite-state machine and hence cannot display chaos (the dynamics eventually has to repeat). The fact that we observe chaos gives a hint that an RC realized with an ABN reservoir on an FPGA could display beyond-Turing computation, although observing chaos is not a necessary or sufficient condition for beyond-Turning computation.

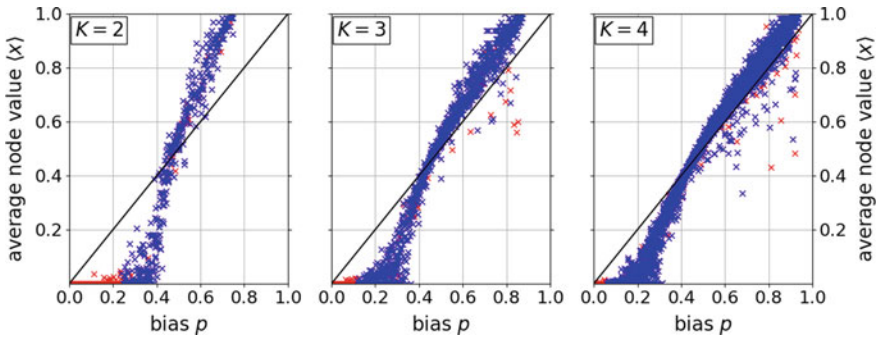


Fig. 6 Average value of the network nodes for different network parameters

Close inspection of Fig. 5b reveals apparent horizontal stripes, indicating that the average value of the nodes takes on different values. For this data set, the average value of the nodes range from $\langle x_i \rangle = 0.36$ to 0.74, with an average for all nodes of $\langle x \rangle = 0.57 \pm 0.07$, which is consistent with the value of p. We currently do not have an explanation for the appearance of these patterns.

We find that another qualitative measure of the order-chaos boundary can be obtained by measuring $\langle x \rangle$ for different network parameters as shown in Fig. 6. Here, we perform the average from time steps 150 through 200 to minimize the effects of the transients. At $p = 0$, we expect that $\langle x \rangle = 0$ because all nodes in the network are inactive regardless of their input. As p increases, we see that $\langle x \rangle$ remains near zero until an apparent transition to complex behavior indicated by a rapid rise in $\langle x \rangle$. The value p at this transition is smaller as K increases, consistent with the phase-transition behavior of clocked RBNs shown in Fig. 2. We also show a line indicating $\langle x \rangle = p$, which is expected for a RBN with $N, K \to \infty$. As K increases, the average node value approaches this line, which is an interesting result because our network is quite different from an RBN. Specifically, we are using an ABN, we restrict the node LUTs so that the all-zero state is a solution, and N and K are not all that large.

4 Reservoir Computing with ABNs on an FPGA

The experiments described in the previous section demonstrate that an ABN-based reservoir can be constructed on an FPGA whose behavior can be adjusted from ordered to chaotic simply by adjusting the node bias and in-degree. Operating near or within the chaotic domain demonstrates that the reservoir embeds the dynamics in a high-dimensional phase space, one crucial characteristic needed for reservoir computing. It is also widely believed that the reservoir must display other characteristics when data is input to the reservoir, including (Jaeger and Haas 2004):

- **Separation of different input states**. Input data in different classes should have distinct output dynamics.
- **Generalization of similar inputs to similar outputs**. Different input data within a class should have similar output dynamics.
- **Fading memory**. The output dynamics should be correlated with the input dynamics over a time known as the *consistency window*, but this temporal correlation should fade.

Based on these properties, an RC is well suited for classification or prediction based on correlations in data over time because it inherently has temporal memory. For classification tasks on static image data such as the MNIST data set, the correlations are inherently spatial. Thus, we break the image up into sub-images and feed them rapidly into the reservoir (at the 200 MHz limit imposed by our write and read procedure) to create spatial-temporal correlations. While this process is a bit contrived, it may be well suited for processing high-speed video imagery data where the data is usually generated in a progressive scan of the image.

In the next section, we introduce the MNIST classification task and modifications we make to the input data to make a better match to our reservoir characteristics, then inject this data into ABNs to verify the behaviors described above. Finally, we describe the performance of an FPGA-based RC for the classification task.

4.1 The MNIST Classification Task

The MNIST database (MNIST 2021) of handwritten digits is a collection of 60,000 training samples and 10,000 test samples of handwritten digits from 0 to 9. Each written digit in the original data set, referred to as NIST after the U.S. federal agency that collected the data, is represented by black-and-white 20 × 20-pixel, which was modified (MNIST) by centering the image and padding with white space, resulting in a gray-scale 28 × 28-pixel image. Figure 7 shows four sample MNIST images for each written digit.

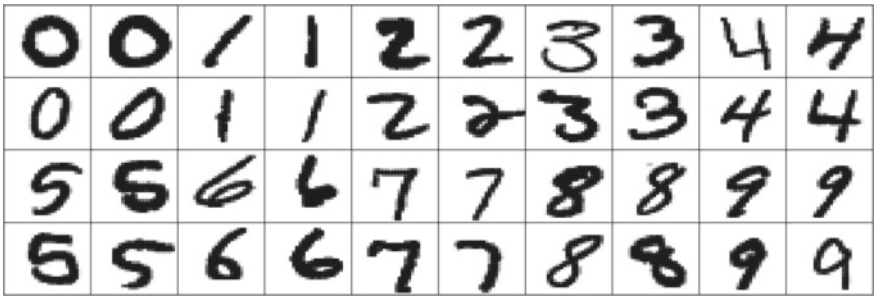


Fig. 7 Four examples for each digit of the MNIST data set shown as gray-scale images

As mentioned above, we seek to inject sub-images into the reservoir to exploit its fading memory. There is a trade-off in using too many sub-images because the time interval for injecting data may be longer than the reservoir consistency window. We also want to use sub-images with no more than pixels than reservoir nodes. To reduce the parameter space, we choose to map each pixel to a single reservoir node and thus to have a sub-image size of 64 pixels. Thus, in this case, the input layer is the sub-image and we use a restricted connectivity of the input layer to the reservoir: each input node only connects to a single reservoir node. For cases when there is not enough image data to fill a full 64-element input data vector, we pad the data with zeros. When the reservoir is pruned by the *Verilog* compiler, the corresponding pixel is not mapped into the reservoir and hence this information is lost.

For most of the studies presented below, we reduce the MNIST images to 16×16 pixels distributed over 4 sub-images. We tested many methods for creating sub-images, such as vertical or horizontal image segments, or using random masks. We find no statistically significant differences in the performance of the RC for the MNIST classification task for different protocols for sub-images creation; the data shown below uses vertical image segments.

The data is injected into the network using the same method described in Sect. 3.2 above using an OR gate at the output of each node (see Fig. 3) controlled by our data write/read FSM described in the Appendix. Here, the data is loaded into the links, which have a nominal propagation delay of 12.8 ns. A sub-image is input every 5 ns (200 MHz input data rate) and hence some of the data is 'blended' with the reservoir dynamics if the total image time exceeds the link delay time. For example, it takes $4 \times (5\,\text{ns}) = 20\,\text{ns}$ to inject a full image with 4 sub-images so that the first sub-image is fully blended with the reservoir dynamics, the second sub-image is blended with the reservoir dynamics for ~2.2 ns, while the last two sub-images are presented fully for 5 ns. We did not realize this shortcoming of our method until after substantial data collection. This represents a limitation of this initial study and the error rates of the RC presented in Sect. 4 below are likely lower bounds. This limitation will be addressed in future studies.

Figure 8 illustrates the steps involved to create the reduced image. This first step is to determine an image 'focus,' which is an image coordinate determined by an average of the pixel coordinates weighted by their respective gray-scale value. The intersection of the solid-blue lines shown in Fig. 8a shows the image focus, which is not necessarily the center of the image indicated by the intersection of the black dashed lines. Next, we determine exterior white pixels to remove the excess padding. We then create a grid of $q \times q$ pixels within this region centered on the focus with $q = 8 - 20$, but $q = 16$ typically. A resulting pixel is labeled by filling each pixel like a 'bucket' by adding the value overlapping area $\times$ gray-scale weight (0–255) from each original MNIST pixel to the new grid pixel. The new pixel will be set to Boolean 1 (0) if the filled value reaches (does not reach) a threshold set at approximately the half-gray-scale point. This process is depicted in Fig. 8b with Boolean 1 shown as blue and the bucket gray scale shown as an underlay.

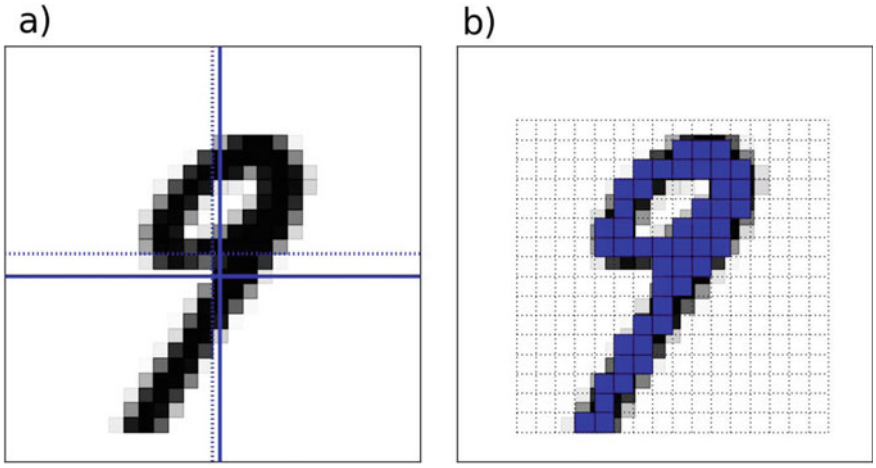


Fig. 8 **a** Determining the focus of the MNIST image. **b** Creating a reduced image shown here for the case of 16×16 pixels

4.2 Dynamics of ABNs with Data Injection: The Consistency Window

As mentioned above, an important feature of an RC is to separate data from different classes and to give similar output for data from the same class. Even if the reservoir is operating in the chaotic state, the transient dynamics can still be used for classification when the output data is restricted to the consistency window (Haynes et al. 2015; Uchida et al. 2004).

Haynes et al. (2015) proposed using the Hamming distance for Boolean data given in Eq. (5) as a measure for the consistency window. We follow the same approach here. Figure 9 shows examples of the dynamics of the average Hamming distance, where the bias is set in the ordered regime (panel a) so that the dynamics evolves to a fixed point for all inputs and the other for a bias in the chaotic regime (panel b). Here, a random Boolean initial condition is injected into the network for a single 5-ns-long period, which destabilizes the network dynamics. This is repeated many times for the same and different initial conditions. The consistency window w quantifies the interval over which similar and different inputs can be distinguished.

Also apparent in this plot is the fading memory property of the ABNs. Beyond the consistency window, the output state is uncorrelated with the input state in both the ordered and chaotic regimes.

We repeat this experiment for a large number of randomly chosen reservoirs as a function of network parameters. We determine w by finding the intersection point of lines fit to the early part of the data. This procedure can result in values greater than the 200 steps over which we collect data when the slopes are shallow. As seen in Fig. 10, the consistency window sharply peaks for bias values at the ordered-chaotic

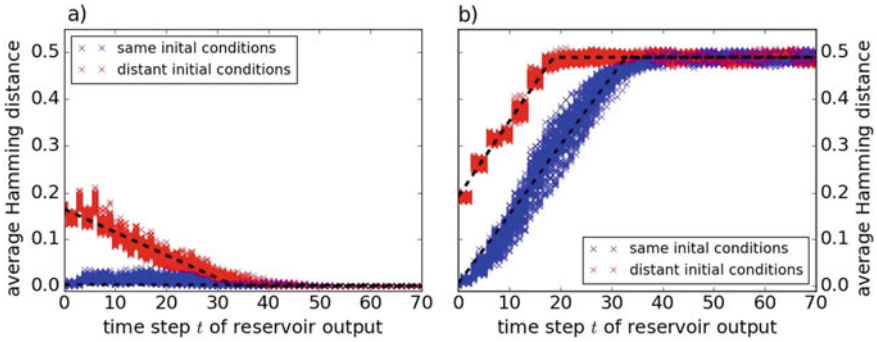


Fig. 9 Temporal evolution of the normalized Hamming distance using two randomly selected data sets with 64 pixels injected into the reservoir for 5 ns. The blue symbols indicate repeated injection of the same data and finding the Hamming distance for each trial, and the red symbols indicate the difference between this data set and a randomly chosen second data set for **a** $K = 2$ and $p = 0.387$ and **b** $K = 4$ and $p = 0.476$. The black dashed lines are fits of the data to straight lines, the intersection point of which are used to indicate the end of the consistency window, about ∼30 time steps for both sets of data

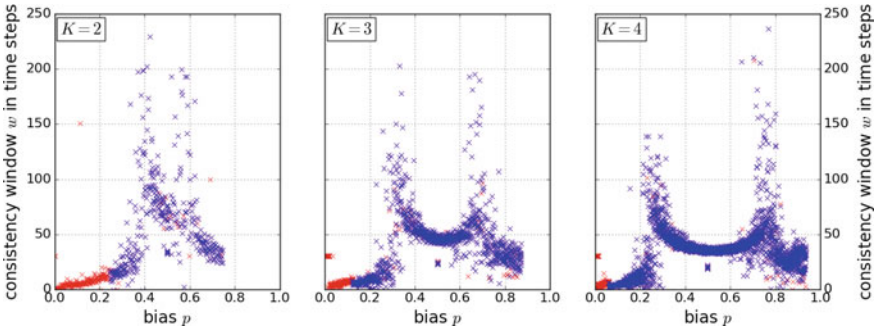


Fig. 10 Average consistency window for as a function of network parameters. Blue symbols indicate unmodified networks and red symbols indicate optimizer-pruned networks

boundary (see Fig. 4) and approaches zero for small p. In the chaotic regime, w takes on it smallest value around $p = 0.5$ of ∼50 time steps for $K = 3$ and ∼35 time steps for $K = 4$.

We also explore the dynamics of the network where all nodes execute the XOR function, which corresponds to $p = 0.5$. This is motivated by the previous work of Haynes et al. (2015) who realized an RC for serial digital data pattern recognition using a single XOR node with self-feedback over two links with different time delays. The results of this study are contained in the tight cluster of blue data points in Fig. 10 at $p = 0.5$ and $w \sim 35$ for $K = 2$, $w \sim 25$ for $K = 3$, and $w \sim 20$ for $K = 4$. Given that these point clusters are below the 'U'-shaped band of points appearing above the point cluster, corresponding to the random networks, indicates that the random

networks outperform the XOR network with regard to the consistency window, which may improve the performance of the RC.

The results presented in this section demonstrate that a random ABN satisfies the criteria of separation, generalization, and fading memory that is believed to be required for reservoir computing.

4.3 Realizing an RC for the MNIST Classification Task

In this section, we report on the performance of the RC on the MNIST classification task using the following procedures. We use the first 42,000 images from the MNIST data set and reduce the images to 16×16 pixels using the procedure given in Sect. 4.1. We use less than the total MNIST image database size to reduce the computational cost of our experiments and subsequent analysis. The output of the reservoir for each written-digit image is a series of 200 time steps for each of the 64 reservoir nodes. The resulting data for the output state of the network is transferred to a personal computer, then to the Open Science Grid (see the Appendix) for training the RC output weights. Even in binary format, the file size for the network output state (42,000 images $\times$ 32 bytes/image $\times$ 200 time steps) is 672 MB for a single reservoir realization. For this reason, we only choose 800 random networks when measuring RC performance as a function of network parameters rather than the 6,000 used in Sect. 3.2 above.

These data are used to determine $W^{\text{out}}_{m,n}$ using a ridge-regression regularization parameter $\beta^2 = 10^{-3}$. The 10-element class output vector $\mathbf{Y}^{\text{expected}}$ has a value of 1 for the corresponding image and -1 for the remaining elements. The output data for each image is reshaped into a single feature vector with a total length $64 \times 200 = 12,800$ elements with a total feature matrix $\mathbf{X}$ in $\mathbb{R}^{12,800\times 42,000}$; performing the matrix inversion given in Eq. 3 with this data set pushes the limits of our accessible computer hardware in terms of memory to store the matrices and computation time.

The classification performance for each individual written digit is evaluated using the cross-validation method. Here, our selected data set of 42,000 written-digit images is split into 5 equal-size groups of 8,400 images each. The training of $W^{\text{out}}_{m,n}$ is repeated 5 times using 4 groups for the training and the remaining for testing the performance of the RC using this 'unseen' data set. After repeating this procedure for all groups, the percent of correct classifications for each written digit is calculated (that is, the fraction of classification attempts that was completed correctly). This procedure is then repeated for each of the 800 random reservoirs and for each network parameter. The performance P is defined as the average of the percentage correct score for each written digit.

4.3.1 Exploring Fading Memory in the RC

In the previous section, our observations indicate the presence of fading memory (see Fig. 9) as quantified by a single metric given by the consistency window. A

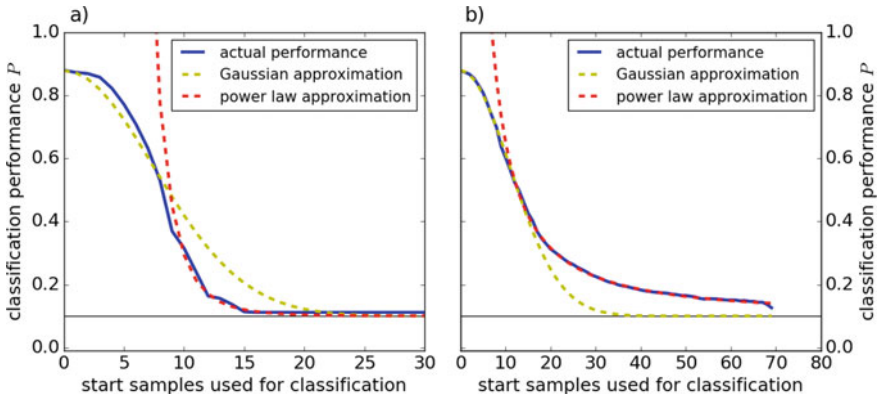


Fig. 11 Fading memory of two example random reservoirs with **a** $K = 4$ and $p = 0.912$ and **b** $K = 4$ and $p = 0.553$

more nuanced measure for how the fading memory affects the performance on the MNIST task is obtained by varying the range of time steps used in the classification task (that is, the location and length of the block of data $\mathrm{X}(t)$ used during RC training). We expect that the initial data is highly correlated with the input data and hence it contains limited new information that can be used for classification. At the other extreme, there is a loss of correlation at later times as the network short-term memory fades. We adjust the location and length of the data block by repeating the classification task using data only from a start time step in the range $0 < t_{\text{start}} < 70$ until a final time step of $t_{\text{end}} = 70$ to determine the classification performance P. That is, X only has data from observation times t_{start} to t_{end}. At the final time step, there is almost no data recorded from the reservoir, explaining the small dip apparent for the last data point of Fig. 11b).

Figure 11 shows two characteristic dependencies of the performance as a function of t_{start}. The performance starts high, followed by a rapid decrease, followed by a long decaying tail. To compare performance across the random networks, we find empirically that the initial high-performance section is well-fit with a Gaussian distribution, which characterizes the duration of 'short-term memory.' The longer tail is well-fit by a power-law of the form

$$P(t_{\text{start}}) = c(t_{\text{start}})^{-\alpha} + 0.1 \tag{6}$$

and quantifies the 'long-term' memory of the reservoir. As $t_{\text{start}} \to \infty$, $P \to 0.1$, corresponding to a random guess of the 10 written-digit classes. As seen in the figure, the fits to these two functions is reasonable.

From the Gaussian fit, we extract a short-term memory coefficient σ equal to the $1/e$ half-width of the Gaussian function in units of start samples used for classification and gives a characteristic time scale for the short-term memory. Already after a few time steps, the performance drops substantially, for which the output data contains

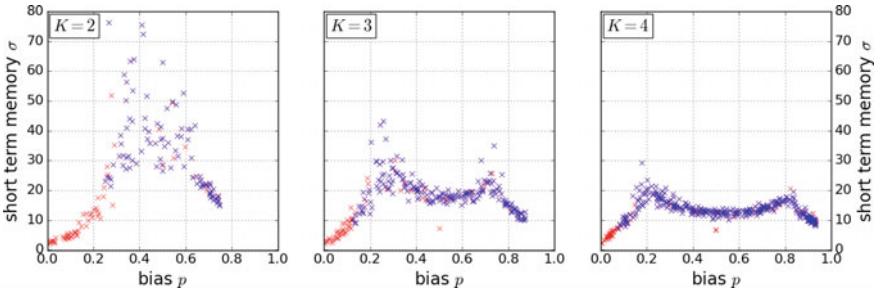


Fig. 12 Short-term memory of the RC for different reservoir parameters

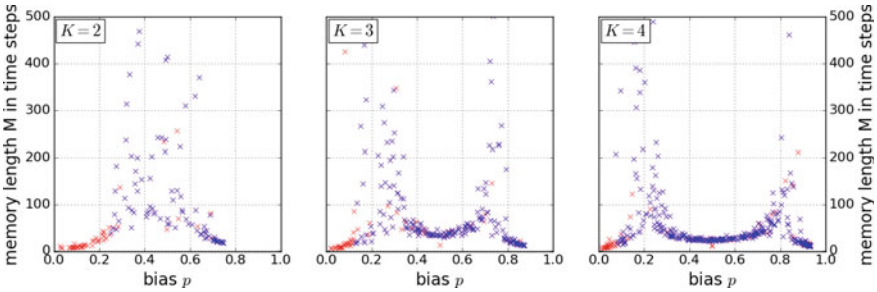


Fig. 13 Long-term memory of the RC for different reservoir parameters

only partial information of the input image (recall we insert data into the network over 4 time steps for the 16 × 16-pixel images). Figure 12 shows the short-term memory as a function of the network parameters. Similar to the time to reach a fixed point shown in Fig. 4, the short-term memory increases near the ordered-chaotic transition boundary, but remains high throughout the chaotic regime for $K = 2$. For $K = 2$ and 3, there is less scatter in the data, the domain of long memory time increases with K, the memory scale peaks at the transition boundary, and there exists a minima in the memory time at $p \sim 0.5$ between the peaks, although the dependence is fairly flat. The longer short-term memory time for $K = 2$ may suggest a reason for the slightly higher best performance found for this network connectivity described below in Sect. 4.

From the power-law fit, we extract a long-term memory coefficient M defined as the time at which the performance drops to 0.11. This is a measure of the time needed for almost all information to vanish from the system. As seen from Fig. 13, M is peaked near the ordered-chaotic boundary, with a larger separation between the two boundaries as K increases, and takes on a minimum value in the chaotic domain when $p \sim 0.5$. Interestingly, it is possible to create networks with long memory (10's of samples corresponding to >50 ns) even for reservoirs with biases as low as 0.1 when $K > 2$, likely due to the large number of recurrent loops in the network and the relatively long link time delays.

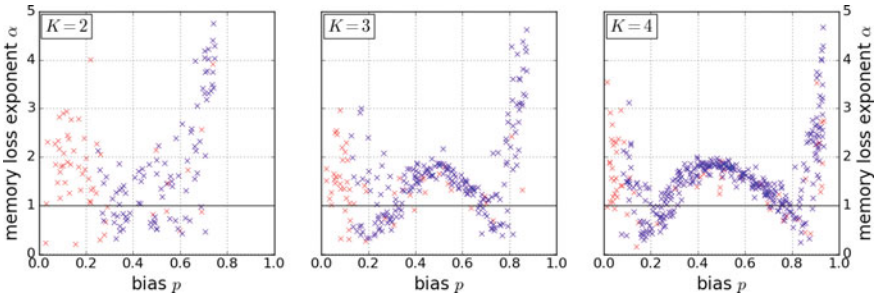


Fig. 14 Power-law exponent for long-term memory in the RC for different network parameters

It is not possible to give an appropriate scale for the long-term memory because it appears to be well described by a power law, which has no scale. As a substitute, we determine the power-law exponent α for different network parameters as shown in Fig. 14. For a power-law with a negative exponent ($\alpha > 0$ in Eq. 6), its integral is finite for $\alpha > 1$ and will diverge otherwise. The exponent becomes large for $p \to 0$ and $p \to 1$, which is due to fact that these networks reach a fix point within a few time steps and almost all nodes are in the inactive (active) state for low (high) bias. For p close to the phase transition, $\alpha < 1$, suggesting the existence of substantial long-term memory and hence long-term retention of information in the network. For $p \sim 0.5$, $\alpha \approx 2$ for all K, demonstrating that the long-term memory is nearly absent ($\alpha > 1$) and fairly insensitive to K, whereas the short-term memory is more sensitive to this parameter.

4.3.2 RC Performance for Different Reservoir Parameters

The previous sections demonstrate that an FPGA-based RC shows a fairly strong dependence of the consistency window and memory on network parameters K and p. Based on the vast literature on RCs that states it is important to optimize the consistency window and fading memory, we find a surprising lack of dependence of the RC performance on K and p for the MNIST task as seen in Fig. 15. As in our previous plots, the red symbols correspond to networks whose inactive nodes are pruned by the *Verilog* compiler and optimizer. These pruned networks tend to have lower performance; the lowest performer outliers have only a few active nodes and hence there is little information available for classifying the written digits. For fully realized networks with no pruning, indicated by the blue symbols, the performance is only weakly dependent on K and p, although the domain of well-performing networks increases with K. Interestingly, we obtain good performance throughout the chaotic domain and well into the ordered domains (Yildiz et al. 2012).

Zooming in on the high-performance, un-pruned networks (Fig. 16), some structure is evident with the largest spread of 4% for $K = 2$. Evident is a slight peaking in performance near the ordered-chaotic boundaries, although the spread in the data

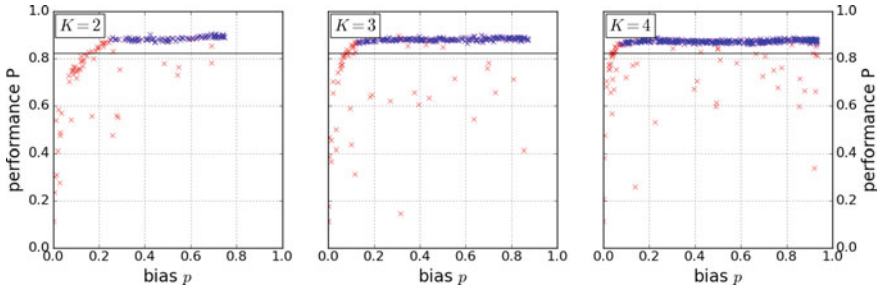


Fig. 15 The classification performance for various network parameters. The black line at $P \approx 0.82$ indicates the score of the linear classifier for comparison. Blue symbols indicate unmodified networks and red symbols indicate optimizer-pruned networks

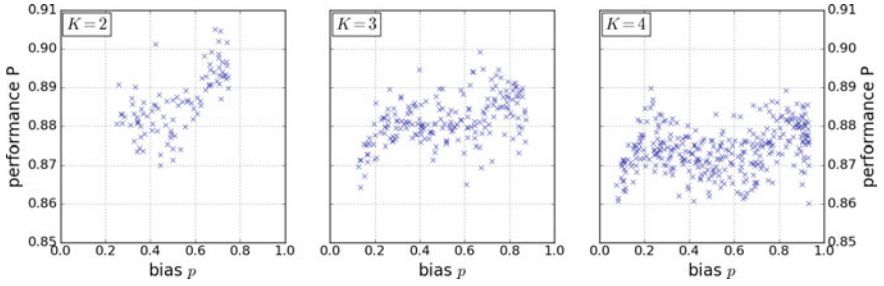


Fig. 16 Higher resolution plots of the data shown in Fig. 15, but only including the results from full (un-pruned) networks

is comparable to the peak sizes. Even within the middle of the chaotic domain, we find some networks that perform nearly as well as the best performers near the ordered-chaotic boundaries, especially for $K = 3$ and 4.

4.3.3 RC Performance Dependence on Output Data Sample Size

We find that it is not necessary to use the data characterizing the network dynamics $X(t)$ collected over the entire 200 time steps. As above, we use the cross-validation method but with the smaller recorded network data. Figure 17 shows the dependence of the performance on the number of time steps used in the training where we always start with the first time step. The performance saturates only after ∼10 time steps (corresponding to a interval of only 50 ns). Note that all nodes are connected to the input layer and hence are activated immediately, but information injected to a node does not fully spread fully throughout the network because of the substantial link-time delay (∼13 ns). This suggests that the recurrent connections, which allow for arbitrarily long loops in the reservoir, may not be as important for the MNIST classification task.

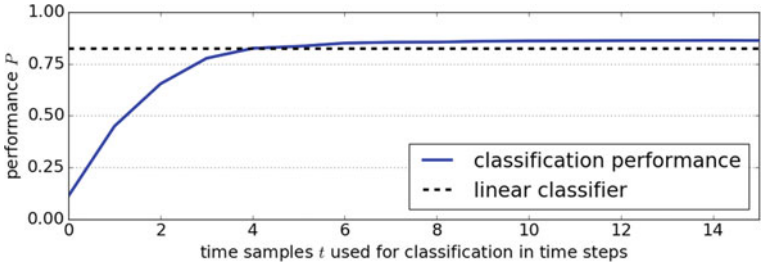


Fig. 17 Performance score as a function of the length of the data record used for training the RC for $K = 2$ and $p = 0.222$. The black dashed line represents the performance of a linear classifier

Table 1 Observed RC best performance on the MNIST task

K	P_K (%)	Optimum p
2	90.5	$p = 0.688$
3	89.9	$p = 0.668$
4	89.0	$p = 0.229$

4.3.4 Best Results

After an exhaustive search over network parameters, we find the highest performance over all p, finding the results given in Table 1. For comparison, we find that a linear classifier (no reservoir) has $P \approx 82.4\%$ for all K and thus the RC has better performance in all cases. For a linear classifier with the full 28×28 image, LeCun et al. (1998) obtained $P = 88\%$. Our results with a compressed image still outperform a full-image linear classifier. Furthermore, the image data is processed by the reservoir within $1\,\mu s$, suggesting high-speed prediction could be possible if the weights are applied in real time on the FPGA (after off-line completion of the training of the output weights), which we will explore in future studies. The best performing networks occurred for p near the ordered-chaotic transition, although the dependence on p is weak as discussed in Sect. 4.3.2 above.

The performance in a typical classification experiment is about 90%. In particular, the classification results were quite different for each written digit. Figure 18a shows the average classification accuracy P_i for each written digit i over all experiments. When the classification fails (that is, the actual digit injected into the RC is mis-classified), the RC will incorrectly classify the data as a different digit. Figure 18b shows the probability that the RC selects a digit for the case when it mis-classified the image.

The digit '1' is classified with the highest accuracy with a score of $P_1 = 94.6\%$, while digit '5' has the worst result of $P_5 = 76.0\%$, nearly a 20% difference from the best digit. Surprisingly, digit '1' is selected with much higher probability when the RC fails to correctly classify an image. The digit selected least during a mis-classification are '0,' '2,' and '6.'

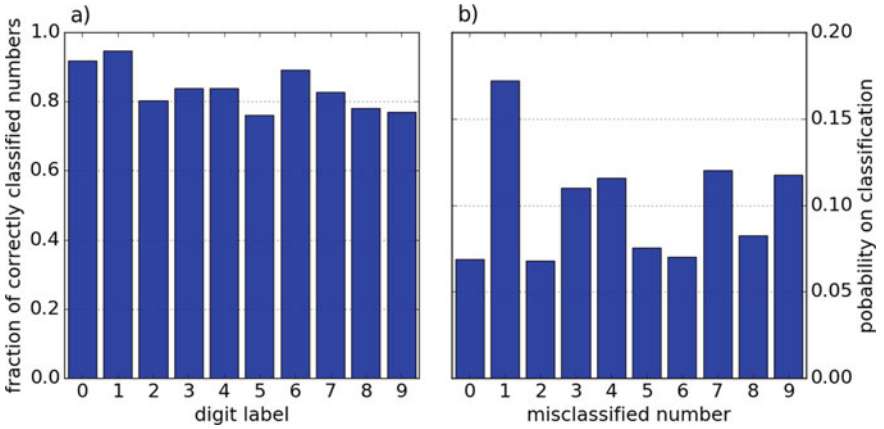


Fig. 18 **a** Average classification accuracy over all experiments and **b** mis-classification of the written-digit images

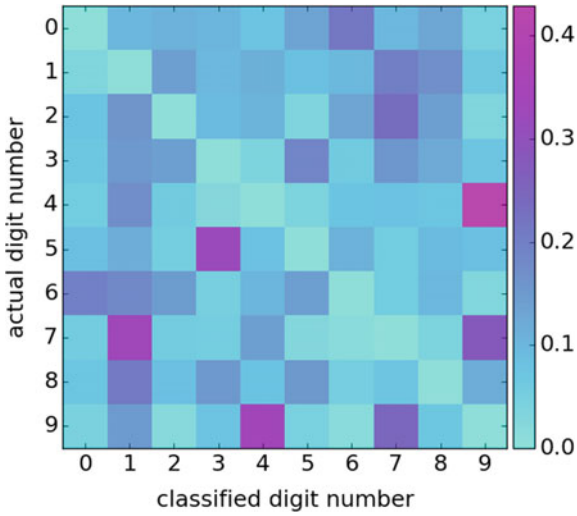


Fig. 19 The confusion matrix visualizing the error for each digit

To go deeper into understanding the errors, Fig. 19 shows the confusion matrix quantifying how often a certain digit is mis-identified as a function of the input digit. Here, the rows of the matrix are normalized to one; in each row, a blue color indicates that this digit is hardly chosen while magenta pixels contribute the most to the error probability.

From the confusion matrix, we see that if a digit is mis-classified as another digit with high likelihood, the same can be said about the other digit. The most obvious mis-classification pairs are $4 \leftrightarrow 9$, $7 \leftrightarrow 9$, $1 \leftrightarrow 7$, and $3 \leftrightarrow 5$. Given the similarities of the strokes to write these numbers, their confusion is sensible. However, it is surprising that the pair $3 \leftrightarrow 8$ is not pronounced.

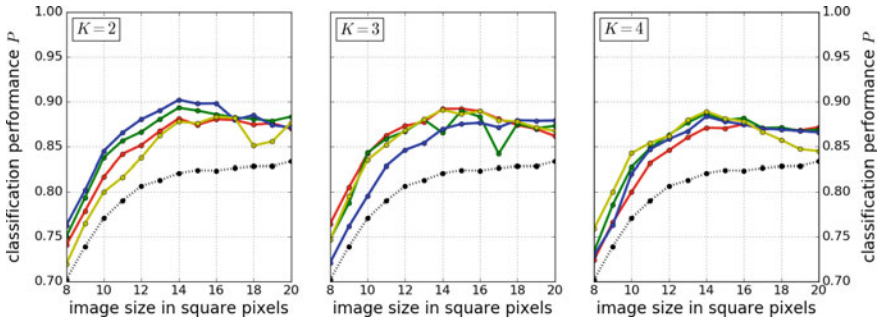


Fig. 20 Classification performance variation with the number of pixels representing the written digit. The colored lines are for 4 randomly chosen reservoirs, while the black line is for the linear classifier. The parameters are as follows: $K = 2$, $p = 0.5$ (red), 0.566 (green), 0.719 (blue), 0.352 (yellow); $K = 3$, $p = 0.758$ (red), 0.521 (green), 0.170 (blue), 0.482 (yellow); and $K = 4$, $p = 0.176$ (red), 0.244 (green), 0.297 (blue), 0.743 (yellow)

We now turn to investigate the influence of the image size on the performance. For images compressed to a very small number of pixels, there is loss of information because the coarse representation will effectively merge features in the written digit. On the other hand, once the image has $\sim 13 \times 13$ pixels, some of the data input to the reservoir is lost because of the shortcoming of the way in which we inject data into the reservoir as discussed above in Sect. 3.2. Figure 20 shows the performance of four different randomly chosen reservoirs for each connectivity K. In addition, we show the performance of the linear classifier for the same data sets.

We see that the FPGA-based reservoirs always outperform the linear classifier for all image sizes. As expected based on increased image resolution, the performance of the linear classifier increased monotonically but there is a noticeable decrease in the slope beyond $\sim 12 \times 12$-pixel image size. On the other hand, the RC classifier performance peaks around a $\sim 14 \times 14$-pixel image size and then decreases somewhat. This decrease is likely due to the fact that only part of these images are input to the reservoir by our method for data injection as mentioned above and discussed in Sect. 3.2, a limitation we will address in future studies. However, for all cases, the performance is above the linear classifier.

4.3.5 Parallel RCs

One method to possibly improve the performance of our reservoir computer is to use multiple reservoirs, which we explore here in some preliminary studies. Similar to deep learning artificial neural networks, a hierarchical structure can allow for forecasting dynamical systems with different time scales or to focus the network on different temporal or spatial features (Jaeger 2007). Also, a serial cascaded of RCs improves the chaotic time series prediction performance (Webb 2008). A recent review summarizes variations on these architectures (Tanaka et al. 2018).

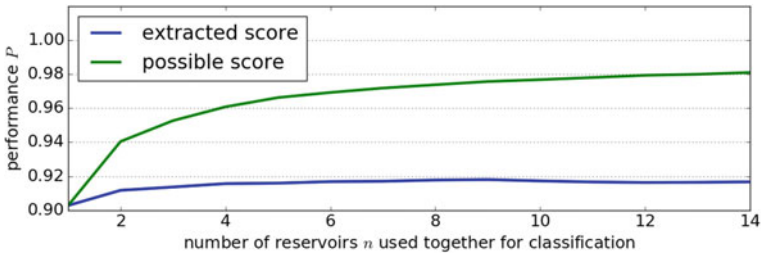


Fig. 21 Classification performance for parallel RCs that each classify all written digits (blue line) and the best combination of reservoirs where each only classifies a subset of the digits (green line)

Specific to the MNIST task, Jalalvand et al. (2015) showed improved performance of an RC using multiple RCs in a parallel, serial, or parallel-serial geometries. Their work is based on a software simulation with each RC having ∼10,000 reservoir nodes, but only feeding in a single column of pixel data from the original MNIST 28×28-pixel images. It is not apparent whether a similar approach will work here where we consider using ABNs with link time delays, whereas (Jalalvand et al. 2015) considers nodes with a hyperbolic tangent nonlinearity and no link delays. There is some hope for an improvement using a parallel approach because specific reservoir realizations perform somewhat better on some digits than others. We use the data collected from the different reservoir realizations in a parallel configuration with ℓ RCs where we choose the ℓ best reservoirs. The results are combined using a majority vote for classification. As seen in Fig. 21, there is a modest increase in the performance, peaking at $P = 91.8\%$ with $\ell \sim 8$.

To obtain a sense of a possible optimal FPGA-based reservoir with 64 nodes, we search through our database of results to find the ℓ reservoirs that perform the best for a single digit. The performance of this hypothetical machine is shown by the green line. Better performance can likely only be obtained by increasing the various resources as discussed in the next section.

5 Discussion and Conclusions

In this chapter, we demonstrate that an FPGA-based RC can perform the written-digit classification task. The performance over a wide range of reservoir parameters exceeds that of a linear classifier, although the error rate is still relatively high in comparison to the state-of-the-art deep learning feed-forward artificial neural networks with fully trained weights. One way to improve the performance of our RC is to increase the number of connections of the input data to the reservoir and to increase the number of reservoir nodes; in other related studies, we find that the performance scales approximately as the square root of the number of input connections and reservoir nodes. In the future, we will also investigate full on-chip training and

classification (Yi et al. 2016), which should lead to extremely fast operation of an RC. The FPGA platform is an especially appealing RC physical substrate because it is commercially available, operates at low power, runs at high speed, easily integrated with traditional computer infrastructure, and can potentially lead to beyond-Turing computing because we run the reservoir in the autonomous mode.

Acknowledgements S.A. and E.S. gratefully acknowledge the financial support of the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—Projektnummer 163436311- SFB 910. N.D.H., O.D., and D.J.G. gratefully acknowledge financial support from the U.S. Army Research Office through Grant No. W911NF-12-1-0099. O.D. also gratefully acknowledges financial support from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 713694. We gratefully acknowledge the IT support of Jimmy Dorff of the Duke University Department of Physics for extensive help in developing the server to automate our experiments.

Appendix: FPGAs and project workflow

Brief introduction to FPGAs

An extensive discussion on using FPGAs for experiments on autonomous Boolean networks, including hardware description language metacode, can be found in Rosin (2015), D'Huys et al. (2016), Lohmann et al. (2017), and Canaday et al. (2018). Briefly, FPGAs contain more than 10^5 programmable logic elements (LE) that can be linked via programmable connections. The logic elements are based on CMOS technology and are arranged in a grid system. Each of these elements consists of a look-up table (LUT) with four inputs for the Altera Cyclone IV chip family used here (EP4CE115F29C7N, Terasic DE2-115 demonstration board), a flip-flop and a multiplexer. Figure 22 illustrates the layout of a logic element.

The multiplexer is used to switch between clocked and unclocked (autonomous) mode and the flip-flop provides the clocked output. The look-up table can be used to define an operation for K inputs, so having a length of 2^K bits and thus can be

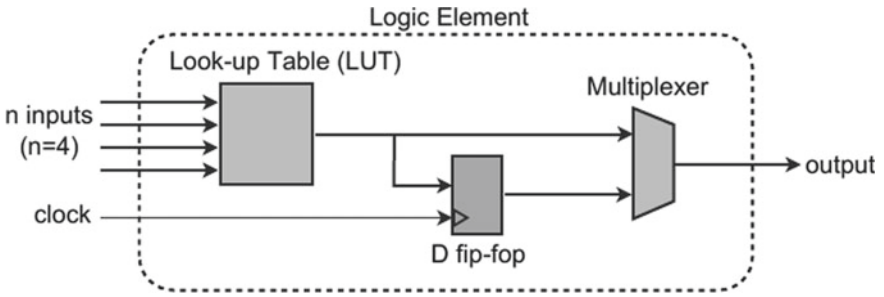


Fig. 22 Structure of a single logic element on the FPGA. The figure is adapted from Rosin (2015)

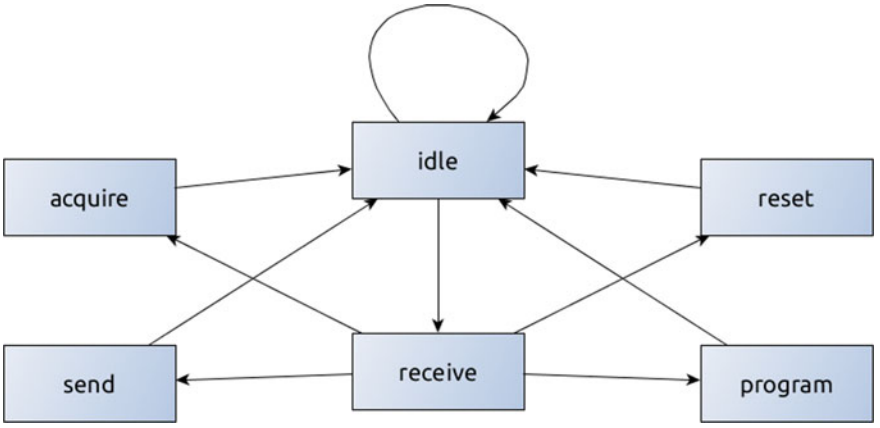


Fig. 23 Illustration of the transition between the six states of the FSM

used to realize up to 2^{2^K} different Boolean functions. The Altera Cyclone IV logic elements used in this FPGA have a maximum input number of $K = 4$ so a total number of 65,536 different operations can be performed by a single LE. This makes it easy to create an arbitrary setup on the device. Their relatively low costs and their operation time on the nanosecond time scale are ideally suited for reservoir computing applications.

The network is defined using the hardware description language *Verilog*, which instructs the FPGA how to configure itself, including the function of the nodes and the network links. This code is compiled using the proprietary software *Quartus II*, version 14.0 (Quartus 2021). The resulting bitstream is loaded onto the FPGA, which causes the configuration of the RC. See the references above for details. Unique to this project is the way in which the input and output state data are handled and interfaced with the autonomous network forming the reservoir. We use a finite-state machine (FSM) synthesized from clocked logic and communicates with a personal computer (PC) via a USB controller chip (FTDI part #FT232H), which is connected directly to the FPGA pins via the GPIO connector on the demonstration board.

The FSM has the distinct states: `idle`, `receive`, `acquire`, `send`, `program`, and `reset` whose possible transitions are depicted in Fig. 23. It starts in the `idle` state and remains without any action as long as it does not receive any input over the USB connection via the PC. After a signal is detected on the USB, the FSM changes to the `receive` state. Next, the first available byte read from the USB determines the next state of the FSM. The `program` state reads a number of bytes from the USB connection and stores it onto the RAM onboard the chip. This number is set in the *Verilog* code defining the reservoir (see below) and can only be changed by recompiling the *Verilog* code. The data transferred to the FPGA in this manner sets the initial state for a reservoir.

When the state changes to `acquire`, all nodes in the reservoir are set to the value `zero` and the data from the RAM that was transferred during the `program`

process is fed into the reservoir. Because the FSM operates with clocked logic, the initial state is inserted over one clock cycle with the length of 5 ns (corresponding to a data rate of 200 MHz). After this data insertion, the reservoir runs freely and autonomously. The only restrictions for the dynamics arise from the finite response time of electric elements within the FPGA. On every clock cycle (5 ns period), a snapshot of the reservoir is taken by saving the current value of each node to the RAM. In this fashion, a series of 200 samples is captured and stored. Note that the dynamics in between clock cycles can't be measured using this method.

After all data is collected, the FSM is set to the `send` state, which will send all the data stored in the RAM to the USB connection and is available to a Python program running on the PC. The same reservoir can be used to collect data for the same input word by sending several `acquire` commands or a new input word can be used just by changing the initial-state RAM contents without having to recompile the *Verilog* code and transferring the resulting bitstream to the FPGA.

The *Verilog* code for the FSM has not been presented in our previous publications so we give the module below for completeness.

```
1   /*
2   Master FSM monitors bytes sent from the PC to
3   look for command signals
4   Valid command signals are:
5     Begin aquiring data
6     Send data from RAM to PC
7     Program input words
8     Reset
9   When a valid command is recognized, the FSM flips
10  a signal bit that other modules are monitoring
11  */
12
13  module master_fsm(CLOCK_50, KEY, rcvd_sig,
      byte_from_pc, acquire_signal, send_signal,
      prog_word_signal, reset);
15
16  // Parameters
17  parameter IDLE = 0, RCVD = 1, ACQD_I = 2,
     ACQD_II = 3, SEND_I = 4, SEND_II = 5, PROG_I = 6,
     PROG_II = 7, RST_I = 8, RST_II = 9;
18
19  // Internal elements
20  input CLOCK_50;
21  input KEY;
22  input rcvd_sig;
23  input [7:0] byte_from_pc;
24
```

```
output acquire_signal;
output send_signal;
output prog_word_signal;
output reset;

reg [3:0] state;
reg [5:0] count;

wire hardware_reset;

reg acq_sig, send_sig, prog_sig, rst_sig;
assign acquire_signal = acq_sig;
assign send_signal = send_sig;
assign prog_word_signal = prog_sig;
assign hardware_reset = ~KEY;
assign reset = hardware_reset | rst_sig;

// The finite-state machine
always @(*)
   begin
      case (state)
         IDLE:
            begin
               acq_sig <= 1'b0;
               send_sig <= 1'b0;
               prog_sig <= 1'b0;
               rst_sig <= 1'b0;
            end
         RCVD:
            begin
               acq_sig <= 1'b0;
               send_sig <= 1'b0;
               prog_sig <= 1'b0;
               rst_sig <= 1'b0;
            end
         ACQD_I:
            begin
               acq_sig <= 1'b1;
               send_sig <= 1'b0;
               prog_sig <= 1'b0;
               rst_sig <= 1'b0;
            end
         ACQD_II:
            begin
               acq_sig <= 1'b1;
```

```
70                send_sig <= 1'b0;
71                prog_sig <= 1'b0;
72                rst_sig <= 1'b0;
73             end
74          SEND_I:
75             begin
76                acq_sig <= 1'b0;
77                send_sig <= 1'b1;
78                prog_sig <= 1'b0;
79                st_sig <= 1'b0;
80             end
81          SEND_II:
82             begin
83                acq_sig <= 1'b0;
84                send_sig <= 1'b1;
85                prog_sig <= 1'b0;
86                rst_sig <= 1'b0;
87             end
88          PROG_I:
89             begin
90                acq_sig <= 1'b0;
91                send_sig <= 1'b0;
92                prog_sig <= 1'b1;
93                rst_sig <= 1'b0;
94             end
95          PROG_II:
96             begin
97                acq_sig <= 1'b0;
98                send_sig <= 1'b0;
99                prog_sig <= 1'b1;
100               rst_sig <= 1'b0;
101            end
102         RST_I:
103            begin
104               acq_sig <= 1'b0;
105               send_sig <= 1'b0;
106               prog_sig <= 1'b0;
107               rst_sig <= 1'b1;
108            end
109         RST_II:
110            begin
111               acq_sig <= 1'b0;
112               send_sig <= 1'b0;
113               prog_sig <= 1'b0;
114               rst_sig <= 1'b1;
```

```
115             end
116          default:
117             begin
118                acq_sig <= 1'b0;
119                send_sig <= 1'b0;
120                prog_sig <= 1'b0;
121                rst_sig <= 1'b0;
122             end
123       endcase
124    end
125
126 always @(posedge CLOCK_50 or posedge
       hardware_reset)
127    begin
128       if (hardware_reset)
129          state <= IDLE;
130       else
131          case (state)
132             IDLE:
133                begin
134                   if (rcvd_sig)
135                      state <= RCVD;
136                   else
137                      state <= IDLE;
138                end
139             RCVD:
140                begin
141                   case (byte_from_pc[7:0])
142                      8'b00000001: state <= ACQD_I;
143                      8'b00000010: state <= SEND_I;
144                      8'b00001000: state <= PROG_I;
145                      8'b00010000: state <= RST_I;
146                      default:
147                         state <= IDLE;
148                   endcase
149                end
150             ACQD_I:
151                state <= ACQD_II;
152             ACQD_II:
153                state <= IDLE;
154             SEND_I:
155                state <= SEND_II;
156             SEND_II:
157                state <= IDLE;
158             PROG_I:
```

```
159                 state <= PROG_II;
160             PROG_II:
161               state <= IDLE;
162             RST_I:
163                 state <= RST_II;
164             RST_II:
165                 state <= IDLE;
166             default:
167                 state <= IDLE;
168         endcase
169     end
170
171 endmodule
```

Experimental pipeline

Creating networks, running the experiments, and analyzing the results presented in this chapter was simplified by automating the process with the following feature specifications:

- The system runs in an automated fashion with as few manual steps as possible
- The system is separated into the distinct steps: creating the network; compiling the network; running the experiment; and analyzing the data
- Scalability: Each individual step has any number of computers that can be added or removed to or from the system at any time to run tasks in parallel
- Extensibility: New methods for running experiments or analyzing data can be added at any time and all connected machines will automatically receive all updates
- Supervision: The system is accompanied by a website that always shows the current state of all ongoing actions

The code for this project is extensive; we provide it on a repository on an as-is basis for those interested in exploring or extending the system (GitHub 2021).

The entire system is implemented in Python and runs on a centralized server set up for us by the Duke University Physics Department IT staff. The core of the system is based on a MySQL database that provides tables for *reservoir*, *experiment* and *analysis*. All tables contain the data unique to each step of the workflow, such as the number of reservoir nodes, number of delay elements, network connectivity, number of written-image input words, and experimental repetitions. The pipeline is controlled by two additional fields: *status* and *tag*, which are assigned to all items. The status is an integer number that indicates open, running, successful, error, or canceled. All sub-systems have a unique identifier that creates a *tag* on each item, which allows us to determine which machine creates a certain item. The run-time order is usually random and depends on the internal ordering of the database. If some items are preferred over others, we assign a *priority* tag to force a schedule order.

It is challenging to interact with the *Quartus II* system to automate creating different reservoirs in an automated manner. Therefore, this step was completed by a user who creates a new python script to add new reservoirs. This process was

facilitated as much as possible by providing many functions to create random nodes and links and assign a bias to node functions. Finally, the *Verilog* code defining the reservoir is created with appropriate metadata to identify the reservoir.

After uploading a compiled bitstrea file, a new entry is inserted into the database with a unique *id* and all attributes extracted from the reservoir file. The new item is created as *status* open and has no tag assigned. For this step, a compile daemon checks for new reservoirs, specifically looking for *status* open. If a new item is found, it is updated to *status* running with the *tag* of the machine that claimed the reservoir, thus preventing any other machine from taking the same item. The daemon then downloads the reservoir file from the server and uses the *Quartus II* software to compile it. If the process ends successfully, the new file is then uploaded back to the server and the *status* becomes successful. Otherwise, the *status* is set to error and a file containing the error message is uploaded instead.

To run experiments, an input creation function must exist. This function defines the initial values of the network nodes. If a new function is required by a user, it is developed and added to the system. Developing the project is managed by the versioning software *git*, which has its repository on the Duke University centralized data storage system.

Creating a new experiment is undertaken by inserting the experimental definition for an existing reservoir with input method and all required parameters into the database with *status* open. An experiment daemon similar to reservoir daemon checks for new experiments that have their *status* open, but additionally requires the used reservoir to have *status* successful. It then assigns *status* running and sets the tag identifier. If a new experiment is found, the daemon checks if an update has been made to the code via the *git* repository and updates if necessary. Next, the input creation function is called with all parameters and returns the defined number of different input words. These initialize the network and each input is run with the set repetition number. The entire data is stored in a single binary file that is uploaded if the collection succeeds. An error message is uploaded in case of an error.

To analyze the data, we use the Open Science Grid (OSG) (OSG 2021), which is a giant computer cluster that includes machines from many universities all over the United States of America. This platform allows us to start hundreds of jobs at the same time and distributes the work to all connected machines, thus providing massive computational power. However, this resource is not always reliable and sometimes fails.

To account for this possibility, we use an analysis daemon that runs on local machines as a fall back. All analysis code is developed in advance and synchronized with *git* repository. The local and OSG daemons check for open analyses that have successful experiments as their predecessor, and verifies that the latest version of the code is in use. For the OSG, the data is uploaded and a configuration file sent with all analysis parameters. When the analysis completes successfully, the results files are uploaded back to the Duke University data storage system and all data on the OSG belonging to this analysis is deleted. In case of an error, the data remains on the OSG for further investigation. In any case, the *status* is set to the result.

References

M.L. Alomar, M.C. Soriano, M. Escalona-Morán, V. Canals, I. Fischer, C.R. Mirasso, J.L. Rosselló, Digital implementation of a single dynamical node reservoir computer. IEEE Trans. Circuits Syst. II: Exp. Briefs **62**, 977 (2015)

P. Antonik, *Application of FPGA to Real-Time Machine Learning* (Springer, Cham, 2018)

P. Antonik, A. Smerieri, F. Duport, M. Haelterman, S. Massar, FPGA implementation of reservoir computing with online learning, in *24th Belgian-Dutch Conference on Machine Learning (Benelearn)*, Benelearn, vol. 24, 19 June 2015, Delft, Netherlands (2015)

S. Apostel, Dynamics of driven complex autonomous Boolean networks with application to reservoir computing. M.S. thesis, Technische Universität Berlin (2017). Unpublished

L. Appeltant, M.C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C.R. Mirasso, I. Fischer, Information processing using a single dynamical node as complex system. Nat. Commun. **2**, 468 (2011)

L. Appeltant, G. Van der Sande, J. Danckaert, I. Fischer, Constructing optimized binary masks for reservoir computing with delay systems. Sci. Rep. **4**, 3629 (2014)

D. Canaday, A. Griffith, D.J. Gauthier, Rapid time series prediction with a hardware-based reservoir computer. Chaos **28** (2018)

F. Denis-Le Coarer, M. Sciamanna, A. Katumba, M. Freiberger, J. Dambre, P. Bienstman, D. Rontani. All-optical reservoir computing on a photonic chip using silicon-based ring resonators. IEEE J. Sel. Top. Quantum Electron. **24**, 1–8 (2018)

B. Derrida, Y. Pomeau, Random networks of automata: a simple annealed approximation. Europhys. Lett. **1**, 45 (1986)

O. D'Huys, J. Lohmann, N.D. Haynes, D.J. Gauthier, Super-transient scaling in time-delay autonomous Boolean network motifs. Chaos **26**, 094810 (2016)

R. Edwards, L. Glass, A calculus for relating the dynamics and structure of complex biological networks, in *Adventures in Chemical Physics: A Special Volume of Advances in Chemical Physics*, ed. by R.S. Berry, J. Jortner (John Wiley & Sons, Inc., Hoboken, 2006), pp. 151–178

R. Edwards, P. van den Driessche, L. Wang, Periodicity in piecewise-linear switching networks with delay. J. Math. Biol. **55**, 271 (2007)

H. Flyvbjerg, An order parameter for networks of automata. J. Phys. A. **21**, L955 (1988)

D.J. Gauthier, Reservoir computing: harnessing a universal dynamical system. SIAM News **51**(2), 12 (2018)

M. Ghil, A. Mullhaupt, Boolean delay equations. II. Periodic and aperiodic solutions. J. Stat. Phys. **41**, 125 (1985)

M. Ghil, I. Zaliapin, B. Coluzzi, Boolean delay equations: a simple way of looking at complex systems. Phys. D **237**, 2967 (2008)

GitHub (2021), https://github.com/nickdavidhaynes/boolean-reservoir-computer

N.D. Haynes, M.C. Soriano, D.P. Rosin, I. Fischer, D.J. Gauthier, Reservoir computing with a single time-delay autonomous Boolean node. Phys. Rev. E **91**, 020801 (2015)

H. Jaeger, Discovering multiscale dynamical features with hierarchical echo state networks. Technical report 10, School of Engineering and Science, Jacobs University (2007). Unpublished

H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**, 78 (2004)

A. Jalalvand, K. Demuynck, W. De Neve, R. Van de Walle, J.-P. Martens. design of reservoir computing systems for noise-robust speech and handwriting recognition, in *Conference on Graphics, Patterns and Images (SIBGRAPI)*, vol. 28. Salvador. Porto Alegre: Sociedade Brasileira de Computação (2015). On-line. IBI: 8JMKD3MGPBW34M/3JUJ5DP, http://urlib.net/rep/8JMKD3MGPBW34M/3JUJ5DP

A. Katumba, J. Heyvaert, B. Schneider, S. Uvin, J. Dambre, P. Bienstman, Low-loss photonic reservoir computing with multimode photonic integrated circuits. Sci. Rep. **8**, 2653 (2018)

L. Larger, A. Baylóon-Fuentes, R. Martinenghi, V.S. Udaltsov, Y.K. Chembo, M. Jacquot, High-speed photonic reservoir computing using a time-delay-based architecture: million words per second classification. Phys. Rev. X **7**, 011015 (2017)
Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. Proc. IEEE **86**, 2278 (1998)
J. Lohmann, O. D'Huys, N.D. Haynes, E. Schöll, D.J. Gauthier, Transient dynamics and their control in time-delay autonomous Boolean ring networks. Phys. Rev. E **95**, 022211 (2017)
B. Luque, R.V. Solé, Lyapunov exponents in random Boolean networks. Phys. A **284**, 33 (2000)
J.P. Mason, P.S. Linsay, J.J. Collins, L. Glass, Evolving complex dynamics in electronic models of genetic networks. Chaos **14**, 707 (2004)
C. Mesaritakis, A. Bogris, A. Kapsalis, D. Syvridis, High-speed all-optical pattern recognition of dispersive Fourier images through a photonic reservoir computing subsystem. Opt. Lett. **40**, 3416–3419 (2015)
MNIST (2021), http://yann.lecun.com/exdb/mnist/
OSG (2021), https://opensciencegrid.org/
Y. Paquot, F. Dupart, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, S. Massar, Optoelectronics reservoir computing. Sci. Rep. **2**, 287 (2012)
B. Penkovsky, L. Larger, D. Brunner, Efficient design of hardware-enabled reservoir computing in FPGAs. J. Appl. Phys. **124**, 162101 (2018)
Quartus (2021), https://www.intel.com/content/www/us/en/programmable/downloads/download-center.html
A. Röhm, K. Lüdge, Multiplexed networks: reservoir computing with virtual and real nodes. J. Phys. Commun. **2**, 085007 (2018)
D.P. Rosin, *Dynamics of Complex Autonomous Boolean Networks* (Springer, Heidelberg, 2015)
E.S. Skibinsky-Gitlin, M.L. Alomar, C.F. Frasser, V. Canals, E. Isern, M. Roca, J.L. Rosselló, Cyclic reservoir computing with fpga devices for efficient channel equalization, in *Artificial Intelligence and Soft Computing. ICAISC 2018*, ed. by L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, J. Zurada. Lecture Notes in Computer Science, vol. 10841 (Springer, Cham, 2018)
G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, A. Hirose, Recent advances in physical reservoir computing: a review (2018), arXiv:1808.04962v2
A. Uchida, R. McAllister, R. Roy, Consistency of nonlinear system response to complex drive signals. Phys. Rev. Lett. **93**, 244102 (2004)
K. Vandoorne, P. Mechet, T. Van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, P. Bienstman, Experimental demonstration of reservoir computing on a silicon photonics chip. Nat. Commun. **5**, 3541 (2014)
Q. Vinckier, F. Duport, A. Smerieri, K. Vandoorne, P. Bienstman, M. Haelterman, S. Massar, High-performance photonic reservoir computer based on a coherently driven passive cavity. Optica **2**, 438–446 (2015)
R.Y. Webb, Multi-layer corrective cascade architecture for one-line predictive echo state networks. Appl. Artif. Intell. **22**, 811 (2008)
Y. Yi, Y. Liao, B. Wang, X. Fu, F. Shen, H. Hou, L. Liu, FPGA based spike-time dependent encoder and reservoir design in neuromorphic computing processors. Microprocess. Microsyst. **46B**, 175 (2016)
I.B. Yildiz, H. Jaeger, S.J. Kiebe, Re-visiting the echo state property. Neural Netw. **35**, 1 (2012)
R. Zhang, H.L.D. de S. Cavalcante, Z. Gao, D.J. Gauthier, J.E.S. Socolar, M.M. Adams, D.P. Lathrop, Boolean Chaos. Phys. Rev. E. **80**, 045202(R) (2009)
H. Zhang, Z. Feng, B. Li, Y. Wang, K. Cui, F. Lin, W. Dou, Y. Huang, Integrated photonic reservoir computing based on hierarchical time-multiplexing structure. Opt. Express. **22**, 31356–31370 (2014)

Programmable Fading Memory in Atomic Switch Systems for Error Checking Applications

Renato Aguilera, Henry O. Sillin, Adam Z. Stieg, and James K. Gimzewski

Abstract Disruptive technology in computational devices is required as the universal computing machines approach quantum mechanical limits. Integration of state-of-the-art memristive devices provides optimal scaling of current technologies beyond this limit through the adoption of neuromorphic models. Universal computing machines pioneered by Alan Turing are strictly based on top-down intelligent design. Neuromorphic models instead engage in bottom-up programmability by emulating mammalian brain design and characteristics. Here we show the design, characterization, and implementation of a massively parallel memristor neuromorphic network based on metal chalcogenide atomic switch network (ASN) systems with key characteristics such as short- and long-term potentiation, power-law dynamics, and scale-free topology.

1 Introduction

Fundamental work by Carver Mead and colleagues (Mead 1990) in the development of the concept neuromorphic technology enabled a disruptive paradigm shift in computing technologies. Unlike other conceptions of machine learning, neuromorphic computing attempted to completely emulate neuron functionality within a physically realizable computing hardware. In doing so, the power-efficiency and complexity of neurons can be harnessed without bottlenecking data as in CMOS technology (Backus 1978). Additionally, neuron clusters in the brain can recognize patterns and are capable of performing unconventional computing similar to Alan

R. Aguilera · H. O. Sillin · J. K. Gimzewski (✉)
Department of Chemistry and Biochemistry, University of California, Los Angeles 607 Charles E. Young Drive East, Los Angeles, CA 90095, USA
e-mail: gimzewski@cnsi.ucla.edu

A. Z. Stieg · J. K. Gimzewski
California NanoSystems Institute, 570 Westwood Plaza, Los Angeles, CA 90095, USA

National Institute for Materials Science (NIMS), WPI Center for Materials Nanoarchitectonics (MANA), 1-1 Namiki, Tsukuba, Ibaraki 305-0044, Japan

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_12

Turing's B-machine (Turing 1950). The evolutionary optimization of the brain in both structure and functionality provides an exciting new zeitgeist in a fast stalling technology (Waldrop 2016).

Current works using learning dedicated computer hardware provide possible throttling pass the information bottleneck (Husband et al. 2003; Tour et al. 2003). Additionally, developments of beyond-CMOS devices such as magnetic tunneling junctions (Furuta et al. 2018), photoelectronic (Hermans et al. 2015), and memristors (Ascoli et al. 2017; Du et al. 2017; Strukov et al. 2008) explored computing architectures outside of typical transistor-based models. In 2000, the International Center for Materials Nanoarchitectonics (MANA) commenced investigation of viable neuromorphic materials utilizing nanowire mechanisms and constraints for material design. Specifically, Professor Masakazu Aono developed the atomic switch as a nano-equivalent neuron operating under quantum mechanical limits at GHz speeds (Tsuchiya et al. 2015; Tsuruoka et al. 2017). Aono's work on atomic switches developed the underlying principles for integration and development of nanowires for neuromorphic computing elucidating nanowire plasticity and memory capabilities. Single transistor-like atomic switches were introduced into memory storage devices by Nippon Electric Company (NEC 2009) using non-dynamic Cu-TiO_2/TaSiO atomic switches for 32 nm CMOS technology. However, the plasticity present in neurological functions is inherently non-static and dynamic (Büsing et al. 2010; Lukoševičius and Jaeger 2009). Further development of scalable neuromorphic atomic switch devices required a holistic nanoarchitectonic design incorporating dynamic and nonlinear network behaviors.

An emerging mathematical model developed by Leon Chua sought to integrate CMOS technology with nonlinear and chaotic systems (Chua 1980, 1988). In 1971, Chua theorized that, in addition to the 3 fundamental elements in the lumped element circuit model, there was a 4th element he called the *memristor*. Complementing the relations provided by the resistor, capacitor, and inductor, the memristor was able to relate the magnetic flux with electric charge. In order for circuit theory to utilize this 4th element, the model required adaptation of a purely nonlinear circuit theory more akin to Turing's B-machine. Particularly, the model emphasized harnessing emerging behaviors due to coupled circuits similar to the ideals of neuromorphic computing. Here, we present the fabrication of a physical random neural network via growth of memristive atomic switch networks, harnessing phenomenological fractal growth to directly imitate neural evolution for neuromorphic computing.

Unlike conventional neuromorphic platforms which require meticulous design, atomic switch networks (ASN) produced by interconnected nanowires introduce a unique methodology of controlled evolution. Meticulous design of a system required a complete model of understanding such as the CMOS computer architecture modeled by the universal Turing machine. However, it was more practical to develop a scheme of top-down adaptation than by bottom-up selective modification toward the desired function due to hardware limitations. A combined effort of theoretical predictions and experimental verification is presented here to design a methodology that was physically practical in implementing a computation referred to as *reservoir computing*. A physically realizable recurrent network comprised of

gapless-type atomic switches with Ag_2S as the active material previously demonstrated device tenability as functionally compatible neurons (Demis et al. 2015, 2016; Sillin et al. 2013).

An analysis of atomic switch networks at the interface of theory and experiment was accomplished by implementing theoretical paradigms of computation from the perspective of experimental feasibility. Considerations of physical practicality and CMOS compatibility were given priority over an ideal model with microscopic details. Reservoir computing was implemented on the proposed device to accomplish a series of error checking parity tasks and activation control to assess its computational capability.

We proposed to answer the following questions:

1. What is the relationship between the dynamical properties of a random system and its computational capability as a reservoir?
2. How do these dynamical properties emerge in macroscopic tools available to an experimentalist?

1.1 The Atomic Switch

Atomic switches were a class of devices that enabled the use of quantum tunneling for signal transduction. The first experiment to measure the transition from an electron quantum tunneling to single point contact regime was reported in 1987 using a scanning tunneling microscope (STM) in ultra-high-vacuum (UHV) on a silver surface (Möller 1987). Current-distance characteristics showed an abrupt increase in conductance, $G \sim \frac{e^2}{2h} \approx \frac{1}{13}$ kΩ, at sufficiently small tip-surface gaps, establishing the quantized unit of conductance. Subsequent theoretical analysis verified that at small gap distance the effective tunnel barrier collapses prior to point contact via ballistic electron injection (Lang 1986). Later work demonstrated further jumps of $G \sim n\frac{2e^2}{h}$, where $n = 1, 2, 3 \ldots$, in the conductance occur as the contact area is increased. Such observations were not limited to STM experiments; even two macroscopic wires brought in contact also displayed this effect, albeit in a less controlled manner. Houten *et al*. provide an excellent review of quantized conduction, which also introduces Landauer's concept of transmission $G = \frac{2e^2}{h}\sum_n t_n$, where the term t is the transmission (van Houten et al. 1996).

In 2002, experiments by Terabe et al. found that Ag atoms could be transported through a STM tip made of silver coated with silver sulfide and deposited on a surface in a controlled manner (Terabe et al. 2002). The characteristics of this process also occurred via quantized conduction, however, the mechanism involved ion migration under the influence of an electric field, a process called "electroionics", meaning that in addition to electron motion, ion motion also occurs simultaneously. Normally, ionic diffusion processes on the macro-scale are considered to be slow, but when they are induced on the nanometer scale, they are actually quite fast and can occur on a (sub-) nanosecond scale depending on the geometry and dimensions of the junction.

In 2005, using junctions fabricated using conventional microelectronics, Terabe et al. demonstrated atomic switching in silver sulfide junctions with discrete and reversible quantized jumps from n = 1 to 10. This was the birth of the "atomic switch". Since that date, quantized conduction has been observed by a number of researchers in a wide range of materials including sulfide junctions of copper, tungsten sub-oxides as well as various metal-doped polymers.

Aside from the fundamental science of their quantization, the interesting electronic features of atomic switches were hysteresis, on/off conduction ratio, switching speed and volatility characteristics as well as CMOS compatibility because of their potential in digital electronic memory applications. Indeed, NEC recently has incorporated atomic switch technology into field programmable gate arrays (FPGAs) (Aramaki et al. 1991; NEC 2009) where a reduction in device footprint, speed, and energy consumption was achieved by replacing certain memory tasks, normally using transistors, into the circuitry.

Additional atomic switch functionality was reported in 2011 when studying switching near-threshold conditions (Ohno et al. 2011; Hasegawa et al. 2011). It was found that atomic switches have an on-off memorization property of past switching events. For instance if switching is performed infrequently, the switches remain in the on-state only briefly whereas if frequent switching events are made in rapid succession then the on-state persists for a longer time. A series of careful experiments were able to relate these physical observations to a psychological model of learning called the Akinson-Schriffin multi-store model (Atkinson and Shiffrin 1968). The essence of the model involves sensory memory (SM), short-term memory (STM), and long-term memory (LTM). New information arrives in the brain as sensory memory and that information was passed to short-term memory. In the absence of similar information it was forgotten. However, if the process was repeated many times, the information was moved into long-term memory. In terms of bio-inspiration, the operational characteristics of the atomic switch under threshold switching were also related to characteristics of biological synapses. The atomic switch therefore has also been called a synthetic synapse where memory was represented by conduction state.

The next step in creating a "brain inspired" device was the fabrication of networks of synthetic synapses (Atomic Switches). Taking the neocortex as a biologically inspiration, self-assembly was used to incorporate atomic switches into a dense dendritic tangle of silver nanowires resulting in a density of ~10^8 connections/cm^2 (Avizienis et al. 2013). In response to electrical inputs, which inject energy into the network, these networks exhibited self-organization and critical power-law dynamics and spatio-temporal nonlinear outputs at multiple electrodes.

1.2 Neuromorphic Atomic Switch Networks

The clear desire for neuromorphic architectures has led to further investigations and developments of different synthetic synapse models. Establishing specific connections between patterns of electrical activity and brain function was a difficult task

that requires studying general features of neuronal structure in order to determine the essential properties required to construct a device capable of learning in a physical sense. These features are believed to include at least synaptic plasticity, allowing physical reconfiguration of the network to enable functional differentiation and the development of hierarchical structures in which all possess correlated memory distributed throughout the dynamically coupled synapses. Therefore, learning and computational capacity in the brain are connected to dynamic activity and functional connectivity. Specifically, a near-critical or "edge-of-chaos" operational (Langton 1990) regime has been associated with the fast, correlated response to stimulation necessary for computation and learning. Developing computational machinery whose operation results from intrinsic critical dynamics was a complex task; with Atomic Switch Network (ASN) devices demonstrated such functionality (Fig. 1).

Utilizing the theoretical concepts presented in the previous section, we designed a neuromorphic device by incorporating atomic switches in a highly recursive interconnected network. The work of Aono (Terabe et al. 2002; Ohno et al. 2011; Hasegawa et al. 2011) established the fundamental principles and design of atomic switches. Observation of plasticity and information retention in atomic switches enabled us to successfully implement them in neuromorphic hardware for reservoir computing (Demis et al. 2016; Sillin et al. 2013). A number of materials were available and various functionalities may be tuned depending on the active material. Here, metal-chalcogenides were chosen due to their ready integration into CMOS technology and capability for spontaneous fractal growth.

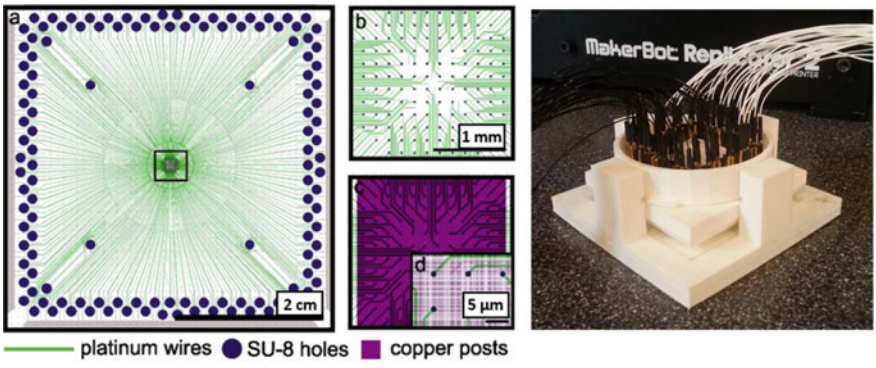


Fig. 1 Network diagram and analogue interface. A circuit schematic is shown in **a** showing the platinum circuit (green wires) and readout PC interface (blue circles). The silver network was placed in the central region (boxed) and a closer inspection of the region wiring is shown in **b**. Pre-patterned posts were lithographically placed within the boxed region in **c** ready for network growth. The device was interfaced to a National Instruments PXI—e using a custom device housing

2 Theoretical Constraints and Consideration

The concept of neuromorphic hardware as conceived by Mead (1990) intended to emulate the problem-solving capability of biology, which has been evolutionary optimized by nature. Observations of DNA folding and editing (Romero et al. 2017) demonstrated the capabilities of evolutionary algorithms to enable DNA to execute complex protein interaction and regulation. Natural selection has inherently optimized these systems in their task-specific function, thereby minimizing energy, maximizing information transfer, and encoding fault-tolerant and adaptive behaviors (Kim and Han 2002). Neuromorphic hardware attempted to adapt this architectural design within the context of circuit theory and analysis. A premier model for computational design was the human–brain, where complex computation such as speech, multisensory control, and chaotic predictions was commonly executed while operating under relatively simple rules.

Fundamentally, the brain utilized synaptically interconnected neurons to transfer and process information. Each neuron operated under the Hebbian fire-diffuse-fire principal (Hebb 1950; Timofeeva and Coombes 2003) which activated neurons with a sigmoidal function profile similar to transistors with voltage replacing ions in the latter case. Unlike contemporary digital transistors, each neuron was heavily coupled to other elements behaving effectively as a history-dependent nonlinear device. Circuit network theory has analogous examples of coupled inductive circuits communicating across devices, but such circuits were typically designed to eliminate coupled cross-talk and the overall circuit was linearized in its functionality. However, simplification of these interactions invariably destroys emergent behaviors observed in complex systems (Chen and Wang 2003; Goudarzi et al. 2012), which was capable of accomplishing complex computation. Nonlinear circuit design and analytical models by Chua (Ascoli et al. 2017; Chua and Wu 1995) attempted to utilize these complex interactions but have limited integration within information technology. Instead, machine learning algorithms were implemented in software which mimics the design and learning rules of biological systems. Here, a combination of machine learning architecture and nonlinear circuit design is briefly presented and discussion was restricted within a feedforward network for brevity; however, a formal and comprehensive discussion has been previously published (Lukoševičius and Jaeger 2009; Verstraeten et al. 2009).

The neural network machine learning paradigm traditionally attempted to achieve learning by modification of network topology and connectivity via adjustments to neuronal coupling strength. The general architecture of a neural network was designed similar to the human–brain—a collection of nodes or neurons interconnected with synapses to other neurons in a hierarchal layered structure (Graves et al. 2013; Krizhevsky et al. 2012; Abraham 2005; Schrauwen 2007; Hassoun 1995; Hopfield 1988). Neuronal nodes were typically designed to integrate incoming signals and transform them using a sigmoidal transfer function. The integration was the weighted sum of all signals received by the neuron from a predetermined set of input neurons from the previous layer:

$$\vec{I}^{0}(t) = \hat{N}_0 \vec{I}^{D}(t) \equiv \vec{f}^{0}\left(\vec{I}^{D}(t)\right), \tag{1}$$

where $\vec{I}^{0}(t)$ was a vector with elements of the transduced signal of each neuron in layer 0, $\hat{N}_0$ was a matrix representing the transformation of the driving signal, $\vec{I}^{D}(t)$ the driving signal, and $\vec{f}^{0}$ was a vector whose elements are the transfer functions of each neuron. The neuron activated according to its designed transfer function, then propagated the signal to its pre-designated output neurons in the next layer, i.e., layer 1:

$$\vec{I}^{1}(t) = \hat{N}_1 \vec{I}^{0}(t) \equiv \vec{f}^{1}\left(\hat{w}_1 \cdot \vec{I}^{D}(t)\right) \tag{2}$$

Here, $\vec{w}_i^{1}$ was the coupling strength between neuron i in layer 1 and all other neurons. This process was repeated from neuron to neuron in a hierarchal structure composed of layers or networks of neurons until it arrives at an output neuron layer, represented by $\vec{I}^{T}(t)$, where the user observes and process the final signal $I^{F}(t)$:

$$\vec{I}^{T}(t) = \prod_{k}^{\forall} \hat{N}_k \vec{I}^{D}(t) \equiv \hat{O}\vec{I}^{D}(t) \tag{3}$$

$$I^{F}(t) = \hat{w}_D \cdot \vec{I}^{T}(t). \tag{4}$$

The operator $\hat{O}$ represented the overall transformation by the network, and $\hat{w}_D$ was the design matrix in the output layer whose rows were the number of observable parameters and columns were the coupling strength to the sensors. While the above assumed a feedforward architecture, Eq. (3) and onwards may be generalized for any network if one allows $\hat{O}$ to represent any network transformation. Learning was achieved by designing the network connections in such a way that the output signal was transformed into a desired target signal. Each desired computational process corresponded to a desired signal, $I^{\text{test}}_{\text{desired}}$, for a given input signal, $I^{D}(t)$, and network transformation, $I^{F}(t)$. The synaptic strength of individual connections was adjusted in incremental corrective steps according to a learning rule using a training dataset until the network's effective function was the desired mathematical operation. Various learning techniques exist and depend on network type, connectivity structure, neuronal transfer function, I/O implementation, task complexity, and computational constraints (Büsing et al. 2010; Haimovici et al. 2013; Nedaaee Oskoee and Sahimi 2011; Sporns 2006). Here, we focused on the linear-regression learning rule as it was the typical and simplest learning rule:

$$\hat{w}_D \hat{O} = \left(\left[\vec{I}^T(t) \right]^\dagger \left[\vec{I}^T(t) \right] \right)^{-1} \left[\vec{I}^T(t) \right]^\dagger I^{\text{training}}_{\text{desired}}(t). \tag{5}$$

The process was done recursively by using a number of controlled training datasets to determine error propagation and correction. Defining a metric for error was nontrivial and clever designs and calculations of error existed that can drastically determine learning performance. However, we focused on the most commonly used and simplest definition of signal error which was the normalized mean squared error (NMSE) and adopted accuracy as a more intuitive measure of performance.

$$error \equiv \frac{E\left[\left(I^{\text{test}}_{\text{desired}} - I^F(t) \right)^2 \right]}{E\left[\left(I^{\text{test}}_{\text{desired}} - E\left[I^{\text{test}}_{\text{desired}} \right] \right)^2 \right]}; \; accuracy \equiv 1 - error \tag{6}$$

A network's computational capability was nearly defined by its network size, size of $\hat{O}$, while simultaneously increasing the complexity in learning. Unfortunately, implementing such a model using traditional photolithography manufacturing inevitably approached the Abbe diffraction limit (Abbe 1873), which was incapable of physically addressing elements on similar scale as current software implemented neural networks. Reservoir computing (Schrauwen et al. 2007; Verstraeten et al. 2009) was a distinguished computational model for scalable neuromorphic hardware as it does not require comprehensive control of the network, omitting the $\hat{O}$ in (5). Learning algorithms only required training on the output layer of neurons, $\hat{w}_D$, while the inner "reservoir" neurons, $\hat{O}$, are unattended and replaced $\vec{I}^T(t)$ with $I^F(t)$ in (5) (Lukoševičius and Jaeger 2009; Schrauwen et al. 2007; Verstraeten et al. 2009). Here, we utilized the reservoir computing paradigm as the functional model for computation in our ASN experiments.

2.1 Nonlinear Circuits

Regardless of the network construction or stimulation, a neural network was not capable of performing complex calculations if individual neurons behaved linearly (Carbajal et al. 2015). A brief proof of the desire for nonlinearity was by *contra positive* and to logically investigate the behavior of a network with purely linear elements. We constrained discussion using the above mathematical formalism where systems were represented only by neurons, and any post-processing or contributions from instrumentation were represented as an appropriate neuron layer. Let individual neurons be defined by the gain of an op-amp circuit to simulate linearity, which simply rescales the amplitude of the input signal as in Eq. (7). Suppose the neurons were fully connected to every other neuron by a fully populated network, maximizing the rank of the transformation matrix $\hat{O}$. Inevitably, a linear combination and convolution

of such neurons only resulted in a linearly behaving network, regardless of network connectivity:

$$let\vec{I}^{T}(t) = \hat{O}\vec{I}^{D}(t) = \hat{\lambda}\vec{I}^{D}(t), \tag{7}$$

$$I^{F}(t) = \hat{w}_{D} \cdot \vec{I}^{T}(t) = \hat{w}_{D} \cdot \hat{\lambda}\vec{I}^{D}(t) = \vec{w}_{D\text{eff}} \cdot \vec{I}^{D}(t), \tag{8}$$

where the transformation in Eq. (7) was replaced by a linear function, while the final signal in Eq. (8) was a linear combination of the driving signal with $\vec{w}_{D\text{eff}} = \sum_{i}^{N} w_{Di}\lambda_{i}$. The above demonstrated linear neurons' limited computational capability and be completely defined by the input signal from Eq. (8), while the design matrix $\vec{w}_{D\text{eff}}$ merely scales the input. To enable the network to implement complex computation, a neuronal behavior with nonlinear characteristics and robust mathematical formalism was adopted. Chua's nonlinear circuit analysis (Chua and Kennedy 1988) introduced the concept of memristive systems as a neuron-like two-terminal element with characteristic nonlinear and memory qualities. The memristor nonlinearly related the integrated voltage (magnetic flux, φ) with charge and acts similar to a charge dependent resistor:

$$d\varphi = MdqorM(q) = \frac{d\varphi}{dt}/\frac{dq}{dt} = V/I. \tag{9}$$

The relation was strictly nonlinear and solved differentially, which required holistic circuit analysis when the element was incorporated within a network (Chua 1980). A memory attribute was readily illustrated in the memristor's dependency on charge accumulation, which was desirable to any learning system. Discovery of physical memristor devices (Strukov et al. 2008) and complex circuit oscillations depicting chaotic trajectories (Chua et al. 1993; Chua and Itoh 2008) has enabled the construction of nonlinear circuits capable of harnessing emergent chaotic behaviors. Observations of neurons physically adapted to environmental changes through a recurrent feedback mechanism (Carbajal et al. 2015) paralleled the oscillatory behavior found in Chua circuits. Likewise, incorporation of a continuous feedback enabled adaptive and responsive computing (Hermans et al. 2015).

2.2 *Characterization: Power-Law Dynamics*

The above mathematical construct illustrated the importance and central role of the network connectivity and functional topology described by the transformation $\hat{O}$ within the machine learning platform. As described by Maass and Legenstein (2005), Maass et al. (2002), neuroscience concluded that a small-world network

maximized information transfer while minimizing energy usage. This phenomenon was observed throughout the natural world—occurring in cases such as complex geological formations, flock behavior, and disease proliferation—and continues to be a central topic within chaos and network theory. Heuristic evidence concluded that network design ascribed with such features enable optimal performance. A defining characteristic of a small-world topology was the length distribution of interacting elements to behave as a power-law, i.e., the strength of $\vec{w}_i^{\,n}$ scales as $d^{-\beta}$, where d is the interneuron distance:

$$\left|\vec{w}_j^{\,m}|d|\vec{w}_i^{\,n}\right| \propto d^{-\beta} or \left|fft\left(\vec{w}_j^{\,m}\right)|t|fft\left(\vec{w}_i^{\,n}\right)\right| \propto f^{-\beta}, \tag{10}$$

where the second relation utilized Pontryagin duality, $d \rightarrow c \times t$ with c as the speed of light, and the Fourier transform of the first. A network adhering to these constraints was capable of sustained persistent activity even due to small perturbations (Goudarzi et al. 2012; Haimovici et al. 2013; Sussillo and Abbott 2009) and was in a "critical" state which allowed for maximal information transfer.

We examined the device for emergent nonlinear properties considered fundamental to brain function, which were not observed for individual atomic switches operating in simpler geometries—namely, recurrent dynamics and the activation of feedforward sub-networks. The presence of small-world dynamics within the ASN devices was demonstrated by applying a constant DC bias (Fig. 2a) across a particular region of the network. This produced persistent, bidirectional fluctuations—both increases and decreases—in network conductivity. In the absence of complex structures within the network, conductivity would increase monotonically under constant DC bias, as in the case of a single atomic switch. Previously reported (Stieg et al. 2012) current fluctuations of this kind are ascribed to recurrent loops in the network

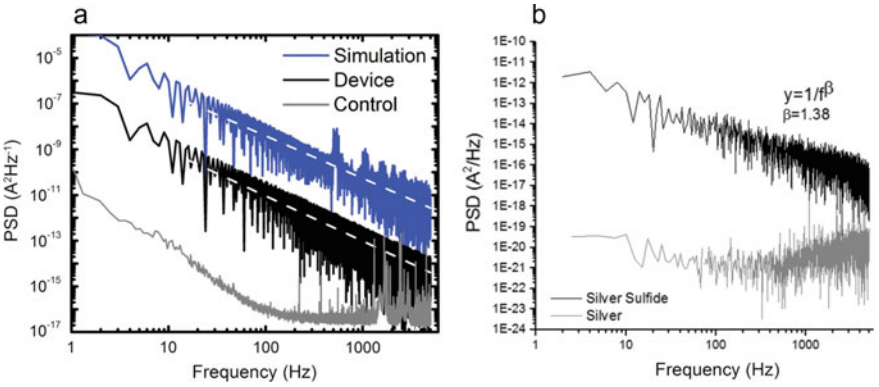


Fig. 2 Dissipative power-law behavior indicative of self-organized criticality. The electrical current response of a physical and simulated ASN device in **a** under constant external voltage bias was used to characterize network activity. Network switching/activity timescales showed **a** dissipative power-law response **b** indicative of a scale-free network

that create complex couplings between switches, resulting in network dynamics that chaotically converge to a semi-steady state even under constant bias. A single switch turning ON did not simply lead to an increased potential drop across the next junction in a serial chain, but entropically redistributed voltage across many recurrent connections that can ultimately perturb the system into a new equilibrium as a net change in network conductivity. These fluctuations were not attributable to uncorrelated flicker noise, as shown by comparing the Fourier transformed current responses in Fig. 2b of the devices to constant voltage before and after sulfurization. The formation of atomic switch junctions expanded the degree of correlation in current fluctuations, producing small-world 1/f-like behavior across the entire sampled range. This behavior was distinct from that of control devices which flattens to white noise and some high energy, high frequency fluctuations attributed to arcing between neighboring wires.

2.3 Atomic Switch Plasticity

In conjunction with optimizing the reservoir's transformation capability in $\hat{O}$, reservoir learning inherently required a memory quality and plasticity for selective information storing (Jaeger 2001). A powerful feature of atomic switches and memristive-like devices was the observation of a brain-like physical phenomena known as *Long-Term Potentiation* (LTP) and *Short-Term Potentiation* (STP). Both function and memory have been ascribed to STP and LTP dynamics in neurological studies (Maass and Legenstein 2005; Maass et al. 2002). Neuron signal transduction through potentiation spikes showed timing dependencies which directly encoded information within the spike's line shape. Simultaneously, brain functionality and behaviors developed as neuron ensembles cooperatively spiked to adopt specific emergent behaviors.

These neurological phenomena were observed (Nelson and Abbott 2000; Sussillo and Abbott 2009) within the active Ag_2S region in the atomic switch as aggregations of Ag^+ cations. Observation of a large impedance change in the atomic switch under an external voltage was attributed to a crystal transition of the active material Ag_2S (Gusev and Sadovnikov 2018). This transition gave rise to a weakly memristive behavior prior to the formation of Ag filaments across the interface. In the absence of continued applied bias, the conductive filaments eventually returned to their stoichiometric, thermodynamically favored equilibrium state, reverting the atomic switch to its initial high OFF resistance (Fig. 3a). Continued application of bias voltage resulted in a concurrent increase in electric current through the device, which then further drove migration of silver cations toward the cathode. At the cathode mobile silver cations were subsequently reduced to Ag^0, forming a highly conductive Ag nanofilamentary wire. The completion of this filament resulted in a strong transition to an ON state (Fig. 3a) with a dramatic increase in conductivity (Fig. 3b). Removal of the applied bias resulted in filament dissolution as the device again returns its thermodynamic equilibrium state (Fig. 3b). The completion and dissolution of this

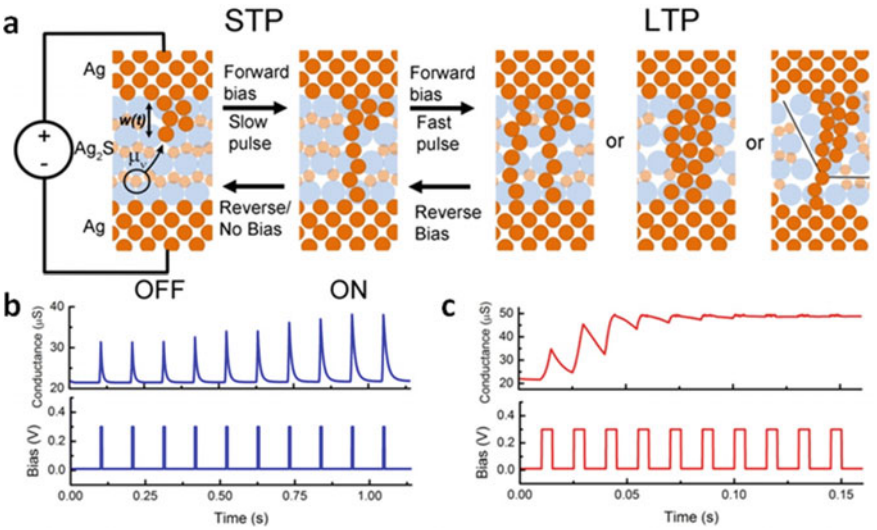


Fig. 3 Spike-time dependent plasticity in a single atomic switch. Continued stimulation of the atomic switch caused formation of metallic filaments across the gap/active layer in **a**. The electrical response became increasingly dominated by tunneling mechanisms derived from single atom "contact". A 300 mV spike 5 ms width voltage train at a period of 100 ms in **b** stimulated the atomic switch to form a single Ag filament. Single atom contact increased conductance to the ON state during stimulation while thermodynamic dissolution drove the system back to the OFF state. In **c**, the pulse train period was shortened to 10 ms allowing multiple filament formations. Measured conductance monotonically increased before reaching a stable conductance state. Filament structure and stability modulated the electrical response and emerge as empirically determined as Short-Term Potentiation (STP) in **b** and Long-Term Potentiation (LTP) in **c**

filament characterized strongly memristive behavior. Continuous application of a bias voltage served to increase filament thickness as additional silver cations was reduced, causing thickening of the metallic filament (Fig. 3c). This dynamic process has been shown to alter the dissolution time constant and can be externally controlled by changing the input bias pattern (e.g., pulse frequency). Such changes in volatility can be interpreted as short-term or long-term potentiation (STP and LTP).

2.4 Resistance Training

The network's ability to physically encode information within the filament led us to develop a resistance training algorithm to control the network's memory capabilities. The dependency of filament formation on voltage history and charge accumulation illustrated memristive behavior within the atomic switch. Circuits utilizing memristive behavior tend to have complex trajectories with nondeterministic solutions and are classified as Chua circuits (Chua et al. 1993). Initial conditions and

stochastic fluctuations helped determine the circuit's trajectories and operational regime, thereby having statistical control on its operation. The circuit parameters of impedance, inductance, and capacitance were used to determine the trajectories of Chua circuits, but other driven systems have included filters, op-amps, and other sources for noise. Though the atomic switches' equivalent parameters evolved with operation, impedance change dominated most of the activity while periphery parameters were treated using thermodynamic approximations (Sillin et al. 2013). A resistance training algorithm was constructed to tune the network operational regime, while using resistance stability as a thermodynamic approximation of the periphery parameters in simulated models, see Sect. 2.5.

The resistance training experiments were performed using a precision source measure unit (National Instruments 4132) and a high-speed switch matrix (National Instruments 2532) within a PXIe unit (National Instruments 8108), enabling rapid resistance measurements between any combinations of 16 chosen electrodes. Resistance training was implemented through repetition of a two-step process as shown in Fig. 4. In the first step, an electrode A was selected randomly and the resistance between this reference and every other electrode was measured using a small (200 mV, 10 ms) bipolar pulse in order to minimize influence on network resistances, as shown in Fig. 4a. The individual resistances of electrode A with each of the other 15 electrodes, R_{Aj}, defined the network state by calculating the total resistance between

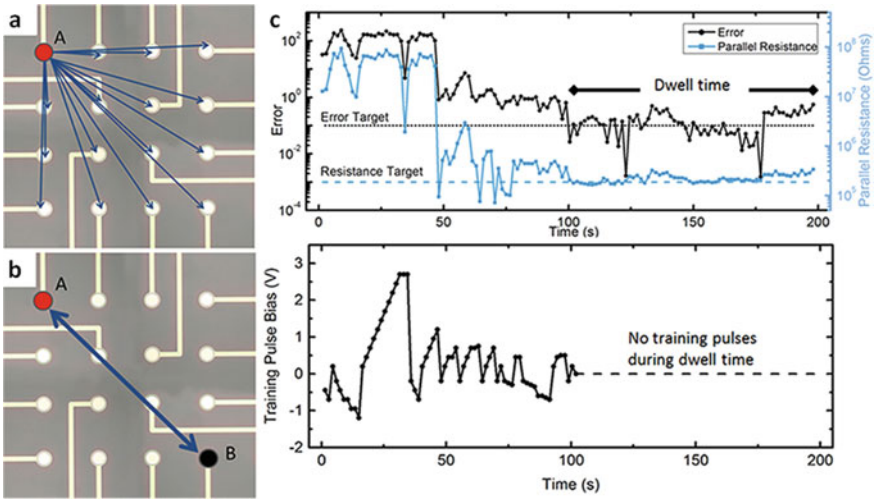


Fig. 4 Resistance learning algorithm. Determination of network-wide stability/activity under operating conditions was conducted using a target resistance learning algorithm. A schematic of the write and verify training scheme, and typical results for an individual training trial. **a** Sub-threshold measurement pulses establish the parallel resistance of A, followed by **b** a larger training/write pulse between A and B. **c** The parallel resistance of A is recorded and compared to the target after each training pulse, when error is minimized the training ceases and the duration of the achieved target state is recorded as the dwell time

electrode A and the rest of the network, as though the paths from electrode A to every other electrode were resistors in parallel:

$$R(i) = \left(\sum_{\substack{j=0 \\ j \neq A}}^{15} \frac{1}{R_{Aj}} \right)^{-1}. \tag{11}$$

This quantity is hereafter referred to as the "parallel resistance". In the second step, a second electrode B was selected randomly, and a large unipolar training pulse (100 ms, >±200 mV) was applied to influence the parallel resistance of electrode A, as shown in Fig. 4b. Using the same electrode I/O scheme, the measure/training cycle was repeated until the parallel resistance of A reached the target resistance. For all trials the target resistance was predetermined, irrespective of the initial network resistance.

In order to achieve training, an error function and rule set was devised. This system was designed to create sensible and consistent voltage adjustments even when both target resistance and parallel resistance error could vary by several orders of magnitude. The error function and rule set also correctly accounted for events in which the parallel resistance overshot the target. Convergence of the parallel resistance to the target resistance was evaluated using an error function:

$$E(i) = \frac{1}{2}\left(\frac{R(i)}{R_g} - \frac{R_g}{R(i)} \right), \tag{12}$$

where R_g was the target resistance, and $R(i)$ was the parallel resistance. The error $E(i)$ was calculated after each pulse/measure cycle, and adjustments to the training pulse bias were made by evaluating the relative change in error $C(i) = \frac{E(i)}{E(i-1)}$ from one cycle to the next using Eqs. (13) and (14), which are described below.

Equation (13) concerned changes in the absolute magnitude of $C(i)$ to evaluate changes in the absolute magnitude of the training pulse, $V(i)$. If the previous training pulse resulted in a large decrease in error, $|C(i)|$ would be less than 1. If significantly less than 1, as determined by an empirically determined threshold, $C_m = 0.6$, then the training pulse $V(i)$ was considered productive and no changes were made. If the previous pulse produced a significant increase in error, $|C(i)|$ would be greater than 1. If $|C(i)|$ was greater than $\frac{1}{C_m}$, the pulse was considered counterproductive and the training pulse magnitude was reset to a minimum value, $V_{\min}$. If $|C(i)|$ was between C_m and $\frac{1}{C_m}$ (i.e., approximately equal to 1) then the error had not significantly changed as a result of the previous pulse, indicating little influence on the parallel resistance. The pulse magnitude was then increased by V_{inc}.

$$V(i+1) = \begin{cases} V(i), if |C(i)| < C_m \\ V_{\min}, if |C(i)| > \frac{1}{C_m} \\ V(i) + V_{\text{inc}}, if C_m < |C(i)| < \frac{1}{C_m} \end{cases} \tag{13}$$

Next, Eq. (14) was used to determine the need for changes to the polarity of the training pulse. If *R(i)* and *R(i – 1)* were both greater or both less than R_g then there was no overshoot and no need to reverse the bias, which is reflected by positive value for *C(i)*. However if *R(i)* changed enough with respect to *R(i – 1)* that it overshot R_g, *C(i)* would be negative. In this case the training pulse voltage *V(i)* was reversed in sign, and its magnitude was automatically reset to the minimum pulse bias $V_{\min}$.

$$sgn(V(i+1) = \left\{ \begin{array}{l} sgn(V(i)), if C(i) > 0 \\ -sgn(V(i)), if C(i) < 0 \end{array} \right\} \tag{14}$$

A single pulse/measurement cycle lasted 1.5 s, and the time required to reach the target resistance state was defined as the "convergence time". Upon reaching the target resistance, training pulses ceased and network resistances were measured every 0.5 s until the parallel resistance decayed away from the target and the error exceeded 0.5 (roughly equivalent to 50% error). This duration was defined as the "dwell time". The entire convergence/dwell time sequence constituted a single resistance training trial, an example of which is presented in Fig. 4c. When a trial completed, new electrodes would be randomly selected and the training process was repeated after a 30 s delay.

Individual resistance states were the result of conductive silver filaments which bridge the Ag|Ag_2S|Ag gaps, and each filament was vulnerable to thermodynamically driven dissolution. Not surprisingly, a deterministic model of interacting thermodynamic variables was not available, and stability of target resistance was hard to predict. Figure 5a shows the distribution of dwell times for networks at the target resistance (R_g=200 kΩ). The distribution suggests a power-law dependency, with

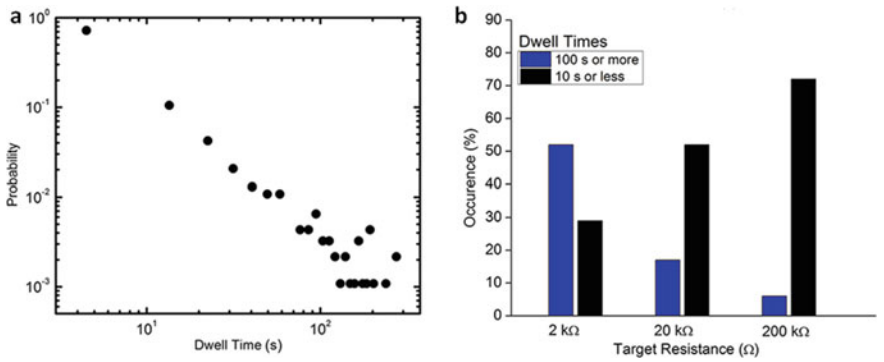


Fig. 5 **Dwell times** vary widely but depend on the target resistance. In **a,** networks are repeatedly trained to 200 kΩ and their dwell times are recorded. By repeating the training program many times on different networks, statistical distributions suggest that the probability *P(D)* of a dwell time lasting for duration *D* follows a power-law relationship. Dwell times are generally 10 s or less, with occasional states lasting 100 s or more. As in **b,** at low target resistances, the final configurations are stable, with over 50% of trials resulting in a final state lifetime of 100 s or more. As target resistance increases, the final states are proportionately less stable

dwell times of less than 10 s being most common and occasionally lasting 10 times longer. This distribution was found to depend heavily on the target resistance value, as shown in Fig. 5b. When R_g=200 kΩ, <10 s dwell times accounted for 72% of trials, but at 2 kΩ, dwell times of 100 s occur in more than 50% of trials. This is the expected result given the underlying operational mechanism of individual atomic switches. Lower resistances were achieved when an individual switch has a thicker conductive filament across the insulating layer, making them more resistant to thermodynamically driven dissolution. In the ASN, lower network resistances are more likely to have an abundance of parallel filamentary pathways, making the target state more resilient against changes from an individual filament. These factors of solved state stability outweigh any effects from repeatedly training the network.

2.5 *Simulation of Atomic Switch Network*

A complementary study on the effects of global stimulation was done in simulation to form a microscopic understanding of the device dynamics. The simulated network was comprised of interconnected atomic switches using a modified state equation (Joshua Yang et al. 2013; Strukov et al. 2008). A current controlled memristor model was adopted undergoing ionic drift dynamics at the Ag|Ag_2S|Ag interface based on previously published works (Demis et al. 2015; Sillin et al. 2013; Biolek et al. 2009). The state variable, $w(t)$, represented the doped region produced by migration of Ag^+ mobile ions from pure Ag into the Ag_2S layer. Reduction of Ag^+ at the cathode precipitated Ag nanowire formation with its physical dimensions determining its impedance and characteristic memristive behavior. The atomic switch was observed to have at least two operational regimes characterized by a low and high resistance state, ON/OFF, respectively. Simple linear super positioning of the two states captured memristive behavior eloquently and a state variable $w(t)$ was defined:

$$V(t) = \left[R_{on} \frac{w(t)}{w_0} + R_{off} \left(1 - \frac{w(t)}{w_0} \right) \right] I(t). \tag{15}$$

Above is the classical Ohm's law equation with $w(t)$, the characteristic filament length, capturing filament formation, and determined using the ionic drift model:

$$\frac{dw(t)}{dt} = \left[\mu_v \frac{R_{on}}{w_0} I(t) \right] \Omega(w). \tag{16}$$

A physical restraint was imposed on $w(t)$ to account for finite dimensions through the use of a window function Ω:

$$\Omega(w) = \frac{w(w_0 - w)}{w_0^2}. \tag{17}$$

Modifications were made to the above model to account for nanowires forming Ag|Ag_2S|Ag interfaces. The formulation for voltage-induced α/β phase transition of the Ag_2S from monoclinic acanthite to the more conductive body-centered cubic argentite was introduced as well as formation/dissolution of conductive filaments (Gusev and Sadovnikov 2018). Applied voltage triggers the α/β phase transition creating a more conductive Ag|β-Ag_2S|Ag junction and, more importantly, allowed for Ag cation migration within the Ag_2S in the direction of the electric field. Reduction of Ag cations into Ag^0 occurs at the cathode thereby creating substructures within the β-Ag_2S to ultimately form conductive filaments. The removal of the applied voltage no longer induced Ag cation migration and the system was allowed to return to its thermodynamically favored equilibrium state. A stochastic term and dissolution term incorporated this thermodynamic behavior to the system. The term further modeled any variability among the nanowires and the structural stability of the Ag filament. Stratonovich integrals were employed to solve the stochastic differential equations. The network was numerically solved as an ordinary differential equation using standard Kirchhoff's current laws with each node–node connection considered as a single atomic switch.

$$\frac{dw(t)}{dt} = \left[\mu_v \frac{R_{on}}{w_0} I(t)\right] \Omega - \tau(w(t) - w_0) + \eta(t) \tag{18}$$

A numerical simulation was constructed based on experimentally determined parameters to model and verify theoretical propositions. Emulating the construction of the device, voltage nodes/electrodes were arranged in a square grid and subsequent node–node connections were introduced to represent nanowires (Fig. 6b). Connections were categorized either as short-range, within a lattice constant, or long-range and randomly assigned to produce characteristics of nearest neighbor or random network topologies (Sillin et al. 2013). The initial strength of each atomic switch was randomly sampled following a power-law distribution in (10) (Maass and Legenstein 2005) with $\beta = 1.38$.

Resistance training was successfully conducted using the simulated ASN device. Network connectivity was created by randomly distributing 250 connections with 10% of the links constrained to a length of a lattice constant within a 5×5 grid. The grid size was increased as previously published SEM images showed connections outside the 4×4 area (Avizienis et al. 2012a, b; Demis et al. 2015; Stieg et al. 2012). Training pulses were administered between two nodes using the scheme described in Eqs. (11)–(14). Resistance training in the simulated network proceeded as observed in the device (Fig. 6) and could involve a direct approach to the target, or through a series of overshoots. The simulation allowed a complete analysis of every change in resistance in each link, and Fig. 6b showed the net change that occurred in each link during the training process. The changes were widespread rather than localized along a single conductive pathway, which supports the hypothesis that network training was achieved by global interactions.

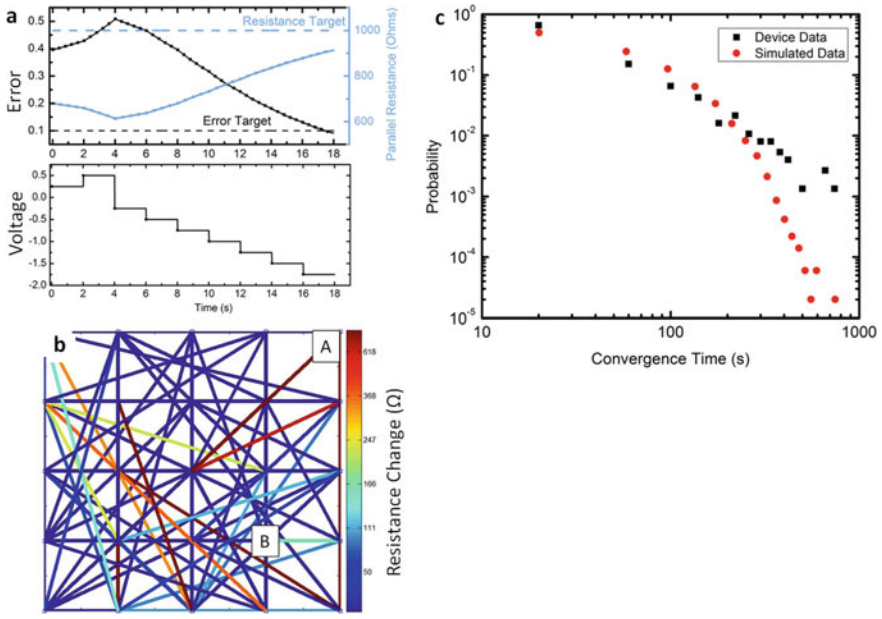


Fig. 6 Resistance learning algorithm convergence of models. A simulated ASN shows similar behavior in resistance training, and network-wide changes in resistance. A parallel resistance training program identical to the experimental one was used to successfully train parallel resistance. **a** Target resistance was 1000 Ω, error target was 0.1, training pulses were 100 ms in 250 mV increments, measurement pulses are not necessary in simulation. The effects of resistance training are presented in **b**, which shows the net resistance change in each link from start to finish. The simulation shows network-wide changes in resistance even though training pulses were applied exclusively from *A* to *B*

2.6 Implementation: Error Checking

As an illustration of the ASN's utility as a reservoir, the benchmark task of determining bit parity was taken to both measure memory quality and network tenability. As outlined in Furuta et al. (2018), Natschläger and Bertschinger (2004), the task was a fundamental algorithm in signal processing and error checking. Typical data streams of bytes of bits required one bit, the parity bit, to record the parity of the overall byte. Information transfer across multiple servers can corrupt data by inverting one bit thereby changing the overall parity of the data byte. The parity bit ensured identification of corrupted bytes and subsequent repairing to allow for reliable data transfers. Typical data bytes are 8 bits long, which our experiment adapted as time-separated binary pulse sequences.

Previously published work (Demis et al. 2015, 2016; Sillin et al. 2013) enabled us to conclude that reservoir computing was not a universal computing paradigm, but more similar to a B-machine as imagined by Turing (1950). As such, the reservoir and task needs to be tailored for optimal utility in performing the parity test. Simulations

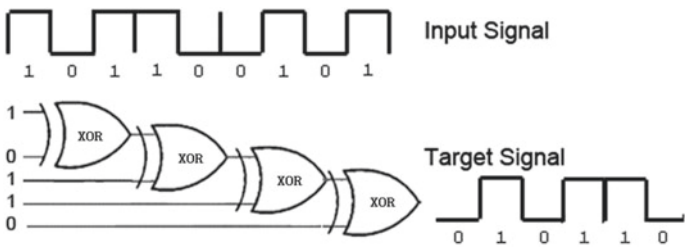


Fig. 7 Error checking task. Presented is an illustration of the parity check used in data transmission for error checking process. The parity of the number of 1's within a 5-bit byte is evaluated with a sliding window 5 bits wide to generate multiple tasks. The initial input shows an odd parity and evaluated as 0 for the desired target signal. As the 5-bit window moves across the signal, the parity changes and reflected in the target signal. The above task was encoded as a voltage pulse sequence into the ASN device where each bit was represented by V_0 or V_1 voltages in a time-separated series. Task complexity increased with increasing number of bits per bytes rather than number of bytes as the check was only executed once per byte

of the ASN were highly leveraged for this purpose for its ease of use, device editing, and high throughput despite statistically underperforming w.r.t. the ASN device. A number of simulations were conducted to determine optimal signal encoding, activation regime, and processing timescales (Fig. 7).

2.7 Simulated ASN Error Checking Results

Implementation of machine learning tasks required the design of an encoding technique such that signal transduction stimulates the network into an excited state with the proper mathematical transformation. As outlined in Sect. 2, reservoir computation can be represented into a mathematical design matrix $\hat{O}$ through spectral analysis where its rank and eigenvalue determine the complexity of the transformation (Verstraeten et al. 2009). However, reservoir size limits the reservoir's computational capability as the rank of the design matrix cannot exceed the readout layer. This limitation is typically overcome by ensuring overlap of the design matrix within the desired mathematical operating regime by applying constraints to network activity. However, signal transduction can perturb the reservoir outside desired activity and clever design of transduction was required such that the signal can both encode information while maintaining the reservoir at a specific state.

Encoding of digital information was explored by modulating the signal in either the amplitude, frequency, or phase space. For the parity test, digital information was spread among 8 bits with each bit in a binary state of either 0 or 1. Bytes of digital information were represented as pulse voltages, Gaussian wave packets, and phase-shifted sine waves for amplitude, frequency, and phase modulation, respectively. The binary states 0 and 1 were assigned to preset voltage amplitudes (V_0, V_1), wave packet frequency shifts (f_0, f_1), and phase shifts (φ_0, φ_1) and the Euclidean

distance between the binary states empirically optimized. Inspection of the state equation in (9) and (16) reveals the dependencies of these parameters w.r.t. network activity. Indeed, simulation results concluded that the total voltage was the deterministic factor on reservoir activity while changes in frequency and phase negligibly perturbed the reservoir. An amplitude-modulated encoding procedure was adopted for all subsequent experiments with voltages ranging from 0.1 to 7.0 V.

Reservoir activity was initialized in simulations using the resistance training algorithm in Sect. 2.4 (Sellers 2007). Resistance values with short convergence times were desired as the resistance training algorithm invariably encoded unnecessary information from the procedure which limited the reservoir's memory capacity. These states coincided with resistance values within R_{ON} or R_{OFF} as well as states that had been thermodynamically stable after repeated approaches. Introducing a highly stochastic signal while maintaining the resistance state was capable of cleansing any information encoded by the resistance training algorithm and was incorporated into a post-experiment protocol. Optimal network activity was determined heuristically while prioritizing reliability over performance.

A typical proportional–integral–derivative (PID) loop algorithm provided a constant feedback voltage which maintained target reservoir activity. Constant stimulation by application of the driving signal eventually accumulated charge and excited the reservoir outside the target resistance state, observed as LTP in Sect. 2.3. Conversely, STP dynamics concluded that inactivity or sub-threshold voltages unable to counterbalance the thermodynamic inhibitive processes relaxed the system. Stability of the target resistance state was controlled by a PID feedback loop by dedicating one of the I/O nodes for this purpose. The feedback applied a constant DC signal for an integral time equal to the training time. Maintenance of the resistance state followed identical trends as the resistance training algorithm in Sect. 2.4.

Optimal training times were determined by maximizing the dwell times at target resistances and empirically investigated in simulation. Learning was implemented on the reservoir using a number of training datasets, following the mathematics in Sect. 2 and details found in Sect. 4.3. Each training set was followed by a testing dataset to determine the effectiveness of the learning algorithm using the accuracy in Eq. (6) as a metric of success. The procedure of providing a training dataset for the learning algorithm and subsequent testing of performance was repeated, while constraining reservoir activity using the PID feedback loop.

The ASN's performance dependencies w.r.t. dataset size and number of learning repetition was investigated in simulation to determine optimal dwell times. Simulations of the ASN device revealed an occurrence of under-learning at 0.250 s (Fig. 8a) and over-learning at 4.000 s (Fig. 8b). This was observed as drastic increases in the NMSE at these timescales as well as a deterioration of signal propagation. The occurrence of over-learning was theoretically predicted as we approached the network's memory capacity by saturating it with training data. Under-learning manifested as fluctuations in performance across various reservoir sizes due to limited memory retention times. The over-learning occurred as the learning algorithm became ill-posed and over-determined with excessive training sets. Optimum dataset lengths

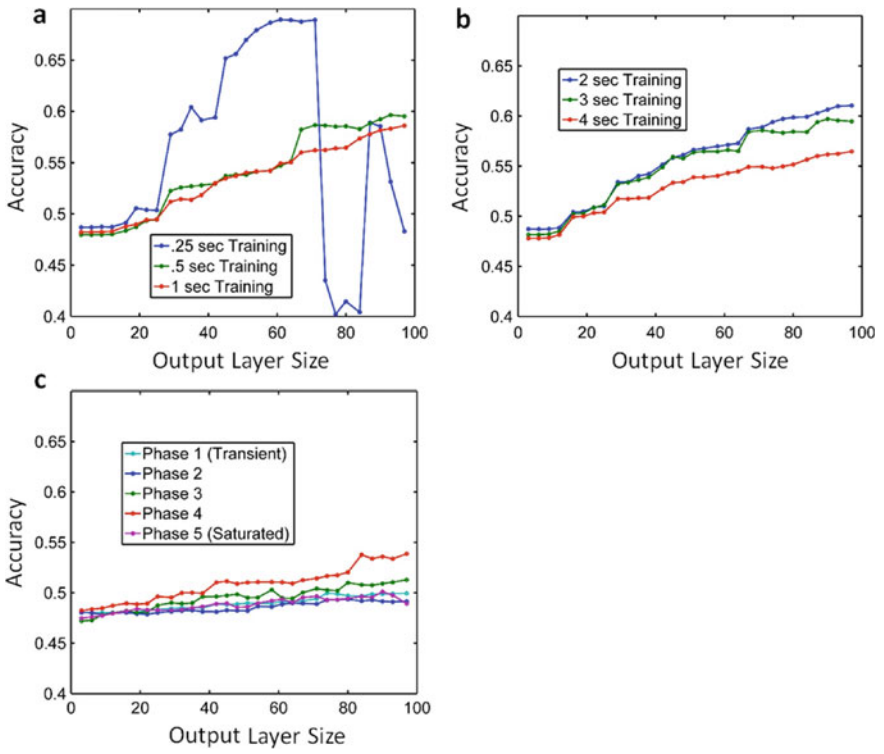


Fig. 8 Encoding optimization using ASN simulation platform. A simulation of the ASN device performing the parity check task was conducted to determine optimal operating parameters. Temporal memory quality was evaluated w.r.t. the size of the output layer, length of the learning sequence used a, b, and operating time c. Under-learning was observed at 0.25 s (blue) length datasets as chaotic performance was measured regardless of network size. Over-learning in b at 4.00 s (red) as continued increase in the dataset length reduced reservoir performance. Subsequent phases of operation c each 1.00 s in duration determined optimal operating time. Omitting the transient phase (light blue), subsequent phases monotonically increased performance and peaked at 4.00 s (red) while further operation in phase 5 decreased performance

were discovered to be 1.000 s while optimal total operating time to be 4.000 s (Fig. 8c). Subsequent experiments were thus encoded as amplitude-modulated datasets 1.000 s in length with 0.250 s pulse width and learning applied within a 2.000 s window.

2.8 Neuromorphic ASN Device Error Checking Results

The optimal parameters found from simulation were implemented on the ASN device and investigated for routes of optimization. Identical instrumentations were used as the resistance training algorithm while incorporating a PID feedback mechanism

for sustaining reservoir activity (Fig. 9a). The error checking task was implemented over a population of 5 devices using all possible permutations of the 16 I/O electrodes and followed similar trends as depicted in Fig. 9. Initial experimentation on

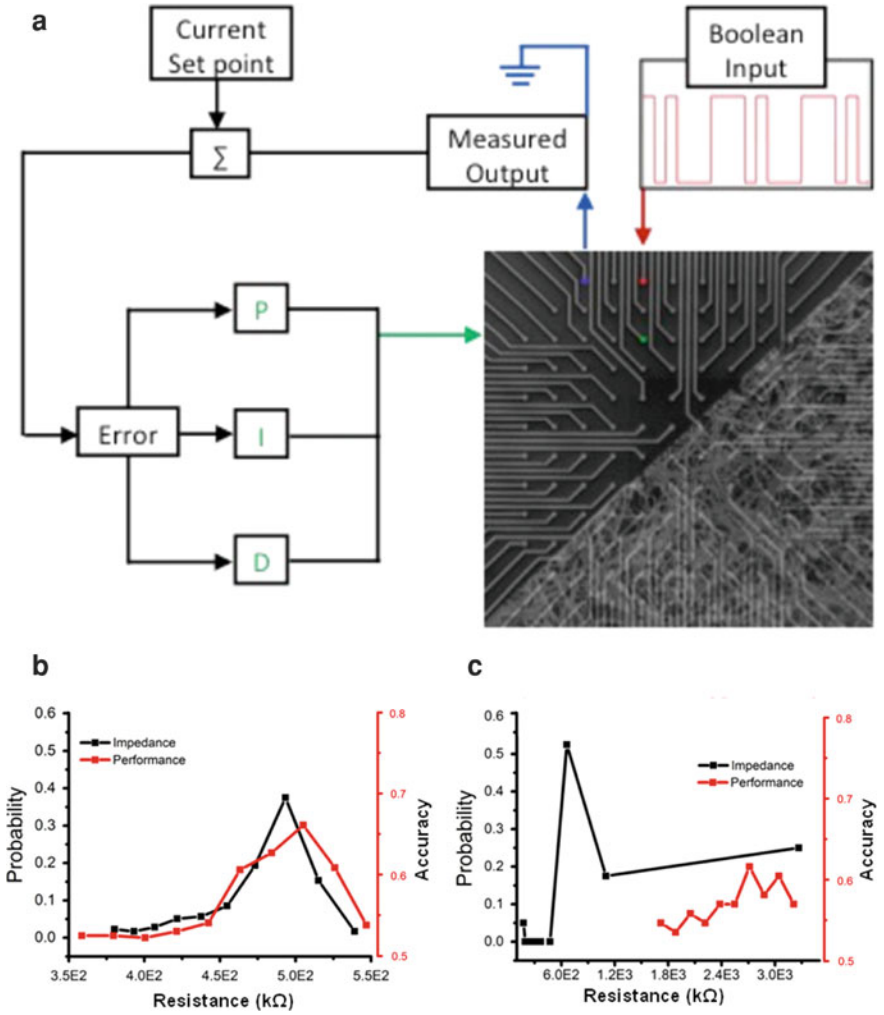


Fig. 9 Error checking of ASN platform. Schematic of RC using ASN devices: Three I/O electrodes are selected to form the stimulus/control loop for RC: Boolean input streams are delivered to an individual I/O electrode underlying the ASN network (red); a system ground (blue) enables real-time monitoring of current flowing through the network controlled by a feedback-driven bias voltage delivered to (green) a nearby location. The ASN was stimulated with a statistical survey of pulse widths ($n\Delta t$) and pulse heights ($n\Delta V$) ranging from 250 ms to 0.01–7.00 V. Testing occurred immediately after resistance training with a fixed weight configuration. The datasets above achieved accuracies **a**, **b** between 65 and 78% from ~5,000 trials compared to ~50% from a purely stochastic reservoir

individual ASN devices was performed to determine relevant optimal amplitude and timescales using identical procedures as in simulation. The simulation's predicted optimal parameters were corrected to include amplitude scales of 0.01–7.0 V while other parameters were retained.

Device performance and reliability were heavily dependent on the device's resistance state following their description in Sect. 2.4. Shorter dwell time devices at correspondingly higher resistance states performed with increasing reliability and accuracy, despite implementing similar training and operating times with devices at lower resistance states. A bimodal distribution of metastable resistance states was found with dwell times that exceeded 100 s for 3 different devices that followed similar dynamics to Fig. 5a. The presence of these metastable states and similar power-law behavior indicated the device activating toward a self-organized critical state (Goudarzi et al. 2012; Stieg et al. 2012) with the two resistance states centered at 500 kΩ and 600 kΩ possible chaotic attractor states. However, true verification of device criticality required statistical experimentation using exact and identical parameters which proved impractical.

Continued trials revealed devices initialized outside of the near-critical resistance states performed poorly with accuracies below 50%, which prompted subsequent device testing to operate within the bimodal states to explore device optimization. Devices initialized below 500 kΩ (Fig. 9b) performed at 71.35% ± 6.38% accuracy while those initialized above 600 kΩ attained a similar and maximized performance of 73% ± 5%. Despite seemingly small differences, this trend manifested throughout all trials alongside a characteristic high dispersion in the distribution with kurtosis values of 2.73 and 5.94 for devices at 600 kΩ and 500 kΩ, respectively. Kurtosis values beyond 3 indicated a non-Gaussian distribution and increasingly became dominant below 500 kΩ. Rapid bipolar switching manifesting as abrupt changes in current supply was observed below this range and simulation experiments revealed increased filaments forming under similar conditions. The non-Gaussian statistics and filament completion events indicated a shift in the operational characteristics of atomic switches and decreased performance metrics. Consequently, network resistance state became increasingly complex and divergent thereby driving network dynamics toward increasingly nonlinear behaviors and outside target functionality. Past results (Demis et al. 2016; Sillin et al. 2013) and similar experiments (Carbajal et al. 2015; Hermans et al. 2015) clearly indicated the requirement for task-specific network design. Diverging resistance states, dynamic changes in atomic switch behaviors, and poor performance concluded the network was being driven outside of its error checking design.

Further experimentation evaluating other device parameters such as stimulation amplitude, size, and learning timescales resulted in minor changes to network performance, highlighting the importance of network dynamics. The stability of the network resistance state was an evident metric in controlling computational capability and network state. Spontaneous organization of 2 convergent resistance states highlights the underlying critical dynamics which maximized device performance. Controlling such device dynamics through the use of mechanisms such as a feedback loop (Hermans et al. 2015) seems evident for further progress.

3 Outlook

We concluded that we were able to use network activity and stability via resistance initialization to describe the network state for a thermodynamically driven reservoir, the ASN. Due to the task-specificity inherit in machine learning, it was paramount to characterize and catalogue a reservoir's "state" that corresponds to task-specific functionalities. Likewise, previous results (Sillin et al. 2013) developed a map for pattern recognition using higher harmonics. Typical reservoir characterization in the literature utilized entropy and Shannon theory, which requires repeated experiments under identical conditions. Current devices utilizing memristor-like reservoirs are difficult to control with such precision, thus, a desire for an alternative characterization of reservoir state has been necessary. In general, characterization of reservoir functionality has proven difficult for real "edge-of-chaos" systems. Although this requirement has strictly not been within the reservoir computing framework, the development of reservoirs with diverse and rich functionality expands the framework's utility.

Despite limited addressable electrodes, the ASN device was capable of outperforming simulated networks as network complexity, density, and critical dynamics were utilized more effectively in the device. We have presented a clear methodology to implement reservoir computing on a neuromorphic device by developing observable metrics such as power-law behavior, activation of STP/LTP, and resistance state. As outlined in Sect. 2, reservoir performance was theoretically predicted to depend on nonlinear dynamics, network topology, and task design. The commensurate development of simulations aided in implementing theoretical models onto neuromorphic a platform and task evaluation. Previously accomplished tasks such as pattern classification, bit logic, and T-maze decision-making task highlighted the capabilities of the atomic switch as an integrated memory and logic component.

4 Methods

Structurally complex networks comprising of highly interconnected, functional nanostructures are fabricated using varying degrees of top-down and bottom-up processes, ranging from the random deposition of monodisperse nanowires to the electroless deposition (ELD) of metallic nanostructures. By identifying the benefits and limitations of each technique, a nanoarchitectonics approach was adopted (Demis et al. 2015, 2016) whereby the size of nucleation sites for ELD was used as a control parameter for network growth in order to maximize atomic switch connectivity while retaining control over network topology. The silver networks were functionalized to have atomic switch interfaces at the junctions of their component nanostructures. This process produced a complex network of interacting elements whose operational properties provide a basis for the memorization and transformation of environmental information. Further, their inherent volatility results in patterns of robust electrical

activity. Through this specific combination of nanoscale elements and design principles for the production of structurally complex systems, ASN devices provide the balance of intrinsic memory capacity and nonlinear operation required for advanced hardware implementations of neuromorphic computing, specifically in the reservoir computing paradigm.

Patterned seed networks proved the most versatile fabrication method, and utilized a combination of top-down with bottom-up fabrication, a powerful fabrication method described as nanoarchitectonics. Random neural networks are grown on a SiO_2 substrate with Cu post nucleation sites via electroless deposition producing a massively interconnected nanowire network with functionally brain-like features (Avizienis et al. 2012a, b). A patterned 150 nm layer of platinum electrodes are prepared on the substrate with a 500 nm layer of SU-8 polymer insulating the platinum, only circular nodes 30–50 μm in diameter are left exposed for electrical contact and arranged in a square grid. Standard lithographic techniques using a negative resist were used to then deposit pre-pattern Cu posts serving as nucleation sites. Dendritic nanowires are grown via electroless deposition with a 50 mM $AgNO_3$ solution controlling the size and shape of Cu posts to control the topological distributions of the Ag nanowire network (Avizienis et al. 2013). Exposure to sulfur gas at 10^{-1} Torr at 130° C for 3 min developed functional Ag|Ag_2S|Ag interfaces.

With a density controlled network in mind, our group started using electrochemistry to grow a recurrent silver network using copper seeds. Network growth occurs through an electroless deposition (ELD) reaction through individual atom displacement reactions between Ag^+ and Cu^0 based on respective electric potentials. A spontaneous ELD reaction is preferred over an electric one due to the lack of a need for external power and the delicate nature of electrochemical reactions. In this particular case, silver atoms are oxidized while copper is reduced during the galvanic displacement reaction:

$$Cu^0(s) + 2Ag^+(aq) \rightarrow 2Ag^0(s) + Cu^{2+}(aq)\ E_{red} = -1.26\ V.$$

Successful implementations of the ELD reaction above allowed us to design a technique using highly patterned top-down photolithography combined with the complex spontaneous growth provided from the reaction above. These patterned seed networks consist of a 2 μm layer of AZ nLOF 2020 (a negative photoresist), a soft bake, followed by UV photolithography, and a post-exposure bake. This resist is developed in MF26A, rinsed with isopropanol, and a 300 nm layer of copper is then deposited and lifted off overnight in acetone. At the end of this process, a patterned grid of copper posts 300 nm high is left. The size and pitch of these posts were refined over time to give the most desirable silver crystal growth.

When first designing a purpose-built device to emulate mammalian brain activity, dendritic silver structures were desired. However, over time it was realized that the connections provided by these structures were unreliable and difficult to reproduce. Through changing the size of the copper posts, a morphological transition was found showing that a seed site of $1 \times 1\ \mu m^2$ up to $3 \times 3\ \mu m^2$ leads to fine nanowires. Seeds

between $3 \times 3\ \mu m^2$ and $10 \times 10\ \mu m^2$ yield a mixture of nanowires with branched dendritic structures, while posts larger than $10 \times 10\ \mu m^2$ produce only dendrites.

4.1 Design Optimization Results

Directed nanowire growth to create dendritic structures followed diffusion-limited aggregation with Mullins–Sekerka instabilities. For sparse concentrations of $AgNO_3$, diffusion-limited aggregation (DLA) dynamics prevail where Ag^+ cations displace Cu^0 atoms in discrete non-interacting reactions. Reduced silver atoms accumulate on the surface of the copper posts and develop, in a steady-state evolution, metallic nanostructures. Solidification of silver particles undergoing DLA obeys the mathematical formulations of Fick's Law, modified by Mullins–Sekerka instabilities which describe pattern formation of accumulated metal nanostructures. We describe the kinetics of formation using ion clusters to describe the heterogeneity of the solution's concentration. Clusters of ions diffuse through the solution, creating a wave of ions that initiate the ELD process at the seed. Starting with Fick's law to describe the diffusion:

$$D_{Ag}\nabla^2\mu_{Ag} = \frac{\partial\mu_{Ag}}{\partial t};\ D_{Cu}\nabla^2\mu_{Cu} = \frac{\partial\mu_{Cu}}{\partial t}; \tag{19}$$

Here we use D_{Ag} and D_{Cu} the diffusion constants for $AgNO_3$ and pure copper, respectively, with as the diffusion potential. As the wavefront of silver reacts with copper, aggregated silver atoms at the seed sites accumulate, pushing the growth front toward the wavefront. The solid–liquid interface perturbs the diffusion field, moving slowly and continuously renormalizing the ion gradient in solution. Growth of the solid–liquid interface, via the non-equilibrium process of electroless deposition, is mediated by the continuity equation:

$$Mv_n = \left[\delta\mu_{Ag}D_{Ag}\nabla\mu_{Ag} - \delta\mu_{Cu}D_{Cu}\nabla\mu_{Cu}\right]\cdot\hat{n}. \tag{20}$$

The miscibility gap, M, and the normal velocity, v_n, of the interface, determine the population exchange during single displacement reactions with $\delta\mu_{Ag}$ as the fluctuation in chemical potential due to concentration heterogeneity. Growth of the solid–liquid interface results in the Mullins–Sekerka instability which is due to competing dynamics between steady silver nanostructure growth and dynamical expansion of the growth front. Once the rate of metal nanostructure growth exceeds the diffusion rate, a depletion region emerges that no longer contains enough silver atoms for sustainable displacement. Regions adjacent to the initial growth front contain sufficient ion concentration to participate in ELD, forming side branches. Depending on the rate of formation, the chemical potential at the interface is described by the Gibbs–Thomson boundary condition:

$$\mu(r_0) = -d_0\varepsilon. \tag{21}$$

Here, the chemical potential at the interface, $\mu(r_0)$, is dependent on the surface curvature, ε, and d_0 the characteristic length of the seed. In the simplest case, the value of Eq. (21) was approximated to be the value of Eq. (20) during equilibrium. The non-equilibrium process at the solid–solution interface determines a characteristic length scale:

$$d_0 = \frac{\gamma}{M}. \tag{22}$$

where γ is the surface tension. Solutions to Eqs. (19)–(22) for a planar interface with small perturbations in ionic concentrations are solved in (Langer 1980). Extending this model for multiple perturbations will show dendritic growth that we experimentally demonstrate in controlled fabrication of the ASN. Equations (21) and (22) show parameters of control over the morphology of seed-directed nanowire growth. The reactivity dependence on curvature can be controlled by varying shape, size, and pitch of copper seeds. Surface tension and miscibility gap can be controlled through varying the copper spacing and distribution. Understanding DLA under Mullins–Sekerka instability conditions provides control and reproducibility over self-organizing nanowire networks. Pattern formation due to Mullins–Sekerka instabilities presented here is a linear approximation of the dynamical behavior of dendrite formation. Experimental testing confirmed that when the size of the copper seed is on the order of 1–5μ Mullins–Sekerka instabilities are suppressed, and the growth of metallic nanowires continues without nucleation of side branches.

In order to explore the concept of fabrication through self-organization, the mathematical principles of diffusion-limited aggregation (DLA) and ELD are combined to guide a nanoarchitectonics approach using the electroless deposition of silver. An extensive experimental study of this fabrication method found that the critical parameter for the growth of nanowires was the size of the copper seed post which is theoretically predicted in Eq. (22) due to the factors of surface curvature and surface tension (Avizienis et al. 2013). The diverse wire lengths included long-range and short-range atomic switches, facilitating both globally and locally distributed patterns of switching activity in the ASN. Due to the variation in nanowire diameters, we infer each junction to have a variable gap size and subsequent atomic switch size, thereby increasing the number of available resistance states to the ASN (Avizienis et al. 2012a, b; Stieg et al. 2012). This fabrication method offers control over network density and structure by introducing two important parameters: seed size and spacing, which nucleate wire growth.

4.2 Hardware and Instrumentation

Electrical characterization of the devices was conducted through current–voltage (I–V) spectroscopy using a bipotentiostat (Pine Instruments model AFCBP1) in conjunction with either a data acquisition module (National Instruments USB 6259) or a multiplexed (National Instruments PXI 1073) source-measurement unit (National Instruments PXI 4130). The maximum bandwidth of the measurement systems was 1 MHz and 10 kHz enabling 2 Ms and 20 ks s^{-1} with 16-bit resolution, respectively. ASN devices were designed to accommodate 64 electrode 40 μm Pt contacts within a $2.5 \times 2.5\ mm^2$ grid where atomic switches were grown. Subsequent data analyses were carried out using MATLAB 2018b (MathWorks) and Origin 8.1 (OriginLab Corporation).

4.3 Reservoir Computing Implementation

All reservoir experiments were conducted on an 8×8 grid containing an estimated 10^8 atomic switch junctions using the 64 electrodes as I/O interface layers. A single electrode was selected to inject the electrical input signal, while another electrode was chosen as the counter electrode as shown in Fig. 9a. The control signal delivered a feedback voltage to an electrode in proximity to the input electrode. Voltage signals were simultaneously measured from the remaining 61 electrodes using the data acquisition module (National Instruments USB 6259). Reservoir computing was implemented following the mathematics presented in Sect. 2 with the input layer consisting of only the input electrode and the output layer constructed from the 61 measuring electrodes.

References

E. Abbe, *Contributions to the Theory of the Microscope and the Microscopic Perception* (Springer, 1873)

A. Abraham, Artificial neural networks, in *Handbook of Measuring System Design*, ed. by P.H. Sydenham, R. Thorn (Wiley, 2005)

N. Aramaki, Y. Shimokawa, Y. Fuwa, A parallel ASIC VLSI neurocomputer for a large number of neurons and billion connections per second speed, in *IEEE International Joint Conference on Neural Networks* (Singapore, 1991)

A. Ascoli, R. Tetzlaff, L.O. Chua, *Continuous and Differentiable Approximation of a TaO Memristor Model for Robust Numerical Simulations.* Springer Proceedings in Physics (2017)

R.C. Atkinson, R.M. Shiffrin, Human memory: a proposed system and its control processes. Psychol. Learn. Motiv. **2**, 89–195 (1968)

A.V. Avizienis, H.O. Sillin, C. Martin-Olmos, H.H. Shieh, M. Aono, A.Z. Stieg, J.K. Gimzewski, Neuromorphic atomic switch networks. PLoS ONE **7**(8), e42772 (2012a)

A.V. Avizienis, H.O. Sillin, C. Martin-Olmos, H.H. Shieh, M. Aono, A.Z. Stieg, J.K. Gimzewski, Neuromorphic atomic switch networks. PloS One **7**, e42772 (2012b)

A.V. Avizienis, C.M.-O. Henry, O. Sillin, M. Aono, J.K. Gimzewski, A.Z. Stieg, Morphological transitions from dendrites to nanowires in the electroless deposition of silver. Cryst. Growth Des. 13 (2013)
J.W. Backus, Can programming be liberated from the von Neumann style? A functional style and its algebra of programs. Commun. ACM **21** (1978)
Z. Biolek, D. Biolek, V. Biolkova, SPICE model of memristor with nonlinear dopant drift. Radioengineering **18** (2009)
L. Büsing, B. Schrauwen, R. Legenstein, Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons. Neural Comput. **22**, 1272–1311 (2010)
J.P. Carbajal, J. Dambre, M. Hermans, B. Schrauwen, Memristor models for machine learning. Neural Comput. **27**, 725–747 (2015)
G. Chen, X.F. Wang, Complex networks: small-world, scale-free and beyond. IEEE Circuits Syst. Mag. **3**, 6–20 (2003)
L.O. Chua, Device modeling via basic nonlinear circuit elements. IEEE Trans. Circuits Syst. **27** (1980)
L.O. Chua, M.P. Kennedy, Neural networks for nonlinear programming. IEEE Trans. Circuits Syst. **35** (1988)
L.O. Chua, C.W. Wu, Synchronization in an array of linearly coupled dynamical systems. IEEE Trans. Circuits Syst.-I Fundam. Theory Appl. **42** (1995)
L.O. Chua, M. Itoh, Memristor oscillators. Int. J. Bifurc. Chaos **18** (2008)
L.O. Chua, C.W. Wu, A. Huang, G.-Q. Zhong, A universal circuit for studying and generating chaos-Part I: routes to chaos. IEEE Trans. Circuits Syst.-I: Fundam. Theory Appl. **40** (1993)
E.C. Demis, R. Aguilera, H.O. Sillin, K. Scharnhorst, E.J. Sandouk, M. Aono, A.Z. Stieg, J.K. Gimzewski, Atomic switch networks nanoarchitectonic design of a complex system for natural computing. Nanotechnology **26** (2015)
E.C. Demis, R. Aguilera, K. Scharnhorst, M. Aono, A.Z. Stieg, J.K. Gimzewski, Nanoarchitectonic atomic switch networks for unconventional computing. Jpn. J. Appl. Phys. **55** (2016)
C. Du, F. Cai, M.A. Zidan, W. Ma, S.H. Lee, W.D. Lu, Reservoir computing using dynamic memristors for temporal information processing. Nat. Commun. **8** (2017)
T. Furuta, K. Fujii, K. Nakajima, S. Tsunegi, H. Kubota, Y. Suzuki, S. Miwa, Macromagnetic simulation for reservoir computing utilizing spin dynamics in magnetic tunnel junctions. Phys. Rev. Appl. **10**, 034063 (2018)
A. Graves, A.M. Mohamed, G. Hinton, Speech Recognition with Deep Recurrent Neural Netw. (2013)
A. Goudarzi, C.T. N. Gulbahce, T. Rohlf, Emergent criticality through adaptive information processing in Boolean networks. Phys. Rev. Lett. 108 (2012)
A.I. Gusev, S.I. Sadovnikov, Effect of small size of particles on thermal expansion and heat capacity of Ag2S silver sulfide. Thermochim. Acta 660 (2018)
A. Haimovici, E.T. Pablo Balenzuela, D.R. Chialvo, Brain organization into resting state networks emerges at criticality on a model of the human connectome. Phys. Rev. Lett. 110 (2013)
T. Hasegawa, A. Nayak, T. Ohno, K. Terabe, T. Tsuruoka, J.K. Gimzewski, M. Aono, Memristive operations demonstrated by gap-type atomic switches. Appl. Phys. A **102**, 811–815 (2011)
M.H. Hassoun, *Fundamentals of Artificial Neural Networks* (MIT Press, 1995)
D.O. Hebb, *Organization of Behavior* (Wiley, New York, 1950)
M. Hermans, M. Burm, T. Van Vaerenbergh, J. Dambre, P. Bienstman, Trainable hardware for dynamical computing using error backpropagation through physical media. Nat. Commun. **6** (2015)
J.J. Hopfield, Artificial neural networks. IEEE Circuits Devices Mag. **4**, 3–10 (1988)
C.P. Husband, S.M. Husband, J.S. Daniels, J.M. Tour, Logic and memory with nanocell circuits. IEEE Trans. Electron Devices **50** (2003)
H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks - with an Erratum note. *GMD Report 148.* German National Research Center for Information Technology (2001)

J. Joshua Yang, D.B. Strukov, D.R. Stewart, Memristive devices for computing. Nat. Nanotechnol. **8**, 13–24 (2013)
J.-H. Kim, K.-H. Han, Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. IEEE Trans. Evol. Comput. **6**, 580–593 (2002)
A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. (2012)
Lang, (1986) https://doi.org/10.1063/1.97114
J.S. Langer, Instabilities and pattern formation in crystal growth. Rev Mod Phys. **52** (1980)
C.G. Langton, Computation at the edge of chaos: phase transitions and emergent computation. Phys. D **42**, 12–37 (1990)
M. Lukoševičius, H. Jaeger, Reservoir computing approaches to recurrent neural network training. Comput. Sci. Rev. **3**, 127–149 (2009)
W. Maass, R. Legenstein, What makes a dynamical system computationally powerful? in *New Directions in Statistical Signal Processing: From Systems to Brain*, ed. by P. Haykin, T.J. Sejnowski, J. Mcwhirter (2005)
W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. **14**, 2531–2560 (2002)
C. Mead, Neuromorphic electronic systems, in *IEEE* (IEEE, 1990)
Möller, (1987) https://doi.org/10.1103/PhysRevB.36.1284
T. Natschläger, N. Bertschinger, Real-time computation at the edge of chaos in recurrent neural networks. Neural Comput. **16** (2004)
NEC, NEC integrates NanoBridge in the Cu interconnects of Si LSI (2009), https://phys.org/news/2009-12-nec-nanobridge-cu-interconnects-si.html
E. Nedaaee Oskoee, M. Sahimi, Electric currents in networks of interconnected memristors. Phys. Rev. E **83** (2011)
L.F. Nelson, S.B. Abbott, Synaptic plasticity: taming the beast. Nat. Neurosci. 3, 1178 (2000)
T. Ohno, T. Hasegawa, T. Tsuruoka, K. Terabe, J.K. Gimzewski, M. Aono, Short-term plasticity and long-term potentiation mimicked in single inorganic synapses. Nat. Mater. **10**, 591–595 (2011)
A. Romero, P.L. Carrier, A. Erraqabi, T. Sylvain, A. Auvolat, E. Dejoie, M.-A. Legault, M.-P. Dubé, J.G. Hussin, Y. Bengio, Diet networks: thin parameters for fat genomics (2017)
B. Schrauwen, D. Verstraeten, J. Van Campenhout, An overview of reservoir computing: theory, applications and implementations, in *15th European Symposium on Artificial Neural Networks* (2007), pp. 471–482
D. Sellers, An overview of proportional plus integral plus derivative control and suggestions for its successful application and implementation (2007), https://web.archive.org/web/20070307161741/http://www.peci.org/library/PECI_ControlOverview1_1002.pdf
H.O. Sillin, R. Aguilera, H.-H. Shieh, A.V. Avizienis, M. Aono, A.Z. Stieg, J.K. Gimzewski, A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing. Nanotechnology **24**, 384004 (2013)
O. Sporns, Small-world connectivity, motif composition, and complexity of fractal neuronal connections. Biosystems **85**, 55–64 (2006)
A.Z. Stieg, A.V. Avizienis, H.O. Sillin, C. Martin-Olmos, M. Aono, J.K. Gimzewski, Emergent criticality in complex Turing B-type atomic switch networks. Adv. Mater. **24**, 286–293 (2012)
D.B. Strukov, G.S. Snider, D.R. Stewart, R.S. Williams, The missing memristor found. Nature **453**, 80–83 (2008)
D. Sussillo, L.F. Abbott, Generating coherent patterns of activity from chaotic neural networks. Neuron **63** (2009)
K. Terabe, T. Nakayama, T. Hasegawa, M. Aono, Formation and disappearance of a nanoscale silver cluster realized by solid electrochemical reaction. J. Appl. Phys. **91**, 10110–10114 (2002)
Y. Timofeeva, S. Coombes, Sparks and waves in a stochastic fire-diffuse-fire model of Ca2 release. Phys. Rev. E **68** (2003)
J.M. Tour, L. Cheng, D.P. Nackashi, Y. Yao, A.K. Flatt, S.K. St. Angelo, T.E. Mallouk, P.D. Franzon, NanoCell electronic memories. J. Am. Chem. Soc. **125**, 13279–13283 (2003)

T. Tsuchiya, M. Ochi, T. Higuchi, K. Terabe, M. Aono, Effect of ionic conductivity on response speed of SrTiO3-based allsolid-state electric-double-layer transistor. ACS Appl. Mater. Interfaces **7** (2015)

T. Tsuruoka, T. Hasegawa, K. Terabe, M. Aono, Operating mechanism and resistive switching characteristics of two- and three-terminal atomic switches using a thin metal oxide layer. J. Electroceram. **39** (2017)

A.M. Turing, Computing machinery and intelligence. Mind **59**, 433–460 (1950)

H. van Houten, C. Beenakker, Quantum point contacts. Phys. Today **49** (1996)

D. Verstraeten, Reservoir computing: computation with dynamical systems. PhD thesis, Ghent University (2009)

M.M. Waldrop, The chips are down for Moore's law. Nature **530**, 144–147 (2016)

Part V
Physical Implementations: Spintronics Reservoir Computing

Reservoir Computing Leveraging the Transient Non-linear Dynamics of Spin-Torque Nano-Oscillators

Mathieu Riou, Jacob Torrejon, Flavio Abreu Araujo, Sumito Tsunegi, Guru Khalsa, Damien Querlioz, Paolo Bortolotti, Nathan Leroux, Danijela Marković, Vincent Cros, Kay Yakushiji, Akio Fukushima, Hitoshi Kubota, Shinji Yuasa, Mark D. Stiles, and Julie Grollier

Abstract Present artificial intelligence algorithms require extensive computations to emulate the behavior of large neural networks, operating current computers near their limits, which leads to high energy costs. A possible solution to this problem is the development of new computing architectures, with nanoscale hardware components that use their physical properties to emulate the behavior of neurons. In spite of multiple theoretical proposals, there have been only a limited number of experimental demonstrations of brain-inspired computing with nanoscale neurons.

M. Riou (✉) · J. Torrejon · F. Abreu Araujo · P. Bortolotti · N. Leroux · D. Marković · V. Cros · J. Grollier
Unité mixte de physique CNRS/Thales, Université Paris-Sud, Université Paris-Saclay, 91767 Palaiseau, France
e-mail: mathieu.riou@yahoo.fr

J. Torrejon
e-mail: jtorrejon81@gmail.com

F. Abreu Araujo
e-mail: flavio.abreuaraujo@uclouvain.be

P. Bortolotti
e-mail: paolo.bortolotti@thalesgroup.com

N. Leroux
e-mail: nathan.leroux@cnrs-thales.fr

D. Marković
e-mail: danijela.markovic@cnrs-thales.fr

V. Cros
e-mail: vincent.cros@cnrs-thales.fr

J. Grollier
e-mail: julie.grollier@cnrs-thales.fr

S. Tsunegi · K. Yakushiji · A. Fukushima · H. Kubota · S. Yuasa
National Institute of Advanced Industrial Science and Technology (AIST), Spintronics Research Center, Tsukuba, Ibaraki 305-8568, Japan
e-mail: tsunegi.sb@aist.go.jp

K. Yakushiji
e-mail: k-yakushiji@aist.go.jp

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_13

Here we describe such demonstrations using nanoscale spin-torque oscillators, which exhibit key features of neurons, in a reservoir computing approach. This approach offers an interesting platform to test these components, because a single component can emulate a whole neural network. Using this method, we classify sine and square waveforms perfectly and achieve spoken-digit recognition with state of the art results. We illustrate optimization of the oscillator's operating regime with sine/square classification.

1 Context

Artificial intelligence has attracted interest because it offers the possibility of machines outperforming humans at cognitive tasks such as image or speech recognition. In a data-driven society, artificial intelligence will become more and more essential as many industries begin to automate the analysis of ambiguous situations. The algorithms at the base of this progress are artificial neural networks, which take inspiration from the plasticity and non-linearity of biological neural networks. They originate in the nineteen fifties, when the first algorithm allowing a machine to learn abstract representations was developed (Rosenblatt 1958). Artificial neural networks are now becoming popular, because the computation capabilities of microprocessors have improved enough to run these complex algorithms (LeCun et al. 2015) to solve useful tasks. Tasks such as image recognition require computing the response of millions of formal neurons and tuning tens of millions of parameters. Even though these algorithms are still far from the complexity of the human brain, which has one hundred billion neurons, running them on classical computer architectures already costs significantly more energy than appropriate for the range of possible applications.

A. Fukushima
e-mail: akio.fukushima@aist.go.jp

H. Kubota
e-mail: hit-kubota@aist.go.jp

S. Yuasa
e-mail: yuasa-s@aist.go.jp

G. Khalsa
Department of Material Science and Engineering, Cornell University, Ithaca, NY 14853, USA
e-mail: guru.khalsa@cornell.edu

M. D. Stiles
National Institute of Standards and Technology, Center for Nanoscale Science and Technology, Gaithersburg, MD 20899-6202, USA
e-mail: mark.stiles@nist.gov

D. Querlioz
Centre de Nanosciences et de Nanotechnologies, CNRS, Université Paris-Sud, Université Paris-Saclay, 91405 Orsay, France
e-mail: damien.querlioz@u-psud.fr

Modern computers are based on the Von Neuman architecture where the processing unit is separated from the memory and data has to move sequentially back and forth between these two units through a shared bus. While this architecture has been responsible for the dramatic growth in computational capabilities, it is not well adapted for energy efficient cognitive computing. While artificial neural networks can perform comparably to humans for some particular tasks, they do so consuming three to four orders of magnitude more energy than the human brain. The energy efficiency of the brain comes in part from its entanglement of processing and memory, with biological neurons that process the information densely interconnected by synapses that hold the memory. This observation motivates building brain-inspired chips with analog components whose physics mimics the behavior of neurons. Biological neurons encode their information in the spikes they emit. One branch of neuroscience models neurons as non-linear auto-oscillators and the brain as a large assembly of interconnected oscillators that compute through their complex dynamics. A brain-inspired chip based on these principles requires integrating millions of non-linear oscillators in an area as small as one square centimeter. Simple arithmetic shows that this scale requires nanoscale non-linear oscillators.

While there are proposals for neurons based on memristors and Josephson junctions, these have not been experimentally demonstrated yet. Here we describe an experimental realization of nanoscale neurons based on spin-torque oscillators (Kiselev et al. 2003; Rippard et al. 2004) in the approach of reservoir computing. Reservoir computing offers a convenient platform to test the potential of spin-torque oscillators since a single oscillator excited by time-multiplexed inputs can generate a transient signal equivalent to the response of a whole neural network. This approach greatly simplifies the experimental setup and has been successfully used, notably in optics, to perform complex cognitive tasks such as speech recognition.

In this chapter, we summarize our work on reservoir computing using spin-torque oscillators (Riou et al. 2017; Torrejon et al. 2017). Here, we emphasize the measurement system, the results of pattern recognition tasks, and the optimization of the operating regime. We only touch lightly on the single-node reservoir computing approach and the physics of the spin-torque oscillators. Interested readers can find more details in our original papers (Riou et al. 2017; Torrejon et al. 2017) or the other chapters in this volume. Section 2 presents our experimental implementation of single-node reservoir computing based on a single vortex spin-torque nano-oscillator. Then, Sect. 3 presents the classification results obtained using this approach (Torrejon et al. 2017) for two tasks, sine versus square classification (Sect. 3.1.1) and spoken digit recognition (Sect. 3.2.2). Section 4 describes the optimization of the different parameters that can be tuned to improve the recognition rate on the sine and square classification task.

2 Hardware Implementation

2.1 Measurement Set-Up

The samples used for this demonstration are magnetic tunnel junctions (MTJs) with a vortex magnetization in the free layer. The TMR of the junction is about 135% for a resistance of 42Ω. For the dimensions used here (thickness $L = 6$ nm and diameter $\Phi = 375$ nm), the FeB layer has a remanent vortex magnetization. Under DC current injection, the core of the vortex steadily gyrates around the center of the dot with a frequency in the range 250–400 MHz. Vortex dynamics driven by spin-torque are well understood (Grimaldi et al. 2014), well controlled, and have been shown to be particularly stable (Tsunegi et al. 2014), and in this case have an excellent signal-to-noise ratio (for more details see the chapter by Taniguchi et al. 2016).

The measurement setup is shown in Fig. 1a. The experimental preprocessed input signal V_{in} is generated by a high-frequency arbitrary-waveform generator and injected as a current through the magnetic nano-oscillator. The preprocessed input varies with

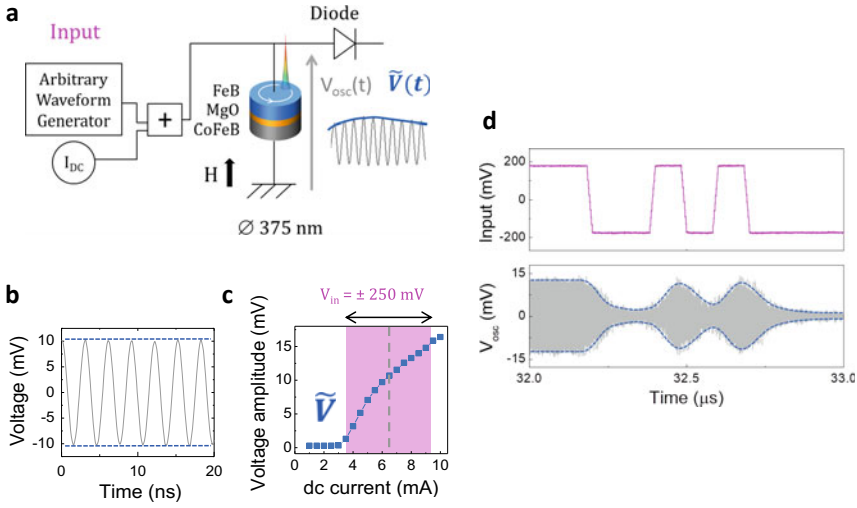


Fig. 1 **a** The measurement setup: A magnetic field delivered by an electromagnet (not shown) and DC source fix the operating point of the oscillator. The signal to analyze is sent by an arbitrary-waveform generator. The magnetic tunnel junction emits an oscillating voltage. A diode allows extraction of the oscillation amplitude. **b** Oscillating response in gray and oscillation amplitude figured in blue. **c** Non-linear variation of the oscillation amplitude $\tilde{V}$ as a function of the input current I_{DC} at $\mu_0 H = 430$ mT. The purple shaded area highlights the typical excursion in the voltage amplitude that results when an input signal of $V_{\text{in}} = \pm 250$ mV is injected. Here $I_{DC} = 6.5$ mA (vertical doted line). **d** Upper graph: input signal sent by the arbitrary-waveform generator (magenta). Lower graph: transient response of the oscillator. The emitted oscillating voltage is plotted in gray. The amplitude of the oscillation is figured by dashed blue lines. Adapted from Torrejon et al. (2017)

a time step θ. The sampling rate of the source is set to 200 MHz, which corresponds to 20 points per interval of discretized time (θ) in the reservoir computing scheme for the spoken-digit recognition task and 500 MHz (50 points per interval) for the classification of sines and squares. The peak-to-peak voltage variation in the input signal is 500 mV, which corresponds to peak-to-peak current variations of 6 mA (part of the incoming signal is reflected due to impedance mismatch between the sample and the circuit). The bias conditions of the oscillator are set by a DC current source and an electromagnet that applies a field perpendicular to the plane of the magnetic layers. These bias conditions determine the operating point of the oscillator. The oscillating voltage emitted by the nano-oscillator is rectified by a planar tunnel microwave diode, with a bandwidth of 0.1–12.4 GHz and a response time of 5 ns. For the range of power and frequency used here, the response of the diode can be considered as linear. The input dynamic range of the diode is between 1 μW and 3.15 mW, corresponding to a DC output level of 0–400 mV. We use an amplifier to adjust the emitted power of the nano-oscillator to the working range of the diode. The output signal is then recorded by a real-time oscilloscope.

2.2 *Physical Properties of the Oscillator Used for Computation*

The two main properties of the spin-torque nano-oscillator used for computation are the non-linearity of the oscillation amplitude with the input DC current and the relaxation of the oscillation amplitude. These two properties are necessary to ensure the separation property and the fading memory needed for reservoir computing (Appeltant et al. 2011).

2.2.1 Non-linearity of the Oscillations Amplitude

A reservoir realizes a non-linear transformation of an input signal. This non-linearity allows the reservoir to project the initial problem into a different space in which classes that were not linearly separable in the original space become linearly separable. The variable used for computation is the amplitude of the oscillations (Fig. 1b). The corresponding non-linearity is thus the non-linearity of the amplitude level as a function of the DC current. Figure 1c shows this non-linearity for a magnetic field of 400 mT. Below a threshold current (which is around 3 mA for this sample at this magnetic field), the oscillator does not emit any signal because the spin-torque is not sufficient to compensate the damping and thus the spin polarized current does not move the vortex core. Above this threshold current, the vortex core gyrates. The amplitude of the oscillation is proportional to the radius of the vortex orbit s. The non-linearity of the amplitude evolves approximately like $\lambda(I, H_\perp)\sqrt{I - I_{\text{th}}}$ (Grimaldi et al. 2014).

Figure 1c illustrates the effect of the bias current on the oscillation amplitude response $\widetilde{V}$ to an AC current. As mentioned in Sect. 2.2.1, the input signal delivered by the arbitrary-waveform generator has an amplitude of 500 mV peak to peak which corresponds to a 6 mA variation of injected current. Thus the input induces variations on a limited part of the non-linear amplitude curve. This area is represented in magenta in Fig. 1c for a DC current of 6 mA. By changing the DC current, one would move the pink area (which is centered on the bias DC current value) and thus the typical excursion in the voltage amplitude which is explored. For computation, it is not necessary that this excursion includes the threshold current (unlike with the classical rectified linear activation function). Indeed the square root-like behavior of the amplitude provides the non-linearity for separating the inputs in the framework of reservoir computing.

The influence of the magnetic field is also crucial because it changes the shape of the non-linear function itself. It changes the threshold current (for smaller fields the threshold current is smaller) and the amplitude of the non-linearity. The optimum choices for DC currents and magnetic fields are discussed in Sect. 4.

2.2.2 Relaxation of the Oscillation Amplitude

The second essential property for single-node reservoir computing is a form of memory. This property is important both for ensuring the connectivity between temporal neurons and to ensure a fading memory. This demonstration only uses the intrinsic memory due to the relaxation of the oscillator's amplitude. When the excitation of the vortex core changes suddenly, the amplitude of its orbit changes more slowly. Thus, the orbit radius and the proportional amplitude also change slowly. Figure 1d shows the variation of the oscillation amplitude when the oscillator is subjected to a varying input signal. The response of the oscillator is plotted in gray and the amplitude of the oscillation is highlighted in blue. Transitions in the input signal are much more abrupt than in the oscillation amplitude signal. The characteristic time of these changes is the relaxation time of the oscillator which is roughly inversely proportional to the frequency and the damping factor ($T_{\text{relax}} \approx \frac{1}{\alpha f}$ Slavin and Tiberkevich 2009). For our sample, the relaxation time is around 200 ns, except when the current is close to the threshold. For the later regime, the relaxation time is larger but the oscillation amplitude is low and very noisy, making it difficult to exploit this regime for computation.

The relaxation is important to determine the discretization time θ which is used to define the state of one temporal neuron in a ring of similar neurons. In order to have connections between the temporal neurons, we should choose $\theta < T_{\text{relax}}$. For the work described in this chapter, $\theta = 100$ ns. This choice is discussed in Sect. 4.

3 Results on Classification Tasks

3.1 Results on Sine/Square Recognition Task

3.1.1 Task

The sine and square waveform classification task has been used in other studies (Paquot et al. 2012) to evaluate the performance of reservoir computing based on an oscillator with delayed feedback. The goal is to classify each point of the input as part of a sine by returning an output 0, or as part of a square by returning an output 1. Each period of sine and square is discretized into eight points giving 16 different cases to classify. The input $u(k)$ is composed of 1280 points (160 randomly arranged periods of sine or square). The first half of the points are used for training (to find optimum output weights W) and the second half for testing.

The different inputs to classify are shown as red dots in Fig. 2a. The inputs take five different values in a sine period and two different values in a square input. To return the same output value for five different input values of a sine (or 2 in the case of the square), the reservoir must be non-linear. In addition, in the sine period the 3rd and the 7th point have a value $+1$ and -1 that corresponds to the values taken by the input in the square. So in the absence of memory, when the input value is $+1$ or -1 it is impossible to know whether these points belong to a sine or to a square. Therefore this temporal pattern recognition task is not trivial because it needs both the non-linearity and the memory of a neural network.

3.1.2 Protocol

Different steps of the protocol are represented in Fig. 2a–c. Figure 2 shows how the oscillator is driven by the input signal. The input $u(k)$ (Fig. 2a) is a sequence of discretized sine and square periods. This discrete input is then preprocessed. This preprocessed input is a time-multiplexed version of the inputs that different neurons should receive in a standard spatial neural network. Here the neural inputs are the value of $u(k)$ multiplied by a coefficient ± 1. This allows the oscillator response to be a time-multiplexed version of a spatial neural-network response. Later we will refer to these equivalent neurons as temporal neurons. If the preprocessed input varies faster than the relaxation time of the oscillator, it creates connections between the temporal neurons because the saturation is never reached so the oscillator state always depends on its previous state. The preprocessed input $J(t)$ for a sine and square period is shown in Fig. 2b. An input with only 12 temporal neurons is chosen here. The reservoir emulates 12 temporal neurons with a mask containing only binary values $+1$ and -1. The number of neurons is smaller than the optimum number for better visualization in the figure. The results in Sect. 3.1.3 are based on a 24 temporal neurons reservoir. The time allocated to each neuron, θ, is taken here to be 100 ns, which we found to be optimal (see Sect. 4). When the oscillator receives

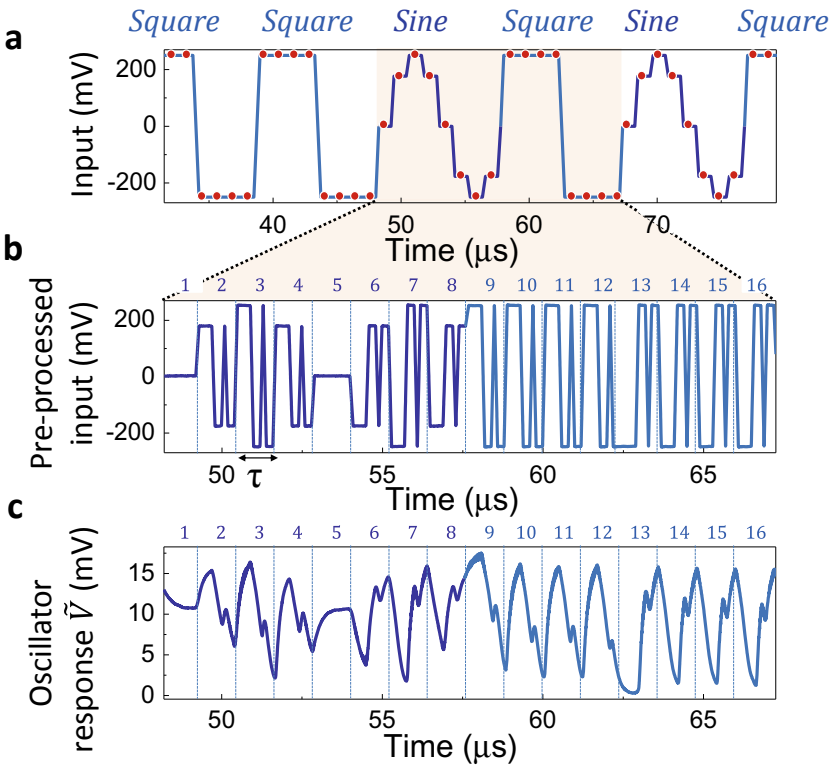


Fig. 2 Sine/square classification process: **a** The signal to classify composed of periods of sine and square with the points to identify as belonging to a sine or to a square in red. **b** The preprocessed input signal for a sine (dark blue) and square (light blue) period for a mask with 12 values used. **c** Oscillation amplitude transient response to the oscillator signal. Figure extracted from Riou et al. (2017)

this preprocessed input, it emits a transient response. The amplitude of the emitted oscillation measured experimentally is plotted in Fig. 2c.

Figure 3a shows how to retrieve the mapping of the preprocessed input by the single oscillator. This step is done offline on a computer. Discrete points are sampled at every time step θ. The values of these points correspond to the response of the temporal neurons. Measuring these values gives the reservoir state. Sampling the temporal traces of the oscillation amplitude properly requires aligning them with the preprocessed input (misalignment can result in bad classification). The rest of the process is a standard reservoir computing procedure. The neuron responses are linearly combined to reconstruct the output. This step is shown in Fig. 3b. The coefficients of the linear combination are obtained by linear regression method. This approach using a single oscillator with time multiplexing emulates the response of a ring shape recurrent neural network. This architecture is represented in Fig. 3c.

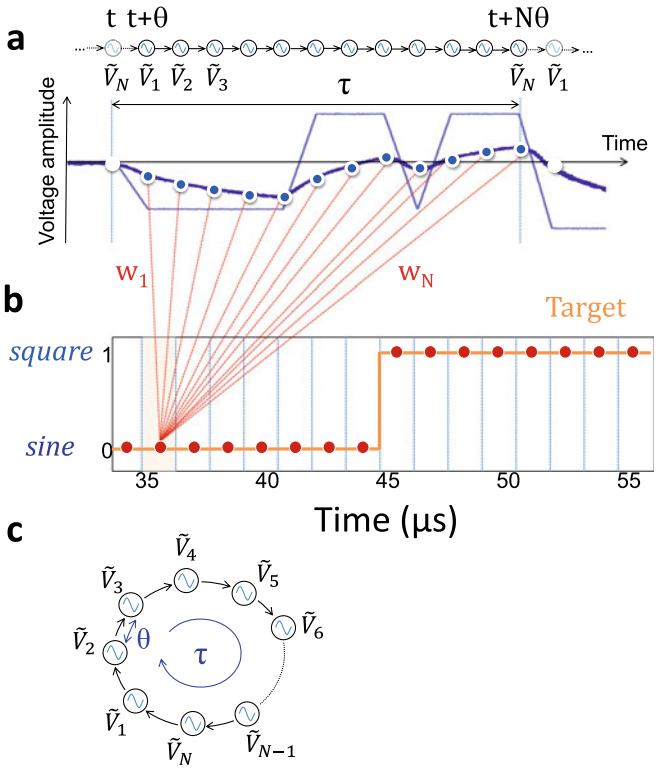


Fig. 3 Mapping and reconstruction of the output: **a** The oscillator voltage amplitude $\widetilde{V}$ during a single time segment $\tau = N\theta$. Here, $N = 12$ neurons (12 samples $\widetilde{V}_i$ separated by the time step θ) are used to construct the output. **b** Target for the output, $\sum_{i=1}^{N} w_i \widetilde{V}_i$, reconstructed from the output voltages $\widetilde{V}_i$ and the weights w_i in each time segment τ. **c** The transient states of the oscillator give rise to a chain reaction emulating a neural network with a ring structure. Figure extracted from Riou et al. (2017)

Response of this equivalent neural network is used to classify the sine and square inputs.

3.1.3 Set Point Dependent Results

Figure 4a shows the best reconstructed output obtained by experimentally emulating a 24-neuron network. The root mean square (RMS) deviation between target and output is 11%, which is small enough to distinguish between sines and squares without any error (perfect classification) for the chosen choice of parameters:DC current $I_{DC} =$ 7.2 mA, magnetic field $\mu_0 H = 447$ mT, input amplitude $V_{\text{in}} = 500$ mV (equivalent to 6 mA peak to peak). Figure 4a shows that if we trace a threshold line (in blue) at 0.5, all the outputs for square inputs are over this threshold and all the points for sine

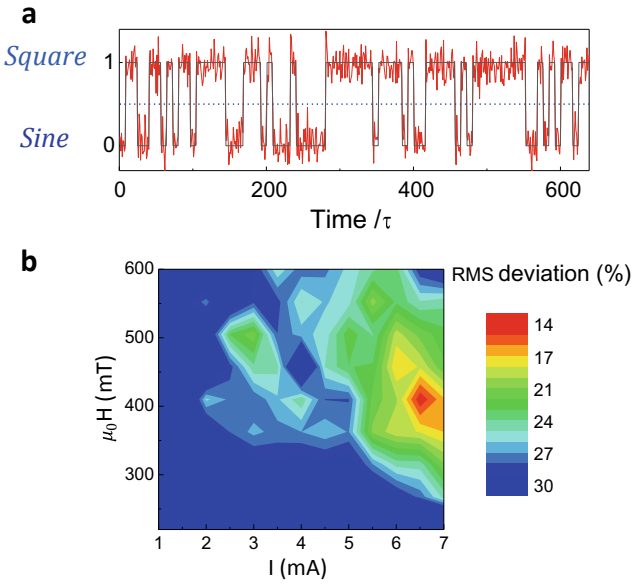


Fig. 4 **a** Reconstructed output (red) and target (gray) in response to an input waveform with 80 randomly arranged sines and squares. The magnetic field is 447 mT, and the applied current 7.2 mA. The results are based on 24 neurons separated by $\theta = 100$ ns. **b** Root mean square deviation of output-to-target deviations: map as a function of DC current I and magnetic field H

inputs are under this threshold. This perfect classification is achieved if the RMS deviation between the target and the output is small enough, here, close to 10%.

Perfect classification is obtained for operating points (the values of the bias DC current and the bias magnetic field) that give a small enough RMS deviation between the target and the output. As seen in Fig. 4b, the RMS deviation varies from 10% to more than 30% depending on the bias conditions. We interpret the optimal operating point conditions in Sect. 4. After identifying this region of magnetic field and DC current leading to high performance of the oscillator for sine/square classification, we discuss the more complex task of spoken-digit recognition.

3.2 Results on Spoken-Digit Recognition

3.2.1 Task

Spoken-digit recognition is a widely used benchmark task in the hardware reservoir computing community (Brunner et al. 2013; Dejonckheere et al. 2014; Larger et al. 2012; Paquot et al. 2012; Vinckier et al. 2015). The goal of the task is to recognize digits from audio waveforms produced by different speakers. For this task, the inputs

are taken from the NIST TI-46 data corpus. The input consists of isolated english spoken digits said by five different female speakers. Each speaker pronounces each digit ten times. The set of ten digits said by all of the speakers is called an "utterance". The 500 audio waveforms are sampled at a rate of 12.5 kHz and have variable time lengths. Only female speakers are chosen to benchmark with the literature. Adding male and children increases the complexity of the task because it increases the variety of voice tones and thus increases the dispersion of the data to classify.

3.2.2 Protocol

The recognition of audio waveforms requires an additional time-domain to time-frequency-domain transformation, before inputting the signal into the reservoir. For this purpose we used two different methods: a spectrogram and a cochlear model. Both methods divide each word into several time intervals N_τ. Each of these time sequences has a fixed length τ and undergoes a frequency analysis through either Fourier transform (spectrogram model; 65 channels, $N_\tau \in 24, \ldots, 67$) or a more complicated non-linear approach (cochlear model; 78 channels, $N_\tau \in 14, \ldots, 41$), which uses several different notch filters followed by non-linear automatic gain controllers (Lyon 1982).

After the transformation from time-domain to time-frequency domain, each word is represented by a matrix with $N_f = 65$ or $N_f = 78$ rows representing the frequency channels and N_τ columns representing the time (Fig. 5b). These inputs are then preprocessed being multiplied by a $N_f \times N_\theta$ matrix (where N_θ is the number of neurons in the reservoir) called a mask, containing binary values. The resulting input for the oscillator is $N_\theta \times N_\tau$ matrix. Here we are emulating $N_\theta = 400$ temporal neurons, each of which is connected to all of the frequency channels for each time interval with the binary input weights defined by the mask.

Each preprocessed input value is consecutively applied to the oscillator as a constant current for a time interval of $\theta \approx 100$ ns (Fig. 5c). This time is short enough to guarantee that the oscillator is maintained in its transient regime so the emulated neurons are connected to each other, but is long enough to let the oscillator respond to the input excitation. The amplitude of the AC voltage across the oscillator is recorded for offline post-processing (Fig. 5d).

As for the previous task, the post-processing is then separated in two different phases: training and testing. During the training phase, a part of the spoken digits is used to determine the optimal sets of output weights $w_{i,\theta}$, where i indexes the desired digit. The recorded traces are multiplied by the output weights $w_{i,\theta}$ and then averaged over the N_τ time steps. It results in 10 output values y_i, which should ideally be equal to the target values $y_i = 1.0$ for the appropriate digit and 0.0 for the rest. In the training process, a fraction of the utterances are used to train these weights; the rest of the utterances are used in the classification process to test the results. The optimum weights are found by minimizing the difference between $\widetilde{y}_i$ and y_i for all of the words used in the training. The reconstruction of the outputs y_i is a linear combination and thus finding the optimal weights for the training examples

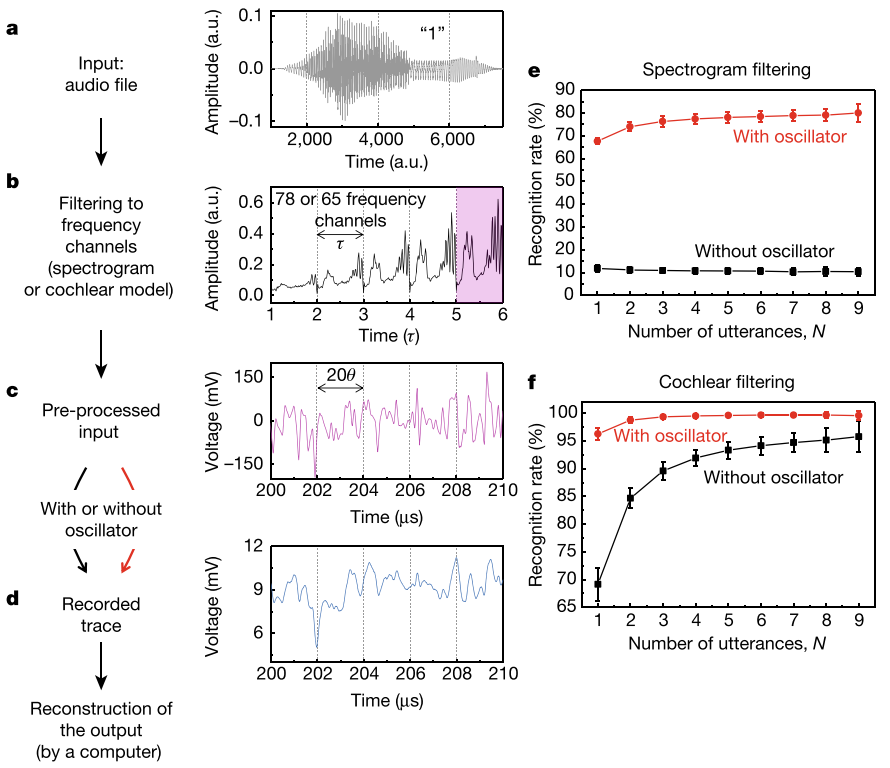


Fig. 5 Protocol for spoken-digit recognition. **a–d** Principle of the experiment. **a** Audio waveform corresponding to the digit 1 pronounced by speaker 1. **b** Filtering to the frequency channels for acoustic feature extraction. The audio waveform is divided in intervals of duration τ. The cochlear model filters each interval into 78 frequency channels (65 for the spectrogram model), which are then concatenated as 78 (65) values for each interval, to form the filtered input. **c** Preprocessed input (transformed from the purple shaded region in b). The filtered input is multiplied by a randomly filled binary matrix (masking process), resulting in 400 points separated by a time step θ of 100 ns in each interval of duration τ ($\tau = 400\theta$). **d** Oscillator output. The envelope $\widetilde{V}(t)$ of the emitted voltage amplitude of the experimental oscillator is shown ($\mu_0 H = 430$ mT, $I = 6$ mA). The 400 values of $\widetilde{V}(t)$ per interval τ ($\widetilde{V}_i$, sampled with a time step θ) emulate 400 neurons. The reconstructed output '1', corresponding to this digit, is obtained by linearly combining the 400 values of $\widetilde{V}_i$, sampled from each interval τ. **e**, **f** Spoken-digit recognition rates in the testing set as a function of the number of utterances N used for training for the spectrogram filtering (a; $H = 430$ mT, $I = 6$ mA) and for the cochlear filtering (b; $\mu_0 H = 448$ mT, $I = 7$ mA). Because there are many ways to pick the N utterances, the recognition rate is an average over all $10!/[(10-N)!N!]$ combinations of N utterances out of the 10 in the dataset. The red curves are the experimental results using the magnetic oscillator. The black curves are control trials, in which the preprocessed inputs are used for reconstructing the output on a computer directly, without going through the experimental setup. The error bars correspond to the standard deviation of the recognition rate, based on training with all possible combinations. Figure extracted from Torrejon et al. (2017)

is a linear regression process. If we consider the target matrix $\widetilde{Y}$, which contains the targets $\widetilde{y_i}$ for all of the time steps τ used for the training, and the response matrix S, which contains all neuron responses for all of the time steps τ used for the training, then the matrix W, which contains the optimal weights, is given by $W = \widetilde{Y}^{\dagger} S$, where the symbol † represents the Moore–Penrose pseudo-inverse (Penrose 1955).

During the testing phase, the weights are fixed and applied to the remaining recorded traces. The ten reconstructed outputs corresponding to one digit are averaged over all of the time steps τ of the signal, and the digit is identified by taking the maximum value of the ten averaged reconstructed outputs. The efficiency of the recognition is evaluated by the word success rate, which is the rate of digits that are correctly identified.

Training is achieved with training sets which can have variable size (number of utterances used for the training) and composition (which combination of utterances is used for the training). Different training sets led to different performances during the testing phase. We trained the system using the ten digits spoken by the five speakers. The only parameter that we changed is the number of utterances used for the training. If we use N utterances for training, then we use the remaining $10 - N$ utterances for testing. However, some utterances are very well pronounced whereas others are hardly distinguishable. As a consequence, the resulting recognition rate depends on which N utterances are picked for training in the set of ten (for example, if $N = 2$, then the utterances picked for training could be the first and second, but also the second and third, or the sixth and tenth, or any other of the $10!/(8!2!)$ combinations of 2 picked out of 10). To avoid this bias, the recognition rates that we present here are the average of the results over all possible combinations. The error bars correspond to the standard deviation of the word recognition rate.

In order to see the contribution of the spin-torque oscillator in the recognition process we compare the results obtained from the oscillator time traces with a control trial. During the control trial, the preprocessed inputs are used for reconstructing the output on a computer directly, without going through the experimental set-up.

3.2.3 Preprocessing Dependant Results

The improvement shown in the experimental results over the control results (see Fig. 5e) indicates that the spin-torque nano-oscillator greatly improves the quality of spoken-digit recognition, despite the added noise that is concomitant to its nanometre-scale size. In this case, the extraction of acoustic features, achieved by Fourier transforming the audio waveform over finite time windows, plays a minimal part in classification. Without the oscillator (black line), the recognition rates are consistent with random choices; with the oscillator (red line), the recognition rate is improved by 70%, reaching values of up to 80%. This example highlights the crucial role of the oscillator in the recognition process.

Using the cochlear filtering (Fig. 5f), which is the standard in reservoir computing and has been optimized on the basis of the behavior of biological ears, we achieve recognition rates of up to 99.6%, as high as the state of the art. Compared to the

control trial, the oscillator reduces the error rate by a factor of up to 15. Our results with a spin-torque nano-oscillator are therefore comparable to the recognition rates obtained with more complicated electronic or macroscopic optical systems (between 95.7 and 99.8% for the same task with cochlear filtering).

3.3 Conclusion

In this section, two tasks were used to evaluate the performance of reservoir computing using the dynamics of a spin-torque nano-oscillator: sine/square classification and spoken-digit recognition. Sine/square classification is a simpler task but allows testing the non-linear behavior and memory of the reservoir which are the key features for good classification on more complex tasks. Using 24 temporal neurons, a systematic study of classification for different magnetic fields and DC current bias conditions was conducted. In the best case a 10% root mean square deviation between the reconstructed output and the target was obtained, which allows perfect classification of sine and square inputs. Best bias conditions were selected to move on to the more complex task of spoken-digit recognition.

Spoken-digit recognition requires frequency filtering of audio files prior to sending the input to the oscillator. Two filtering methods were studied: first a simple linear spectrogram method and a more complex cochlear decomposition to benchmark our result with the existing literature. Using the spectrogram method, the oscillator improves the recognition up to 70% and the overall success rate is 80% by training on 90% of the data. This result stresses the critical role of the nano-oscillator for recognition. Using a cochlear decomposition, 99.6% success rate was reached which is state-of-the-art results for both numerical and hardware methods. Also it is important to note that training a linear classifier directly on a cochleogram enables a success rate of 96%. Thus cochlear decomposition already separates a part of the input. The results in this chapter demonstrate neuromorphic computing performed with a nanoscale “neuron”.

For this demonstration, experimental parameters such as the temporal time scale θ and the operating point (DC current and field) were specially selected to achieve good classification results. The next section elucidates the influence of these different parameters and gives guidelines to choose them appropriately.

4 Optimizing the Experimental Parameters and Data Processing for Improved Classification

4.1 Input Sampling Rate and Amplitude

The memory and connectivity in single oscillator reservoir computing are obtained by driving the oscillator with a quickly varying preprocessed input. The time interval of the input variation is θ. In this part, we discuss how to optimize θ to obtain the best classification performance. Figure 6a–c illustrates the dynamical response of the oscillation envelope for different values of θ and different amplitudes of the input.

The input signal sent to the oscillator by the arbitrary-waveform generator (AWG) takes three different amplitude values (300, 400, and 500 mV peak to peak). Figure 6a–c represents the oscillation amplitude response for, respectively, $\theta = 25$ ns (Fig. 6a), $\theta = 100$ ns (Fig. 6b), and $\theta = 300$ ns (Fig. 6c). For $\theta = 25$ ns, the time rate of the input is much shorter than the intrinsic relaxation time of the oscillator measured around 200 ns ($\theta = T_{\text{relax}}/8$) so the oscillator stays in the transient regime all

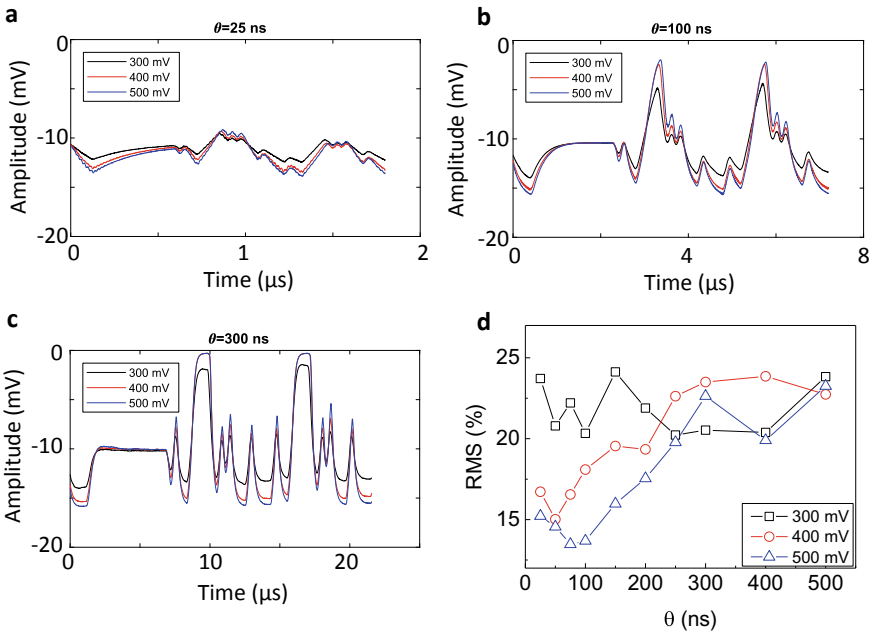


Fig. 6 **a**, **b**, **c**, $\widetilde{V}$ oscillation amplitude variation of the oscillator response for three levels of input 300 mV (black), 400 mV (red) and 500 mV (blue) in the case of time step $\theta = 25$ ns (a), $\theta = 100$ ns **b** and $\theta = 300$ ns (**c**). **d** Classification performance depending on θ: Root mean square of output-to-target deviations as a function of the time step θ (separation between transient states of the oscillator $\widetilde{V}_i$) for different amplitudes of the input signal (300, 400 and 500) mV when the target is in exact phase with the input

the time. The memory effect is in this case strong but the variation of the emitted oscillation amplitude signal is quite small. The oscillator does not have the time to reach higher values. The amplitude of the oscillation also depends on the peak-to-peak input amplitude and for an input amplitude of 300 mV, the signal is particularly small.

At the other extremes for $\theta = 300$ ns, which is greater than the oscillator relaxation time ($\theta = 1.5T_{\text{relax}}$), the measured amplitude response reaches its saturation regime and the oscillator loses the memory of what happened one time step θ in the past. But for such high θ, the amplitude of the signal is much larger because the oscillator reaches its saturation regime.

For an intermediate value $\theta = 100$ ns (approximately $T_{\text{relax}}/2$), the oscillator still remains in a transient regime with a large amplitude. Intuitively, we expect that the trade-off is to choose a value of θ that allows for a significant amplitude (so a decent signal over noise ratio) and still has memory. It is important to note that a higher input amplitude for a fixed θ always improves the signal to noise ratio.

Figure 6d presents the experimental RMS error result for sine and square classification as a function of θ. For 500 ns, increasing θ first decreases the RMS error until an optimum after which the RMS error increase again. This optimum seems to be around 100 ns. In the first part, the error decreases because the signal-to-noise ratio increases but after $T_{\text{relax}}/2$ the error increases again because of a loss of memory. This trend is also seen for an input amplitude of 400 mV. In that case, surprisingly the optimum seems to occur for a smaller θ of 50 ns. However the general result is always worse than for 500 mV input amplitude. Finally for an input amplitude of 300 mV, no clear trend is observed and the classification results are bad. For this particular voltage amplitude input, there does not seem to be any values of θ that have both sufficient memory and a sufficient signal-to-noise ratio.

4.2 Magnetic Field and DC Current Dependence

4.2.1 Noise and Amplitude Ratio

Successful classification depends strongly on the operating point (Fig. 7a). The magnetic field and the applied bias DC current determine the amplitude variation and the noise level of the oscillator signal. The amplitude of the measured signal and its asymmetry are measured through amplitude variations in both positive and negative directions, V_{up} and V_{dw} (Fig. 7b upper graph), while the noise in the voltage amplitude is measured through the noise standard deviation ΔV (Fig. 7b lower graph). We extract these parameters from the time traces of the voltage emitted from the oscillator at each bias point, and plot $V_{\text{up}} V_{dw}$ (Fig. 7c) and $1/\Delta V$ (Fig. 7d) as a function of the DC current I and field H.

Large oscillation amplitudes (red regions in Fig. 7c) are obtained when the applied magnetic field is low, in such case the magnetization is weakly confined and when the applied DC current is high because it increases the spin torque on the magnetization.

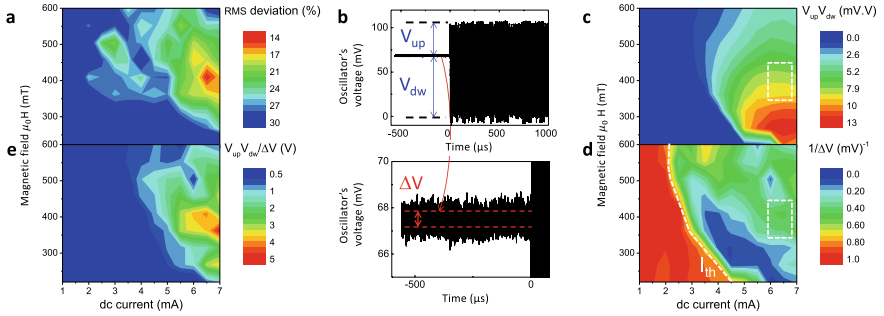


Fig. 7 **a** Root mean square deviation of output-to-target deviations: map as a function of DC current I and magnetic field H. **b** Amplitude variations in both positive and negative directions, V_{up} and V_{dw} (upper graph) and standard deviation of the noise in the voltage amplitude ΔV (lower graph). **c** Maximal reponse of the oscillator to the input ($V_{\text{up}}V_{dw}$) in the H-I plane. **d** Inverse of the noise level ($1/\Delta V$) in the H-I plane. The threshold current I_{th} is indicated in white dashed line. **e** Map of the ratio maximal amplitude to noise $V_{\text{up}}V_{dw}/\Delta V$ in the H-I plane, showing that these parameters largely determine the performance of the oscillator (compare with **a**). Extracted from Torrejon et al. (2017)

High noise regions (blue regions in Fig. 7d) are obtained just above the threshold current for oscillation I_{th}. In such conditions, the oscillation amplitude increases rapidly as a function of the current and is sensitive to external fluctuations. The dotted white boxes in Fig. 7c, d (currents of 6–7 mA and magnetic fields of 350–450 mT) highlight regions of high oscillation amplitude and low noise. Such bias conditions give root mean square deviations below 15%, and there are no classification errors between sine and square waveforms. The similarity between the map of $V_{\text{up}}V_{dw}/\Delta V$ (Fig. 7e) and the map of root mean square error (Fig. 7a) confirms that the best conditions for the recognition are regimes with both high oscillation amplitude and low noise. We expect that the requirement, which we demonstrate here for a spin-torque oscillator, of a high signal-to-noise ratio to obtain good classification would apply to any type of nanoscale oscillator used for neuromorphic computing.

4.2.2 Amplitude Level

Modifying the operating point modifies the oscillation amplitudes level as can be observed in Fig. 8a. In the case of our sample, the highest amplitude is obtained for fields between 300 and 500 mT and for DC currents between 7 and 9 mA. During the reservoir computing experiments, we fix the magnetic field and we vary the current that the sample receives.

First the magnetic field should be selected so the input signal induces large and reproducible amplitude variation. By changing the magnetic field, the threshold current and the derivative of $\widetilde{V}$ with the current are modified (Fig. 8b). $\widetilde{V}$ is directly linked to the vortex orbit s. The oscillation amplitude is given theoretically by the

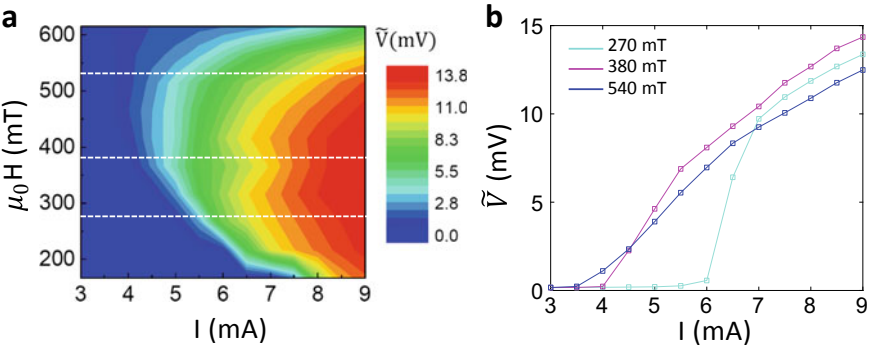


Fig. 8 **a** Voltage amplitude $\widetilde{V}$ of the oscillator in the steady state: map in the I-H plane. **b** Voltage amplitude $\widetilde{V}$ of the oscillator in the steady state as a function of I for $\mu_0 H = 273$ mT (cyan), $\mu_0 H = 379$ mT (magenta) and $\mu_0 H = 537$ mT (blue)

following equation

$$\widetilde{V}(t) = \lambda(H_\perp, I)s(t), \tag{1}$$

where s is the vortex radius and λ is a factor depending on the perpendicular magnetic field $H_\perp$ and the DC current. s is a function of I/I_{th} and evolves approximately as $\sqrt{((I/I_{\text{th}}) - 1)}$ (Grimaldi et al. 2014). Because of this dependency, for low magnetic field such that 297 mT, the threshold current I_{th} is high (5.9 mA) and $\widetilde{V}$ varies very rapidly with the input current I so the effect of an input current variation maybe not reproducible, particularly in the proximity of I_{th}. For higher magnetic field such as 537 mT, I_{th} is lower (4.0 mA) but the variation of $\widetilde{V}$ with I is slow, such as the input current variation results in small variation of $\widetilde{V}$. Finally, optimal variations are obtained for intermediate values of field such as 379 mT, where the variations of input current induce significant change of $\widetilde{V}$ but the value of $\widetilde{V}$ is not too sensitive to noise.

Once the magnetic field is fixed, the DC current should be selected to obtained large variations of $\widetilde{V}$ and ensures that $\widetilde{V}$ does not evolve in a linear regime. The DC current influences both the amplitude and the asymmetry of the measured time traces as seen in Fig. 9 for three different DC currents, 4.5, 6.5, and 9.0 mA. The left part of the figure shows the non-linear dependence of the oscillation amplitude on the current. The parts of this non-linear dependence explored for the chosen inputs are represented in the colored areas. An input of 300 mV induces a typical variation of DC current of roughly 4 mA. When the input is sent, the oscillator receives a current in a range ± 2 mA around the fixed DC current.

Changing the DC current explores different windows of the non-linear dependence of $\widetilde{V}$ on I, which is set by the magnetic field. This exploration window is represented by the vertical dashed lines. The central line is the DC current and the left and right lines are the extreme current values received by the oscillator. The level of the oscillation amplitude when no input signal is sent is the intersection of the central

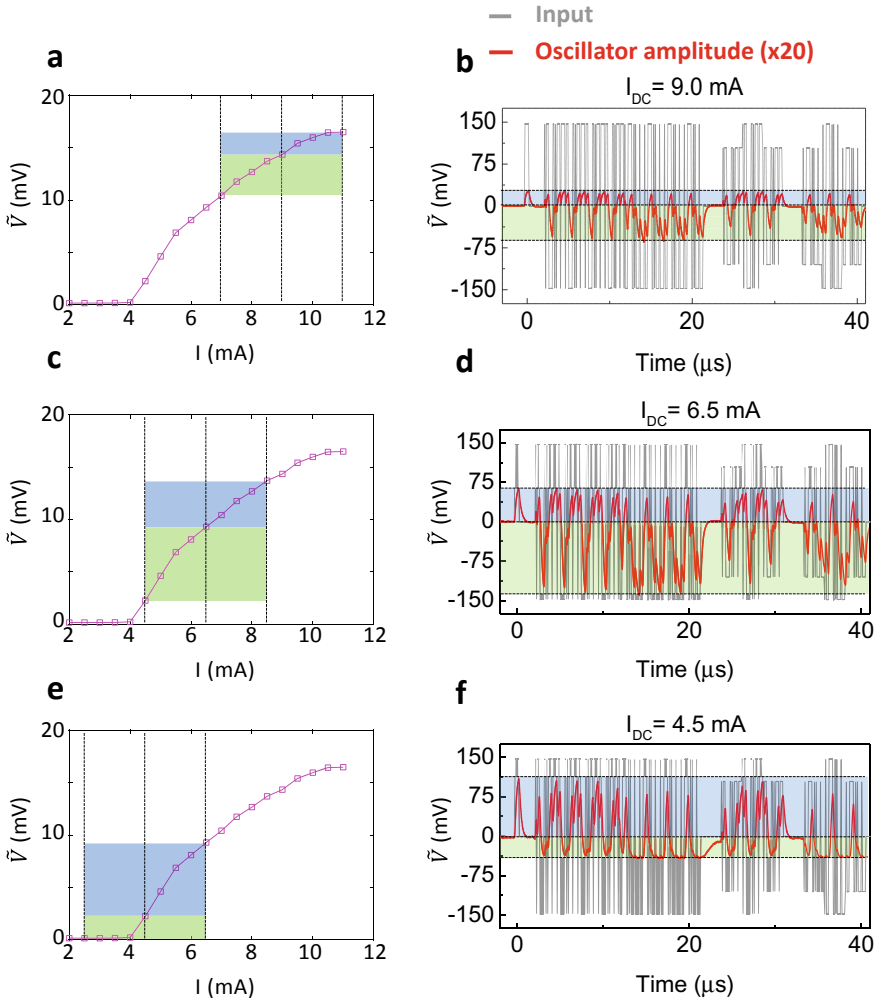


Fig. 9 Influence of the DC current on the oscillation amplitude variation: **a**, **c**, **e** $\widetilde{V}$ as a function of I. The blue (green) shaded area highlights the typical excursion in the voltage amplitude V_{up} (V_{dw}) that results when an input signal of $V_{\mathrm{in}} = +150$ V ($V_{\mathrm{in}} = -150$ mV) is injected. **b**, **d**, **f** The oscillator voltage amplitude, curves (in red) in response to the input waveform (in gray). Blue (green) shaded area represents V_{up} (V_{dw}). Curves are plotted for $\mu_0 H = 379$ mT and $I = 9.0$ mA (**a**, **b**), $I = 6.5$ mA (**c**, **d**), and $I = 4.5$ mA (**e**, **f**)

vertical line and the magenta curve $\widetilde{V} = f(I)$. The higher oscillation amplitude value reached is the intersection of the right vertical line and the curve $\widetilde{V} = f(I)$. V_{up}, the difference between the highest value reached by the oscillator amplitude when input is sent and the oscillation amplitude level due to the DC current, is represented in blue. Similarly the lowest amplitude reached when the input is sent is situated at the intersection between the left vertical line and the $\widetilde{V} = f(I)$ curve. V_{dw}, the difference between the oscillation amplitude due to the DC current and the lowest value reached by the oscillator amplitude when input is sent, is represented in green.

The right part of Fig. 9 shows the time traces corresponding to these different DC current conditions. V_{up} and V_{dw} are shown for the time traces with the blue and green areas. The apparent asymmetry of the measured time traces and the amplitude variation is completely determined by the portion of the non-linearity $\widetilde{V} = f(I)$ that is explored. For 9.0 mA DC current value, the signal is small because the oscillation amplitude evolves in a part of the non-linearity $\widetilde{V} = f(I)$ where variations are small (for high current the growth of $\widetilde{V}$ with I slows down). For 6.5 mA, the lowest current received by the oscillator is close but still greater than the threshold current (the oscillator receives currents between approximately 4.5 and 8.5 mA).

The region close to the threshold current is the region with the largest oscillation amplitude variations. The overall amplitude of the time traces for 6.5 mA is much larger than for 9.0 mA. Because the variation of the oscillation amplitude is stronger close to the threshold current, V_{dw} (green) is much larger than the V_{up} (blue). Decreasing the DC current to 4.5 mA gives saturation because the lowest DC current into the oscillator is then 2.5 mA, which is under the threshold current of 4 mA. Saturation, which maps a whole range of inputs to the same output, is detrimental for computation because these inputs can not be differentiated in later computation. In such cases the asymmetry of the time traces is reversed with V_{up} which is larger than V_{dw}.

An optimal oscillation amplitude variation requires an intermediate magnetic field (typically between 300 and 500 mT) so that the variation of the $\widetilde{V} = f(I)$ non-linearity is not so abrupt such as for low field (because of the high I_{th} value) nor so slow as for high fields. Once the magnetic field is determined, the DC current should be chosen so that the lowest current sent to the oscillator from the AWG is close to the threshold current, giving the largest variations of the oscillation amplitude.

4.2.3 Noise and Non-linearity Trade-Off

Good classification requires non-linearity in the reservoir. The non-linearity allows for separation of the different inputs by projecting the problem is into a different space in the reservoir state. A large signal-to-noise ratio is also necessary, because otherwise similar inputs can be mapped onto very different reservoir states because of the noise so that the approximation property fails for the reservoir. Figure 10 shows that the non-linearity and the voltage noise vary considerably with DC current and magnetic field, suggesting that care is essential in selecting the working conditions.

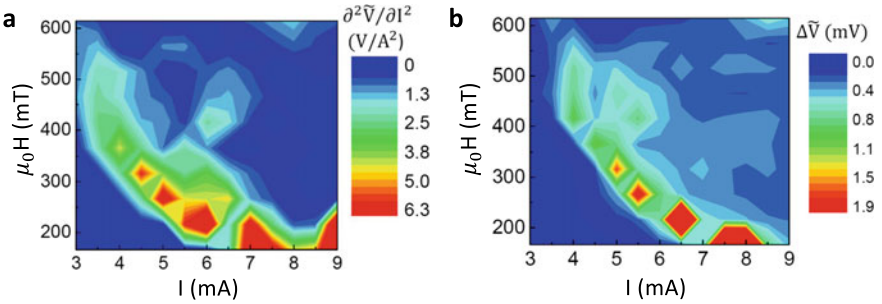


Fig. 10 **a** The non-linearity $\partial^2\widetilde{V}/\partial I^2$ of the oscillator: map in the I-H plane. **b** Amplitude noise $\Delta\widetilde{V}$ of the oscillator in the steady state: map in the I-H plane

To quantify the non-linearity, we computed the second derivative of the voltage with the current $\partial^2\widetilde{V}/\partial I^2$. Figure 10a shows that the largest non-linearities are close to the threshold current where the emission of the oscillator varies strongly. The noise level is evaluated by taking the standard deviation of the fluctuations of the voltage amplitude due to the noise (similarly to the noise evaluated in Sect. 3 Fig. 4). Again, the noise also varies strongly close to the threshold current (Fig. 10b).

Since spin-torque oscillators have a small magnetic volume, thermal noise affects the magnetization dynamics. The resulting voltage amplitude noise is large for large non-linearity, which quantifies the sensitivity of the system to perturbations. The correlation between voltage noise and non-linearity appears clearly in the comparison of Fig. 10a and b. Neuron non-linearity is a key ingredient for classification as it allows the separation of input data. On the other hand, noise in the response of neurons is detrimental for classification as it directly affects the output. Figure 7a shows the classification performance as a function of DC current and magnetic field. We find good performance by choosing a bias point with intermediate non-linearity and therefore intermediate noise, and where the neuron output changes strongly in response to the AC input. Such bias points allow enough non-linearity to classify while keeping a large enough signal-to-noise ratio to distinguish between outputs.

4.3 Conclusion

This section describes the experimental optimization of the θ time scale and the operating point for reservoir computing with a spin-torque oscillator. The θ time step plays both an important role in the memory of the reservoir (particularly when intrinsic memory is the only memory mechanism) and in the connectivity between the virtual neurons. A trade-off leads to a value of θ short enough to assure memory and connectivity and long enough to assure a satisfactory amplitude variation. We find the best trade-off for $\theta = T_{\text{relax}}/2$ where T_{relax} is the relaxation time of the oscillator.

The operating point plays an important role in the optimization of these properties. The magnetic field should be between 300 and 500 mT so the variation of the non-linearity $\widetilde{V} = f(I)$ is not too abrupt nor too smooth. At a given field, the DC current should be large enough to avoid reaching the threshold current and low enough to avoid a regime where $\widetilde{V}$ variation saturates. Finding the correct operating point is a subtle balance, requiring both intermediate non-linearity and noise, while avoiding the threshold current.

5 General Conclusion

In this chapter we demonstrate neuromorphic computing with a nanoscale hardware neuron. We use the non-linear transient dynamics of a single nanoscale spin-torque oscillator to emulate the response of an entire neural network, using the reservoir computing approach. The main physical properties used for this demonstration are the non-linear dependence of the emitted voltage amplitude on the applied current and the memory of the oscillator through the relaxation of the oscillation amplitude.

Using this non-linear dynamics, we classified sine and square waveforms, which requires both memory and non-linearity and we recognized successful digits spoken by different speakers. For this last task we obtained a recognition rate of 99.6% using cochlear decomposition prior to inputting the signal into the oscillator. With a spectrogram as time-to-frequency transformation, the final recognition rate is smaller (80%) but the oscillator provides a higher gain.

The final recognition rate depends on the DC current and the magnetic field that is applied to the oscillator, because these bias conditions modify the regime where the oscillator operates determining both the signal-to-noise ratio and the non-linearity of the oscillator response. The noise and the non-linearity are correlated for spin-torque oscillators and optimal results are obtained for intermediate level of noise and non-linearity. The rate and the amplitude of the input which is sent to the oscillator also play an important role. The amplitude of the input influences the amplitude of the signal emitted by the oscillator and better results are obtained for larger input signals. The rate should ensure that the oscillator stays in a transient regime while keeping large amplitude variations of the emitted signal. Optimal variation time for the input is found for half of the oscillator relaxation time.

Using the reservoir computing approach, we show that spin-torque oscillators are stable enough and have both enough non-linearity and enough memory to perform neuromorphic computing. In this work we used a time-multiplexing approach for computation, allowing for the use of a single oscillator. This work combined with the ability of these oscillators to couple together opens the path to build large-scale hardware neural networks with dense arrays of interconnected spin-torque oscillators.

Acknowledgements This work was supported by the European Research Council ERC under Grant bioSPINspired 682955 and the French ministry of defense (DGA).

References

L. Appeltant, M.C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C.R. Mirasso, I. Fischer, Information processing using a single dynamical node as complex system. Nat. Commun. **2**, 468 (2011)

D. Brunner, M.C. Soriano, C.R. Mirasso, I. Fischer, Parallel photonic information processing at gigabyte per second data rates using transient states. Nat. Commun. **4**, 1364 (2013)

A. Dejonckheere, F. Duport, A. Smerieri, L. Fang, J.-L. Oudar, M. Haelterman, S. Massar, All-optical reservoir computer based on saturation of absorption. Opt. Express **22**(9), 10868–10881 (2014)

E. Grimaldi, A. Dussaux, P. Bortolotti, J. Grollier, G. Pillet, A. Fukushima, H. Kubota, K. Yakushiji, S. Yuasa, V. Cros, Response to noise of a vortex based spin transfer nano-oscillator. Phys. Rev. B **89**(10) (2014)

S.I. Kiselev, J. Sankey, I. Krivorotov, N. Emley, R. Schoelkopf, R. Buhrman, D. Ralph, Microwave oscillations of a nanomagnet driven by a spin-polarized current. Nature **425**(6956), 380 (2003)

L. Larger, M.C. Soriano, D. Brunner, L. Appeltant, J.M. Gutiérrez, L. Pesquera, C.R. Mirasso, I. Fischer, Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing. Opt. Express **20**(3), 3241–3249 (2012)

Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature **521**(7553), 436 (2015)

R. Lyon, A computational model of filtering, detection, and compression in the cochlea, in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82*, vol. 7 (IEEE, 1982), pp. 1282–1285

Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, S. Massar, Optoelectronic reservoir computing. Sci. Rep. **2**, 287 (2012)

R. Penrose, A generalized inverse for matrices, in *Mathematical proceedings of the Cambridge philosophical society*, vol. 51 (Cambridge University Press, 1955), pp. 406–413

M. Riou, F.A. Araujo, J. Torrejon, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, K. Yakushiji, A. Fukushima et al., Neuromorphic computing through time-multiplexing with a spin-torque nano-oscillator, in *Electron Devices Meeting (IEDM), 2017 IEEE International* (IEEE, 2017), p. 36.3.1

W.H. Rippard, M.R. Pufall, S. Kaka, T.J. Silva, S.E. Russek, Current-driven microwave dynamics in magnetic point contacts as a function of applied field angle. Phys. Rev. B **70**(10) (2004)

F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. **65**(6), 386 (1958)

A. Slavin, V. Tiberkevich, Nonlinear auto-oscillator theory of microwave generation by spin-polarized current. IEEE Trans. Magn. **45**(4), 1875–1918 (2009)

J. Torrejon, M. Riou, F.A. Araujo, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, K. Yakushiji, A. Fukushima, Neuromorphic computing with nanoscale spintronic oscillators. Nature **547**(7664), 428–431 (2017)

S. Tsunegi, H. Kubota, K. Yakushiji, M. Konoto, S. Tamaru, A. Fukushima, H. Arai, H. Imamura, E. Grimaldi, R. Lebrun et al., High emission power and q factor in spin torque vortex oscillator consisting of feb free layer. Appl. Phys. Express **7**(6) (2014)

S. Tsunegi, K. Yakushiji, A. Fukushima, S. Yuasa, H. Kubota, Microwave emission power exceeding 10 μW in spin torque vortex oscillator. Appl. Phys. Lett. **109**(25) (2016)

Q. Vinckier, F. Duport, A. Smerieri, K. Vandoorne, P. Bienstman, M. Haelterman, S. Massar, High-performance photonic reservoir computer based on a coherently driven passive cavity. Optica **2**(5), 438–446 (2015)

Reservoir Computing Based on Spintronics Technology

Tomohiro Taniguchi, Sumito Tsunegi, Shinji Miwa, Keisuke Fujii, Hitoshi Kubota, and Kohei Nakajima

Abstract Recent developments in reservoir computing based on spintronics technology are described here. The rapid growth of brain-inspired computing has motivated researchers working in a broad range of scientific field to apply their own technologies, such as photonics, soft robotics, and quantum computing, to brain-inspired computing. A relatively new technology in condensed matter physics called spintronics is also a candidate for application to brain-inspired computing because the small size of devices (nanometer order), their low energy consumption, their rich magnetization dynamics, and so on are advantageous for realization of highly integrated network systems. In fact, several interesting functions, such as a spoken-digit recognition and an associative memory operation, achieved using spintronics technology have recently been demonstrated. Here, we describe our recent advances in the development of recurrent neural networks based on spintronics auto-oscillators, called spin-torque oscillators, such as experimental estimation of the short-term memory capacity of a vortex-type spin-torque oscillator and numerical simulation

T. Taniguchi (✉) · S. Tsunegi · H. Kubota
National Institute of Advanced Industrial Science and Technology (AIST), Research Center for Emerging Computing Technologies, Central 2, 1-1-1, Umezono, Tsukuba, Ibaraki 305-8568, Japan
e-mail: tomohiro-taniguchi@aist.go.jp

S. Tsunegi
e-mail: tsunegi.sb@aist.go.jp

H. Kubota
e-mail: hit-kubota@aist.go.jp

S. Miwa
The Institute for Solid State Physics, The University of Tokyo, Kashiwa, Chiba 277-8581, Japan
e-mail: miwa@issp.u-tokyo.ac.jp

K. Fujii
Graduate School of Engineering Science, Osaka University, Toyonaka, Osaka 560-8531, Japan
e-mail: fujii@qc.ee.es.osaka-u.ac.jp

K. Nakajima
Graduate School of Information Science and Technology, The University of Tokyo, Bunkyo, Tokyo 113-8656, Japan
e-mail: k_nakajima@mech.t.u-tokyo.ac.jp

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_14

of reservoir computing using several macromagnetic oscillators. The results demonstrate the potential high performance of spintronics technology and its applicability to brain-inspired computing.

1 Introduction

A new candidate element for brain-inspired computing has recently appeared: spintronics, which is the main topic of this chapter. In this section, we provide a brief introduction to this research field.

1.1 Recurrent Neural Network and Reservoir Computing

The brain has long been a fascinating research target in science. Nonlinear interactions between neurons in the brain via synapses carry and/or store information, and prompt the growth of living things. Human beings, as well as biological systems in general, show surprising computational ability due to their brains. The brain can perform various functions, including (visual and/or audio logic) recognition, associative memory, prediction from past experiences, action modification, and task optimization. Moreover, the human brain has notably low power consumption for computation. If we could replace current computing systems based on the von Neumann architecture by neural networks, our lives would be drastically changed. Brain-inspired computing aimed at achieving artificial neural networks has attracted much attention in a wide range of scientific research fields, such as physics, chemistry, biology, engineering, and nonlinear science.

A crucial task in the advancement of brain-inspired computing is developing an appropriate model (Gerstner et al. 2014; Goodfellow et al. 2017) and implementing it in a real system. A class of artificial neural network, called recurrent neural network (RNN), is an architecture exhibiting a dynamic response to a time sequence of an input data (Mandic and Chambers 2001). The response of an RNN depends not only on the input and the system's weights at a certain time but also on the input at some previous time. In other words, RNNs store past input information. Thus, RNNs enable a time sequence of input data such as data on spoken languages and movies to be classified and calculated.

A reservoir computing system is an RNN in which the internal weights of the network are not changed by learning; only the reservoir-to-output weights are trained (Maass et al. 2002; Jaeger and Haas 2004; Verstraeten et al. 2007; Appeltant et al. 2011; Nakajima 2020). This approach simplifies the tuning of the weights used for computing and enables any physical system to be used as a "reservoir." Various such systems have been developed, including a photonic architecture with delayed feedback (Brunner et al. 2013), a soft robotic reservoir using body dynamics generated

from a soft silicone arm (inspired by octopus's arm in water) (Nakajima et al. 2015), and a quantum reservoir consisting of several qubits (Fujii and Nakajima 2017).

The purpose of this chapter is to introduce a promising new candidate for reservoir computing: spintronics. Spintronics is a relatively young research field in condensed matter physics and has been developed by investigating spin-dependent electron transport in solids with nanometer (nm) scale. Spintronics technology has several advantages for reservoir computing, such as its applicability to an array structure with nanoscale dimensions, low energy consumption, and strong nonlinearity. However, the research targets in spintronics to date have been electronic devices, such as nonvolatile memory and microwave generators, and the ability of spintronics technology from the viewpoint of artificial neural networks has not yet been thoroughly investigated. What is required in reservoir computing is rich (or complex) dynamics of physical systems because the reservoir should show several and distinguishable responses to different sequences and/or inputs. The richness (or complexity) of a dynamical system is quantitatively characterized by its memory and nonlinear capacities, which characterize, respectively, the amount of information the RNN can store and its nonlinear computational capability. Here, the capacities are estimated by examining the response to binary input as done in Jaeger (2002). Short-term memory capacity is simply characterized how much past information fed into the system can be stored and reconstructed from the current output of the reservoir with trained weight. This task can be accomplished even in a linear system, in principle. Therefore, the nonlinear computational capability on the stored information is further characterized by the parity check capacity by predicting the parity of the past input sequence, which essentially requires nonlinearity. For example, the short-term memory capacity of a quantum reservoir consisting of several qubits ($\geq$4) with virtual nodes ($\geq$4) was reported to be of the order of 10 (Fujii and Nakajima 2017). In this chapter, we describe our recent measurements of memory capacity in spintronics devices (Tsunegi et al. 2018a; Furuta et al. 2018).

We start by describing the history of this research field, and discuss the applicability of spintronics technology to brain-inspired computing.

1.2 History of Spintronics and Key Technologies

An electron has two degrees of freedom, charge and spin. Research on electricity and magnetics has a long history, and it was unified at the end of the nineteenth century as classical electromagnetism (Jackson 1999). People began to notice that both electricity and magnetism originate from a charged particle, but it took time to widely accept the existence of elementary particles (or atoms and molecules) until the beginning of the twentieth century. The rapid growth of quantum mechanics provided a new picture of electrons and of elementary particles, in general, which resulted in a drastic change in our understanding of the nature. In particular, the discovery of spin led the transition from classical to quantum physics because it was the first finding of an internal degree of freedom in elementary particles (Dirac

1928). Spin has relativistic quantum mechanical origin (Weinberg 2005), determines the statistics of the elementary particles (Pauli 1940), and is related to many physics, such as the stimulated emission of light (Dirac 1927), the origin of ferromagnetism (Heisenberg 1928), superconductivity (Bardeen et al. 1957), and the fate of stars (Chandrasekhar 1984). Importantly, the quantum mechanics has also resulted in the development of applied physics. Electronics based on semiconductors has provided many devices useful in everyday life. Magnetic devices as well have become widely used for such things as a storage device. Charge and spin have been, however, used separately for electronics and magnetic devices, respectively.

Spintronics (or spin electronics) is both electronics and magnetism at nanoscale and aims to develop new electronics devices, such as nonvolatile memory [magnetoresistive random access memory (MRAM)] and microwave generators, in which spin plays a key role. Quantum mechanical interactions between the spins of conducting electrons in metals and semiconductors, magnetization in ferromagnets, magnons (spin-wave quanta) in insulating ferromagnets, light helicity, and so on provide interesting new phenomena in fundamental physics and have opened the door to practical applications. The growth of spintronics is due to advances in nanostructure fabrication processes. This is because the spin of electrons in condensed matter is not a conserved quantity. For example, the spin in metals relaxes over the length scale (the spin diffusion length) (Valet and Fert 1993), which is typically of the order of nanometers (Bass and Pratt 2007); therefore, nanostructures are necessary to observe spin-dependent phenomena.

A key phenomenon in spintronics is the tunneling magnetoresistance (TMR) effect, which was found by Julliere in a magnetic tunnel junction (MTJ) consisting of Fe/Ge/Co (Julliere 1975). The magnetoresistance effect is a physical phenomenon in which the resistance of an electrical circuit depends on the magnetization directions of the ferromagnets. The TMR effect originates from spin-dependent tunneling through an insulator, and so it is a purely quantum mechanical phenomenon. Although several other magnetoresistance effects, such as anisotropic magnetoresistance and the planar Hall effect, have been found since the nineteenth century and their origins were revealed to be quantum mechanical phenomena in the middle of the twentieth century (Thomson 1856; Kundt 1893; Pugh and Rostoker 1953; Karplus and Luttinger 1954; McGuire and Potter 1975; Sinitsyn 2008; Nagaosa et al. 2010), the discovery of the TMR effect in 1975 is often regarded as a beginning of spintronics. Since the electrons pass through the interfaces between multilayers, the structure of an MTJ is often called a "current-perpendicular-to-plane (CPP)" structure. The research target shifted from MTJs (Julliere 1975; Maekawa and Gäfvert 1982; Maekawa and Shinjo 2002) to giant magnetoresistive (GMR) systems consisting of metallic ferromagnetic/nonmagnetic multilayers when the relatively large magnetoresistance effect, called the "GMR effect," was found in 1988–1989 by Fert and Grünberg independently (Baibich et al. 1988; Binasch et al. 1989). The GMR effect was first observed in a current-in-plane (CIP) structure, in which the electrons move in a direction parallel to the metallic interface. It was, however, soon noticed that the CPP structure shows a larger GMR effect than the CIP structure (Pratt et al. 1991; Zhang and Levy 1991). Research interest shifted to back MTJs when a large TMR effect was found in

an Fe/AlO/Fe MTJ in 1995 (Miyazaki et al. 1995; Moodera et al. 1995) and a giant TMR effect was found in an MTJ with an Fe/MgO/Fe MTJ in 2004 (Yuasa et al. 2004a; Parkin et al. 2004; Yuasa et al. 2004b).

A magnetoresistance effect, such as the TMR effect, enables the magnetization direction in a ferromagnet to be detected by using an electrical circuit. Typical TMR and GMR devices include two ferromagnets, and the resistance of the circuit is low (high) when the alignment of their magnetizations is parallel (antiparallel). In addition, the theoretical prediction of a spin-transfer torque (STT) effect by Slonczewski and Berger provided a new technology controlling the magnetization direction (Slonczewski 1989, 1996; Berger 1996; Slonczewski 2005). After the discovery of magnetism, the magnetization direction of a ferromagnet has been controlled by applying an external magnetic field. The STT effect is a completely different phenomenon, where an electric current is applied to a ferromagnetic/nonmagnetic/ferromagnetic trilayer structure. One ferromagnet, the "reference layer," acts as a polarizer of the spin angular momentums of the conducting electrons. When these spin-polarized electrons are injected into the other ferromagnet, the "free layer," the spins of the conducting electrons are transferred to the magnetization of the ferromagnet via quantum mechanical interaction and excite STT (Ralph and Stiles 2008). When the magnitude of the electric current is sufficiently large, the STT can change the magnetization direction in the free layer. Such magnetization reversals were observed in a GMR device in 2000 (Katine et al. 2000) and in an MTJ in 2005 (Kubota et al. 2005).

In summary, there are two important phenomena in spintronics, the TMR and STT effects. The magnetization direction of a nanostructured ferromagnet can be detected electrically by using the TMR effect and can be changed by using the STT effect. These are the operation principles for any kind of spintronics device (Locatelli et al. 2014). In fact, as discussed below, brain-inspired computing based on spintronics technology utilizes these phenomena. There are many other interesting phenomena in spintronics, such as the spin pumping effect (Silsbee et al. 1979; Tserkovnyak et al. 2002; Mizukami et al. 2002), the spin Hall effect (Dyakonov and Perel 1971; Hirsch 1999; Kato et al. 2004), and the topological effect (Murakami et al. 2003). Readers who are interested in spintronics can easily find good textbooks and review articles, such as Maekawa (2006), Shinjo (2009), and Maekawa et al. (2012).

1.3 Brain-Inspired Computing Based on Spintronics

There are many advantages of applying spintronics technology to brain-inspired computing (Grollier et al. 2016). First of all, the small size of spintronics device and their applicability to an array structure enable development of high-density computing devices. The typical size of an MTJ is of the order of 1 nm in thickness and 10–100 nm in diameter (Fukushima et al. 2018; Watanabe et al. 2018). The nonvolatility of the magnetization as memory results in low power consumption during operation (Dieny et al. 2016). The large output signal due to the giant TMR effect

is another advantage for electronics applications (Yuasa et al. 2004a; Parkin et al. 2004; Yuasa et al. 2004b; Djayaprawira et al. 2005; Ikeda et al. 2008; Yakushiji et al. 2010; Sukegawa et al. 2013; Kubota et al. 2013; Tsunegi et al. 2016a). The figure of merit for TMR and GMR devices is the magnetoresistance ratio, which is defined as $(R_{AP} - R_P)/R_P$, where R_P and R_{AP} are the resistances of the system for parallel (P) and antiparallel (AP) magnetization alignments. Currently, the TMR ratio is of the order of 100%, which results in a visible change in the output signal that is sufficient to experimentally detect by overcoming noise. Strong nonlinearity of the magnetization dynamics excited by the STT effect (Bertotti et al. 2005, 2007, 2009; Slavin and Tiberkevich 2009) is desirable for computation calculating a time sequence of input data. In fact, as mentioned below, we found that an MTJ with TMR and STT effects is applicable as an element for reservoir computing. The fast relaxation time of the magnetization dynamics (Zhou et al. 2010; Kudo et al. 2010; Suto et al. 2011; Nagasawa et al. 2011; Rippard et al. 2013; Jenkins et al. 2016; Taniguchi et al. 2017; Tsunegi et al. 2018b), of the order of 1–100 nanoseconds (ns), is suitable for fast computing. Application of these features to brain-inspired computing by using spintronics technology has recently been reported.

An associative memory operation in a network consisting of CIP devices was reported by Borders, Fukami, and their collaborators in 2017 (Borders et al. 2017). They used analog switching of the magnetization in a CoNi ferromagnet placed on a PtMn antiferromagnet as memory for storing the weight of computing. A total of 36 devices were used to demonstrate the associative memory operation of identifying a word consisting of three alphabetic letters.

Spoken-digit recognition competitive to that of state-of-the-art neural networks was demonstrated by Torrejon, Grollier, and their collaborators, including some of the present authors of this chapter, also in 2017 (Torrejon et al. 2017). A vortex spin-torque oscillator (STO) was used for the computation, and the information of a sequence of spoken words was stored using the nonlinear dynamics of the magnetic vortex. A high recognition rate ($\geq$95%) was obtained for spoken digits (0–9) pronounced by five speakers. The detail of this work is explained in the other chapter of this book. Vowel recognition with four coupled STOs was also achieved recently (Romera et al. 2018). In the following sections, we will revisit artificial neural networks based on an STO.

There have also been several reports of numerical simulation focused on brain-inspired computing based on a spintronics computing architecture (Furuta et al. 2018; Vodenicarevic et al. 2017; Kudo and Morie 2017; Huang et al. 2017; Nakane et al. 2018; Chen et al. 2018; Pinna et al. 2018; Arai and Imamura 2018; Nomura et al. 2018). For example, Kudo and Morie demonstrated pattern recognition using an array of STOs in 2017 (Kudo and Morie 2017). Nakane and his collaborators investigated the possibility of reservoir computing based on spin-wave excitation and detection in 2018 (Nakane et al. 2018). As can be seen, a wide variety of device designs have been proposed. This is due to the rich physics of nanostructured devices. The magnetization dynamics in a ferromagnet is usually highly nonlinear. Since spintronics devices consist of nanomagnets, magnetic interactions, such as dipole interaction and spin wave, can be used as an operation principle. In addition, electrical

interaction through the STT effect can also be an operation principle for such devices. Accordingly, spintronics provides interesting examples of dynamical systems and therefore can be considered a strong candidate as a fundamental element of brain-inspired computing, see, for example, a review (Grollier et al. 2020) including both experiments and theory.

1.4 Spin-Torque Oscillator (STO)

A key device for brain-inspired computing based on spintronics technology is the STO (Torrejon et al. 2017). An STO typically consists of an MTJ, for which the main structure is a trilayer (reference layer/insulating barrier/free layer), as illustrated in Fig. 1a. The typical material for the barrier is MgO (Yuasa et al. 2004a, b; Parkin et al. 2004). As mentioned above, when an electric current is injected into an MTJ, STT is excited on the magnetization in the free layer. The STT leads to magnetization dynamics when the magnitude of the electric current is sufficiently large. The magnetization dynamics is typically classified as reversal (Kubota et al. 2005) or oscillation (Kiselev et al. 2003). Magnetization reversal has been used as an operation principle for MRAM (Dieny et al. 2016). Magnetization oscillation has been widely studied for both GMR and TMR structures with the aim of developing practical applications such as microwave generator, highly sensitive sensors, and phased array radar (Kiselev et al. 2003; Rippard et al. 2004; Houssameddine et al. 2007). An STO device uses this magnetization oscillation in an MTJ for such practical applications.

Since STOs are nonlinear oscillator showing an auto-oscillation (a limit cycle), there are a dissipation due to friction and source of energy injection, as is any auto-oscillator in nature (Pikovsky et al. 2003). The source of the energy is the work

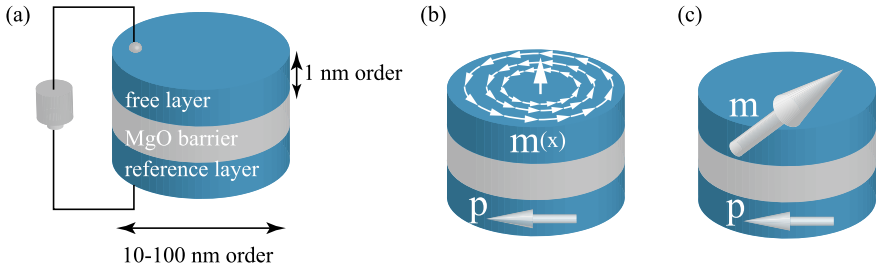


Fig. 1 **a** Schematic diagram of fundamental structure of MTJ connected to electric battery. **b** Schematic diagram of MTJ with vortex-type free layer. Each white arrow represents local direction of magnetic moment $\mathbf{m}(\mathbf{x})$. Magnetic moment at center points in the perpendicular direction with respect to film surface and is called a "vortex core." Magnetic moments around core lie in film plane. **c** Schematic diagram of MTJ with macromagnetic free layer. Almost all local magnetic moments point in same direction, so local variable $\mathbf{m}(\mathbf{x})$ can be replaced with macroscopic variable $\mathbf{m}$. In both **b**, **c**, direction of magnetization in reference layer, which is assumed to be macromagnetic, is represented as $\mathbf{p}$

done by the STT, while the dissipation is relaxation minimizing the magnetic energy of the ferromagnets. The oscillation frequency ranges from 100 MHz to 100 GHz depending on the device structure and magnetic configuration, see Sects. 2.1, 2.2, and 2.3 for more detail. The oscillation of the magnetization in an STO is detected through the TMR effect. The resistance of the MTJ is $R = 1/G$, where conductance G is given by

$$G = \frac{G_{\mathrm{P}} + G_{\mathrm{AP}}}{2} + \frac{G_{\mathrm{P}} - G_{\mathrm{AP}}}{2} \cos\theta, \tag{1}$$

where θ is the relative angle between the magnetizations of the free and reference layers, and $G_{\mathrm{P}} = 1/R_{\mathrm{P}}$ and $G_{\mathrm{AP}} = 1/R_{\mathrm{AP}}$ are the conductances for the parallel and antiparallel magnetization alignments. Therefore, the resistance, or voltage, of the STO oscillates over time when the magnetization in the free layer is in an auto-oscillation state. In other words, the oscillation state of the STO can be detected by electrical measurement.

Our motivation focusing on reservoir computing based on spintronics technology is as follows. Imagine that we fabricate an array of STOs. Each STO interacts with the other STOs through magnetic and/or electrical interactions. Such a system may experience phase synchronization (Slavin and Tiberkevich 2009), in which the phase differences between the magnetizations saturate to certain values. This synchronization phenomenon may be applicable to some portion of brain-inspired computing (Kudo and Morie 2017). It should be noted, however, that the coupling strength between the STOs normally cannot be changed after fabrication of the array. This means that the interconnections (weights) between the elements in the RNN cannot be tuned for computing. Reservoir computing solves this problem and thus opens the door to the applicability of STO devices to brain-inspired computing. The highly nonlinear, and thus complex, dynamics of STOs should provide rich dynamics to a reservoir and is unnecessary to be tuned for computing. To clarify the applicability of STO devices to reservoir computing it is necessary to evaluate the performance capabilities of STOs, which is the main objective of this chapter (Tsunegi et al. 2018a; Furuta et al. 2018).

2 Methods

In this section, we summarize the methods we used to evaluate the short-term memory and parity check capacities of reservoir computing with STOs.

2.1 Landau–Lifshitz–Gilbert Equation

We start by introducing a theoretical model of the magnetization dynamics frequently used in the fields of magnetism and spintronics. It is often assumed that the mag-

netization dynamics in a ferromagnet is described by the Landau–Lifshitz–Gilbert (LLG) equation with spin-transfer torque (Slonczewski 1996; Landau and Lifshitz 1935):

$$\frac{\mathrm{d}\mathbf{m}}{\mathrm{d}t} = -\gamma \mathbf{m} \times \mathbf{H} - \frac{\gamma \hbar P g(\theta) J}{2eM_s d} \mathbf{m} \times (\mathbf{p} \times \mathbf{m}) + \alpha \mathbf{m} \times \frac{\mathrm{d}\mathbf{m}}{\mathrm{d}t}, \tag{2}$$

where $\mathbf{m}$ is the unit vector pointing in the local magnetization direction. Magnetic field $\mathbf{H}$ consists of both internal and external fields; the internal fields include an exchange field, anisotropy field, and dipole field. Parameter γ, the "gyromagnetic ratio," is related to Bohr magneton μ_B and Landé g-factor g_L via $\gamma = g_\mathrm{L}\mu_\mathrm{B}/\hbar$. Saturation magnetization M_s is the magnetic moment per volume, whereas d is the thickness of the free layer. Spin polarization P ($0 \leq P < 1$) characterizes the strength of the STT, and the vector $\mathbf{p}$ ($|\mathbf{p}| = 1$), which is usually parallel to the magnetization of the reference layer, determines the direction of the STT. The current density flowing in the MTJ is denoted as J. Factor $g(\theta)$, a function of the relative angle between $\mathbf{m}$ and $\mathbf{p}$ ($\cos\theta = \mathbf{m} \cdot \mathbf{p}$), determines the angular dependence of the STT. The explicit form of $g(\theta)$ depends on the structure of the spintronics device (Xiao et al. 2004; Taniguchi et al. 2015). Dimensionless parameter α, the "damping constant," which characterizes the strength of the dissipation due to friction of the magnetization dynamics (Gilbert 2004) and is of the order of $10^{-3} - 10^{-2}$ for typical ferromagnets used in spintronics devices (Oogane et al. 2006). The magnetization dynamics in a ferromagnet has been investigated by solving the LLG equation both analytically (Bertotti et al. 2009) and numerically (Lee et al. 2004).

The LLG equation is a nonlinear differential equation. The first term on the right-hand side of Eq. (2) is the field-torque term and describes a steady precession (oscillation) of the magnetization around the magnetic field. The second (STT) term moves the magnetization parallel or antiparallel to the direction of $\mathbf{p}$ depending on the direction of the electric current. The third term (the damping term) describes the energy dissipation of the ferromagnet phenomenologically (Gilbert 2004). Auto-oscillation of magnetization $\mathbf{m}$ in an STO is excited when the energy dissipation ($\propto \alpha$) is cancelled by the energy supplied by the work done by the STT (Bertotti et al. 2009). When this condition is satisfied, the auto-oscillation around the magnetic field, which is described by the field-torque term, is realized. The oscillation trajectory is almost on a constant energy curve (Bertotti et al. 2009).

2.2 *Micro- and Macro-Structures of Nanomagnet*

For the following discussions, we need to briefly review the microstructure of the free layer. As mentioned above and shown in Fig. 1a, the size of the free layer is typically 10–100 nm in diameter and 1 nm in thickness. Such a ferromagnet consists of numerous magnetic atoms, so its macroscopic magnetization consists of numerous magnetic moments. The direction of each magnetic moment in equilibrium is determined to minimize the magnetic energy E, which is related to the magnetic field $\mathbf{H}$

via $\mathbf{H} = -\delta E/\delta(M\mathbf{m})$ (Lifshitz and Pitaevskii 1980), and consists of the exchange energy, magnetic anisotropy energy, dipole field energy, and Zeeman energy (Hubert and Schäfer 1998).

When the free layer, in particular the thickness, is relatively large, the microstructure of each magnetic moment becomes a vortex so as to minimize the stray field energy, i.e., the magnetic moment at the center of the area points in the direction perpendicular to the cross-sectional area of the free layer whereas the magnetic moments around the center exhibit a chiral structure, as illustrated in Fig. 1b. In this case, **m** is a function of location **x** inside the free layer. Another micromagnetic structure called domain wall separates two uniformly magnetized domains. One way to investigate the magnetization dynamics in such systems is to use the LLG equations to compute the magnetization dynamics of all the magnetic moments simultaneously, i.e., by using numerical micromagnetic simulation (Brown and LaBonte 1965; Hayashi and Goto 1971; Nakatani et al. 1989, 2003; Dussaux et al. 2010). Another way to investigate the dynamics is to reduce the LLG equation to an equation of motion of the vortex core. The reduced equation is called Thiele equation, which describes the motions of the radius and rotation angle in the film plane of the vortex core (Thiele 1973; Guslienko 2006; Liu et al. 2007; Khvalkovskiy et al. 2009; Dussaux et al. 2012; Grimaldi et al. 2014).

On the other hand, when the free layer is relatively small, almost all magnetic moments point in the same direction so as to minimize the exchange and anisotropy energies. In this case, we can replace local magnetization **m** in Eq. (2) with a macroscopic single moment as illustrated in Fig. 1c. This model is called a macromagnetic or macrospin model. The macromagnetic magnetization in the thin magnetic film used in spintronics devices usually points in the direction parallel to the film plane. This is because this in-plane magnetized configuration minimizes the stray field generated outside the ferromagnet and reduces the magnetic energy. The magnetization can be, however, oriented in the direction perpendicular to the film plane by controlling the perpendicular anisotropy energy. For example, adding an MgO barrier neighboring a CoFe ferromagnet induces an interface anisotropy effect (Hine et al. 1979; Yakata et al. 2009; Ikeda et al. 2010; Kubota et al. 2012). Applying an electrical voltage also changes the perpendicular anisotropy due to interface effect (Weisheit et al. 2007; Chiba et al. 2008; Maruyama et al. 2009; Nozaki et al. 2010; Shiota et al. 2012). The perpendicularly magnetized MTJ is of great interest for MRAM applications because it is suitable for high-density structures.

The number of variables in Eq. (2) for the vortex structure is $2\mathcal{N}$, where $\mathcal{N}$ is the number of magnetic moments. In the macromagnetic case, the number of variables is 2. One might consider that, since vector **m** is defined in a three-dimensional space, the number of variables should be $3\mathcal{N}$ and 3 for the vortex and macromagnetic case. However, the LLG equation conserves the norm of magnetization **m** because the equation satisfies $\mathrm{d}|\mathbf{m}|^2/\mathrm{d}t = (1/2)\mathbf{m} \cdot (\mathrm{d}\mathbf{m}/\mathrm{d}t) = 0$, so there is a constraint condition $|\mathbf{m}| = 1$, which reduces the number of independent variables. This conservation of the magnetization norm means that the temperature of the system is sufficiently lower than the Curie temperature of the ferromagnet. According to the Poincaré–Bendixson theorem (Wiggins 1990), chaos is precluded in a macromagnetic model

without interaction with other ferromagnets and/or feedback. Magnetization **p** in the reference layer is usually assumed to be macromagnetic.

The ferromagnets used in spintronics devices typically have two energy minima due to uniaxial anisotropy energy. Such a ferromagnet can be used as a bit of memory devices. The strengths of the STT and damping torque represented in Eq. (2) depend on the direction of the magnetization. Related to this fact, there are two thresholds, J_c and J_{sw}, of current density J in Eq. (2). When $J/J_c > 1$, the STT exceeds the damping torque near the equilibrium state and destabilizes the magnetization. When $J/J_{sw} > 1$, the magnetization moves from one equilibrium to another. Note that auto-oscillation of the magnetization is excited in range $J/J_c > 1$ and $J/J_{sw} < 1$. To satisfy these conditions, J_{sw}/J_c should be larger than one. If this condition is not satisfied, i.e., $J_{sw}/J_c < 1$, the MTJ exhibits the switching without showing an auto-oscillation (Taniguchi et al. 2013; Taniguchi 2015). This means that not all MTJs exhibit an auto-oscillation. The magnitude relationship between J_c and J_{sw} depends on the magnetic field and STT angular dependence.

The oscillation frequencies of a vortex and macromagnetic STO are typically of the order of 100 MHz and 1–10 GHz, respectively. A vortex-type STO was used for the experimental estimation of short-term memory capacity (Sect. 3) whereas the macromagnetic model was used for the numerical simulation (Sect. 4).

2.3 Recurrent Neural Network Based on STO

We developed two models of an STO-based RNN. One uses a single vortex-type STO, while the other uses a single macromagnetic STO or nine macromagnetic STOs as a reservoir. In both models, the output voltage from the STO is used as the dynamic response from the reservoir. According to Eq. (1), the resistance, or voltage, of an STO oscillates due to the oscillation of magnetization **m** in the free layer ($\cos\theta = \mathbf{m} \cdot \mathbf{p}$). The output voltage from the STO is given by

$$v_{out}(t) = v_a(t) \sin\left[2\pi f t + \varphi_{out}(t)\right], \tag{3}$$

where f is the oscillation frequency of the TMR. Amplitude v_a and phase φ_{out} of the voltage are constant in an auto-oscillation state. On the other hand, when an additional current (voltage) pulse is applied to the STO, they vary over time. This is because the additional torque changes the balance between the STT and damping torque, causing the magnetization to move a different oscillation state. The relaxation process for the new oscillation state depends on the sequence of additional voltage pulses. In other words, the STO in the relaxation process stores past input information. Therefore, the STO is a candidate for reservoir computing. The ability of a reservoir is quantitatively characterized by its memory and nonlinear capacities, as mentioned in Sect. 1.1. Therefore, it is necessary to evaluate the memory and nonlinear capacities of STO devices.

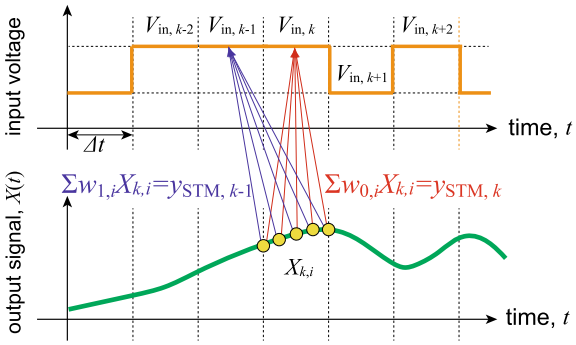


Fig. 2 Schematic illustration of input voltage V_{in} and output signal $X(t)$. The numbers such as k and $k+1$ indicate the pulse numbers. The kth input voltage is $V_{\text{in},k} = V_{\text{low}} + (V_{\text{high}} - V_{\text{low}})s_{\text{in},k}$; the corresponding output signal at the ith node is $X_{k,i}$, where $s_{\text{in},k}$ is input binary data. For evaluation of short-term memory capacity, training data $y_{\text{STM},k}$ was directly related to input data via $y_{\text{STM},k} = s_{\text{in},k}$. Weight $w_{D,i}$ was determined so as to reproduce $y_{\text{STM},k}$ ($V_{\text{in},k-D}$) from $X_{k,i}$. Cases $D = 0$ and $D = 1$ are shown in figure. For evaluation of the parity check capacity, training data $y_{\text{PC},k}$ was related to input binary data (voltage) via Eq. (12) (After Tsunegi et al. 2018a)

2.4 Methods for Evaluating Memory Capacities

Here, we describe methods for evaluating the short-term memory and parity check capacities.

We first explain, with the help of Fig. 2, how we evaluate short-term memory capacity. The figure schematically shows how the input is applied to the STO and how the output is obtained. As mentioned, memory capacity is evaluated by examining the response to binary input. We first sequentially apply Z-bit random voltage pulses (Z: integer) to the STO. The input voltage is expressed as $V_{\text{in},k} = V_{\text{low}} + (V_{\text{high}} - V_{\text{low}})s_{\text{in},k}$, where V_{low} and V_{high} are the lower and higher values of the input voltages, respectively, whereas $s_{\text{in},k} = 0$ or 1 is the random binary data. The memory capacity is evaluated by examining the response of the reservoir to binary input (Jaeger 2002), and therefore masking procedure (Appeltant et al. 2011) is not applied to the input data in this work. The total length of the voltage pulses is $Z\Delta t$, where Δt corresponds to the duration of 1 bit. These input data are called training data. The kth ($k = 1, 2, \ldots, Z$) pulse is applied from time $t = (k-1)\Delta t$ to $t = k\Delta t$.

The STT excited by the input voltage changes the oscillation trajectory of the magnetization, resulting in a change in the voltage output from the STO, as schematically shown in Fig. 2. Not only the output voltage but also the STO variables can be used for the evaluation. For example, in Sect. 3, voltage amplitude v_{a} is used as the output signal, whereas resistance x is used in the simulation described in Sect. 4. In this section, we denote a general signal output from an STO as $X(t)$. We divide the output signal from $t = (k-1)\Delta t$ to $t = k\Delta t$ into N-nodes, where N is the number of virtual nodes. These virtual nodes are used, along with linear regression, to obtain the output from the reservoir. For simplicity, we denote the output signal at the ith

$(i = 1, 2, \ldots, N)$ node as $X_{k,i}$, i.e.,

$$X_{k,i} = X\left((k-1)\Delta t + i\frac{\Delta t}{n}\right). \tag{4}$$

Weight $w_{0,i}$ is then introduced to obtain the output from the reservoir;

$$\sum_{i=1}^{N+1} w_{0,i} X_{k,i}. \tag{5}$$

The learning task here is to find the weight that minimizes the error between the output from the reservoir and the training data, which is defined as

$$\sum_{k=1}^{Z}\left(\sum_{i=1}^{N+1} w_{0,i} X_{k,i} - y_{\mathrm{STM},k}\right)^2, \tag{6}$$

where $y_{\mathrm{STM},k}$ is the training data used for evaluating short-term memory capacity, and is related, specifically for the case without any delay, to the input binary data $s_{\mathrm{in},k}$;

$$y_{\mathrm{STM},k} = s_{\mathrm{in},k}. \tag{7}$$

The meaning of the suffix "0" in $w_{0,i}$ in Eq. (6) will be explained below. The term with subscript $i = N + 1$ corresponds to the bias term used to tune the constant value, and $X_{k,N+1} = 1$ in Eq. (6).

Weight $w_{0,i}$ introduced above is determined to reproduce training data $y_{\mathrm{STM},k}$ applied during time $(k-1)\Delta t \le t \le k\Delta t$. Since an RNN is an artificial neural network used for classifying and calculating a time sequence of input data, it should be able to reproduce the past input voltage from the current output signal. To enable this functionality, Eq. (6) is extended:

$$\sum_{k=1}^{Z}\left(\sum_{i=1}^{N+1} w_{D,i} X_{k,i} - y_{\mathrm{STM},k-D}\right)^2, \tag{8}$$

where $D = 1, 2, 3, ..$ is the delay. Weight $w_{D,i}$ is set so as to reproduce the $(k - D)$th input binary data $y_{\mathrm{STM},k-D}$ applied during time $(k - D - 1)\Delta t \le t \le (k - D)\Delta t$ from output signal $X(t)$ obtained during time $(k-1)\Delta t \le t \le k\Delta t$.

After this learning, other Z'-random pulse sequences $s'_{\mathrm{in},n}$ are applied to the STO as voltages, where Z' is the number of input data instances. The prime symbol is used to distinguish the quantities related to testing from those related to learning. Test data $y'_{\mathrm{STM},n}$ is defined similarly to Eq. (7), i.e., $y'_{\mathrm{STM},n} = s'_{\mathrm{in},n}$. The output signal at the ith node responsive to the nth test data is denoted as $X'_{n,i}$. Using the weight determined by the learning, we define the reconstructed data as

$$y'_{\mathrm{R},n-D} = \sum_{i=1}^{N+1} w_{D,i} X'_{n,i}, \tag{9}$$

where $X_{n,N+1} = 1$. Whether the reconstructed data reproduces the test data with delay D is characterized by correlation coefficient $\mathrm{Cor}(D)$ given by

$$\mathrm{Cor}(D) = \frac{\sum_{n=1}^{Z'} \left(y'_{\mathrm{STM},n-D} - \langle y'_{\mathrm{STM},n-D}\rangle\right)\left(y'_{\mathrm{R},n-D} - \langle y'_{\mathrm{R},n-D}\rangle\right)}{\sqrt{\sum_{n=1}^{Z'} \left(y'_{\mathrm{STM},n-D} - \langle y'_{\mathrm{STM},n-D}\rangle\right)^2 \sum_{n=1}^{Z'} \left(y'_{\mathrm{R},n-D} - \langle y'_{\mathrm{R},n-D}\rangle\right)^2}}, \tag{10}$$

where $\langle \cdots \rangle$ is the average value. The short-term memory capacity is estimated as

$$C_{\mathrm{STM}} = \sum_{D=1}^{Z'} [\mathrm{Cor}(D)]^2. \tag{11}$$

The short-term memory capacity characterizes the ability of the reservoir to store past input as is, which can be done by simply using a linear transformation. The nonlinear transformation ability for the stored information can be examined by using a parity check task, where the following task data are used for the learning and testing:

$$y_{\mathrm{PC},k-D} = \sum_{i=0}^{D} s_{\mathrm{in},k-D+i} \mod 2. \tag{12}$$

The ability of the nonlinear transformation is quantitatively evaluated by introducing parity check capacity C_{PC}, which is defined similarly to Eq. (11). In the evaluation of parity check capacity, the data input to the STO is again binary voltage data. However, when the weight is determined, $y_{\mathrm{PC},k-D}$ given by Eq. (12) should be substituted into the right-hand side of Eqs. (6) and (8). Similarly, during testing, test data $y'_{\mathrm{PC},n}$ are defined from the input binary data in accordance with Eq. (12) whereas the reconstructed data y'_{R} are obtained from the signal output from the STO and the weight determined by the learning, as shown in Eq. (9). The parity check capacity C_{PC} is evaluated from the correlation between the test and reconstructed data.

3 Reservoir Computing with STO (Experiment)

In this section, we report our experimental evaluation of the short-term memory capacity of a vortex-type STO.

3.1 Experimental Methods

The STO used in this study was made from an MTJ with a multilayer structure: sub/buffer/PtMn(15)/CoFe(2.5)/Ru(0.86)/CoFeB(1.6)/CoFe(0.8)/MgO(1.1)/CoFe (0.5)/ FeB(7.0)/CoFe(0.5)/ MgO(1.0)/Ta/Ru (thickness in nm). The films were patterned into an STO with a diameter of 425 nm. The CoFe/Ru/CoFeB/CoFe and CoFe/FeB/CoFe structures correspond to the free and reference layers, respectively. A magnetic vortex state, as schematically shown in Fig. 1b, appears in the free layer in this type of MTJ (Tsunegi et al. 2016a, 2018b).

Figure 3a schematically shows the circuit used for evaluating the short-term memory capacity of the STO. We first prepared random binary bits as input data. The data were converted into voltage pulses by using an arbitrary wave generator (AWG). Voltage pulses with duration Δt and a 5 ns lead edge were applied to the STO through a low-pass filter using a bias-tee with a cutoff frequency of 300 MHz. We used filters corresponding to a Gaussian filter and applied the inverse of the circuit transfer function to the input voltage to reduce the distortion in preprocessing. The waveform of the high-frequency output voltage was measured at room temperature using a real-time oscilloscope (5–10 Gsam/s) using a high-pass filter with a cutoff frequency of 400 MHz. The STO generated an oscillating voltage through auto-oscillation of the vortex core induced by the perpendicular component (film normal) of the STT (Dussaux et al. 2010). Therefore, we applied an out-of-plane magnetic field of 725 mT to the oscillator. The voltage output from the STO was separated from the input voltage by using the high-pass filter and measured with the oscilloscope. Amplitude v_a of the output voltage was estimated by using Hilbert transformation and used as the output signal X for evaluating short-term memory capacity.

Figure 3b, c shows examples of the input and output voltages for offset voltage $V_{offset} = 250$ mV and $\Delta t = 20$ ns. The input voltage was 200 or 300 mV, and the corresponding oscillation frequency of the STO was 540 or 555 MHz, respectively. The output voltage exhibited a temporal change in magnitude, in accordance with the change in the input pulse voltage. The amplitude is the envelope of the oscillating voltage, as shown by the red curve in Fig. 3c. It was used to estimate the short-term memory capacity.

In our experiments, we used 3500 random bits in total, where the first 500 bits were used for the washout, the following 2000 ($Z = 2000$) bits were used for learning, and the last 1000 ($Z' = 1000$) bits were used to estimate the short-term memory capacity. Hereafter, we call this sequence of 3500 bits a "single shot."

Since the dynamics of the vortex core exhibited randomness due to thermal fluctuation, the initial conditions, as well as the amplitude noise in the output voltage, differed for every trial. Thus, the output voltage from the STO differed every run even when the same voltage pulses were used as input. Therefore, we repeated the single-shot experiment 60 times using fixed training and test data. The amplitude was averaged, with which the test data was also reconstructed. Using the average amplitude for both learning and testing should reduce the amplitude noise, leading to an increase in short-term memory capacity.

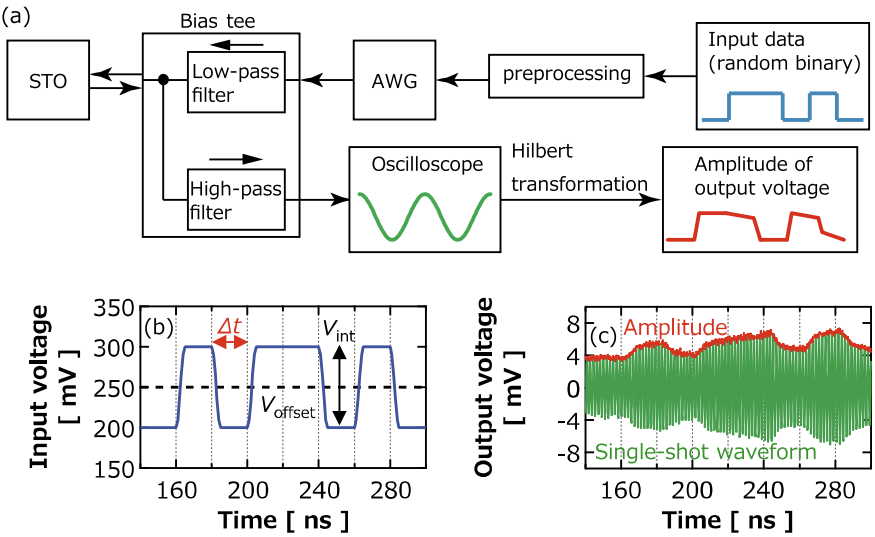


Fig. 3 **a** Measurement circuit used in experiments. Input data were preprocessed and injected into the STO through an arbitrary wave generator (AWG) and low-pass filter. Output voltage was measured with an oscilloscope after passing through the high-pass filter. Amplitude of output voltage was estimated using Hilbert transformation. Memory capacity was evaluated by comparing amplitude of output voltage with input data. **b** Example of sequential input-voltage pulses and **c** voltage output from STO. Intensity, offset, and pulse duration of input voltage were $V_{\text{int}} = 100$ mV, $V_{\text{offset}} = 250$ mV, and $\Delta t = 20$ ns, respectively (After Tsunegi et al. 2018a)

3.2 *Short-Term Memory Capacity in Single STO*

Figure 4a shows comparisons between the test (red solid line) and reconstructed (blue dotted line) data using 200 nodes in a single-shot experiment with delay $D = 1$, 2, or 3. For $D = 1$, the reconstructed data reproduced the test data almost perfectly, indicating that the STO remembered the input voltage applied one bit before the present pulse. In contrast, the ability to reconstruct data was low for $D = 2$ and 3. Using the averaging technique improved the reconstruction data for $D = 2$, as can be seen in Fig. 4b. These reconstructed data and Eq. (10) were used to evaluate the correlation coefficients.

Figure 5a illustrates the relationship between delay D and correlation $\text{Cor}(D)$ for the single-shot (red solid line) and averaged data (blue dotted line). The short-term memory capacities estimated from these data and Eq. (11) were nearly 1.0 for the single-shot experiment and 1.8 for the "averaged 60 times" experiment. These results indicate that reducing the amplitude noise of the output voltage is important for increasing short-term memory capacity.

We performed the same experiment but with values of Δt other than 20 ns. The pulse duration dependency of the short-term memory capacity obtained from the averaged results is plotted in Fig. 5b by circles. The short-term memory capacity

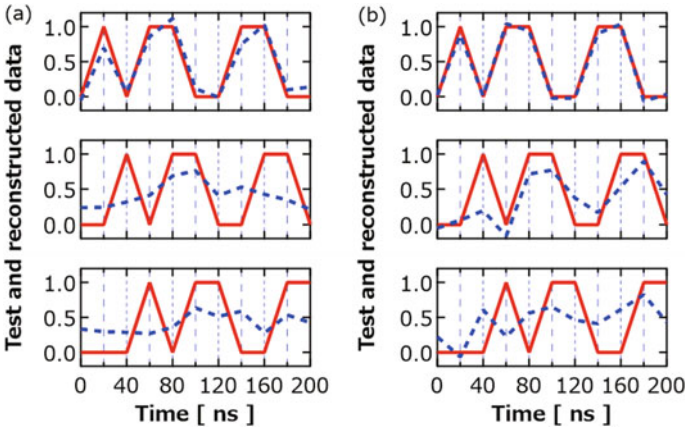


Fig. 4 Comparisons between test (red solid line) and reconstructed (blue dotted line) data for **a** single-shot and **b** averaged data. Delay D was 1, 2, and 3 from top to bottom (After Tsunegi et al. 2018a)

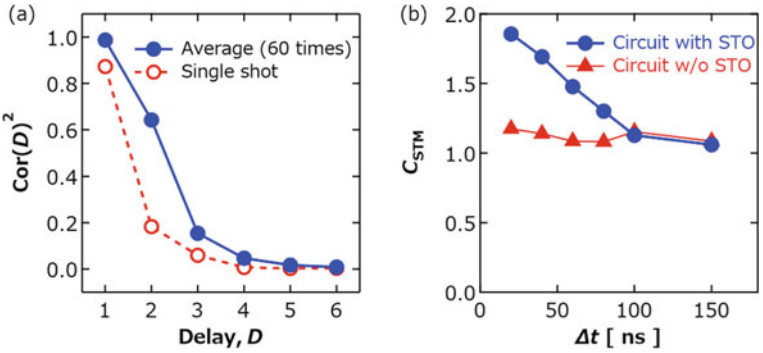


Fig. 5 **a** Square of correlation Cor(D) between the test and reconstructed data for single-shot (red dotted) and averaged (blue solid) data as a function of delay D. **b** Dependence of memory capacity in circuit without (red triangles) and with (blue circles) STO on pulse duration Δt (After Tsunegi et al. 2018a)

decreased monotonically with an increase in the pulse duration for the following reason. When a voltage pulse is applied to the STO, the vortex core starts to relax to an oscillation orbit (limit cycle) determined by the voltage. Now let us assume that a voltage pulse is applied at $t = t_0$ and that the next pulse is applied at $t = t_0 + \Delta t$. If pulse width Δt is sufficiently greater than the relaxation time of the vortex core, the output voltage for $t > t_0 + \Delta t$ is not correlated to that for $t < t_0$. Therefore, the short-term memory capacity decreases with an increase in the pulse width. The pulse width should thus be less than the relaxation time of the STO for reservoir computing. The relaxation time of the STO strongly depends on the type of oscillator. Roughly speaking, the relaxation time is of the order of $1/(\alpha f)$, where α and f are the

damping constant and oscillation frequency, respectively (Taniguchi et al. 2017). The relaxation time of a vortex-type STO is of the order of 10−100 ns (Jenkins et al. 2016; Tsunegi et al. 2018b) whereas that of a macromagnetic STO is of the order of 1−10 ns (Kudo et al. 2010; Suto et al. 2011; Nagasawa et al. 2011). Evaluation of STO relaxation time will be a key issue for designing reservoir computing based on spintronics technology.

3.3 Contributions from Other Circuit Components

The definition used for short-term memory capacity in this work needs to be mentioned. The memory function of reservoir computing arises from the nonlinear response of the system to the input data. In the present system, not only the STO but also the other components in the circuit, such as the AWG and low-pass filter (Fig. 3a), exhibit nonlinear responses to input data. Therefore, the short-term memory capacity evaluated above should be, strictly speaking, regarded as that of the whole system, and includes the contributions from the other components of the circuit.

To validate the existence of the memory function in the STO, we also evaluated the short-term memory capacity of the system without the oscillator. The results are plotted in Fig. 5b by triangles. The short-term memory capacity of the circuit with the STO exceeded that of the circuit without the STO, proving that the STO has memory functionality.

3.4 Future Directions

Although the finite value of memory capacity found in this work supports the potential application of spintronics devices to artificial neural networks, the short-term memory capacity is still low compared with that of other systems such as a quantum reservoir with several qubits and virtual nodes (Fujii and Nakajima 2017). The low memory capacity of the present system is due to the trade-off between noise reduction of noise and capacity enhancement. As explained above, the memory of the past input voltage is stored in the amplitude of the output voltage. Noise in the amplitude of the output voltage reduces memory capacity. Therefore, the changes in the output voltage amplitude between neighboring nodes should be large enough to be distinguished from the noise. However, a large change in the output voltage amplitude between the neighboring nodes means fast relaxation, leading to a reduction in memory capacity. Therefore, solutions that further enhance memory capacity are needed.

A potential solution is to use an STO with delayed feedback (Khalsa et al. 2015; Tsunegi et al. 2016b), where the information of past input is naturally stored in the delayed feedback loop, resulting in an increase in short-term memory capacity (Riou et al. 2019; Yamaguchi et al. 2020). Another is to use frequency and/or phase locking of the STO by using forced or mutual synchronization (Marković et al. 2019; Tsunegi

et al. 2019). This is because locking by synchronization leads to a reduction in the amplitude and/or phase noise and thus stabilizes the output voltage (Kaka et al. 2005; Mancoff et al. 2005; Zhou and Akerman 2009; Locatelli et al. 2015; Urazhdin et al. 2016; Awad et al. 2017; Tsunegi et al. 2018c). Investigating such solution is a future research direction in reservoir computing using spintronics technology.

4 Reservoir Computing with STO (Simulation)

In this section, we discuss the quantitative analysis of the figures-of-merit for reservoir computing using MTJ devices by employing macromagnetic simulation. Specifically, not only a reservoir computing system with single MTJ but also one with multiple MTJs is discussed (Furuta et al. 2018).

4.1 Simulation Methods

Figure 6a, b shows schematics of the system used for simulating reservoir computing using MTJs. As mentioned above, an MTJ contains an insulating tunneling barrier layer with two ferromagnetic layers. For ferromagnetic layer-1 (the reference layer), the magnetization direction is designed to be fixed. This can be done by the exchange bias effect using antiferromagnetic materials, such as PtMn and IrMn (Meiklejohn and Bean 1956), or magnetic anisotropy energy. Here, the magnetization direction of layer-1 was fixed perpendicular to the film plane. For ferromagnetic layer-2 (the free layer), the magnetization direction was not fixed and thus could be controlled by the current (Katine et al. 2000) or voltage (Shiota et al. 2012). The MTJ device resistance reflects the magnetization direction.

The magnetization dynamics in ferromagnetic layer-2 follows the LLG equation with STT given by Eq. (2), without thermal fluctuation in the ferromagnetic layers (Brown 1963). Here, **p** and **m** in Eq. (2) correspond to the unit magnetization vectors for ferromagnetic layers 1 and 2 in Fig. 6(a) and 6(b), respectively. The effective magnetic field in the present study is given by

$$\mathbf{H} = -H_{\mathrm{a}zz} m_z \mathbf{e}_z, \tag{13}$$

where $H_{\mathrm{a}zz}$ is a uniaxial perpendicular anisotropy field. The z-component of **m** is denoted as m_z, and $\mathbf{e}_z$ is the unit vector parallel to the z-axis. In our definition, $H_{\mathrm{a}zz} > 0$ ($H_{\mathrm{a}zz} < 0$) shows in-plane (perpendicular) magnetic anisotropy. The angular dependence of the STT in Eq. (2) is $g(\theta) = 1/(1 + P^2 \cos\theta)$ (Slonczewski 2005). The resistance of the MTJ varies with the relative angle between the spins in the free and pinned layers, $R = 1/G$ [see also Eq. (14)]

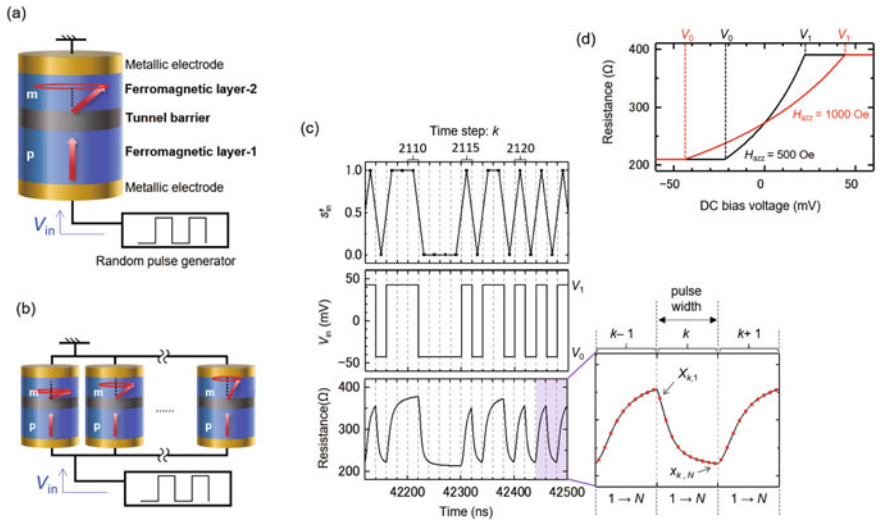


Fig. 6 **a** Schematic of reservoir system using magnetization dynamics in an MTJ. **b** System with multiple MTJs. Magnetization direction of ferromagnetic layer-2 ($\mathbf{s}_2$) could be controlled by input bias voltage V_{in}; Magnetization direction of ferromagnetic layer-1 ($\mathbf{p}$) is fixed. **c** Input s_{in}, bias voltage V_{in} input to MTJ device, and MTJ device resistance as function of time, which are typical characteristics during learning and evaluation processes. Virtual nodes [$x_{k,1}$ to $x_{k,N}$ during kth binary input] were defined as shown in figure. **d** MTJ device resistance as function of static input DC bias voltage. Black and red plots indicate resistance, when uniaxial anisotropy fields were 500 Oe and 1000 Oe, respectively. V_0 and V_1 are voltages that rendered device resistance constant (After Furuta et al. 2018)

$$R = \frac{2R_{\text{AP}}R_{\text{P}}}{(R_{\text{AP}} + R_{\text{P}}) + (R_{\text{AP}} - R_{\text{P}})(\mathbf{m} \cdot \mathbf{p})}. \tag{14}$$

The time evolution of the MTJ resistance is characterized by sequential calculation using the fourth Runge–Kutta method. For evaluating the short-term memory and parity check capacities, an input pulse voltage, V_{in}, corresponding to the computational input, $s_{\text{in}}(= 0 \text{ or } 1)$, was applied to the MTJs, as depicted in Fig. 6c. Figure 6(a) and 6(b) shows the schematics of circuits with single and multiple MTJs, respectively. In this section, the physical parameters are $H_{azz} = 1000$ Oe, $\alpha = 0.009$, $R_{\text{P}} = 210\ \Omega$, and $R_{\text{AP}} = 390\ \Omega$. These values almost follow our previous experimental research (Miwa et al. 2014).

Figure 6c shows an example of MTJ device resistance under input voltage V_{in}. We used a pulse voltage with binary values of $V_0(= -44$ mV) and $V_1(= +44$ mV) as V_{in}. These binary values correspond to 0 and 1 in s_{in}, respectively, in the reservoir computing learning and evaluation processes. The pulse width (20 ns in Fig. 6c, for instance) corresponds to the discrete unit time step. Because the device resistance is scalar, the node dimension is simply one. However, the number of nodes can be increased by using virtual nodes (Appeltant et al. 2011; Nakajima et al. 2018). As

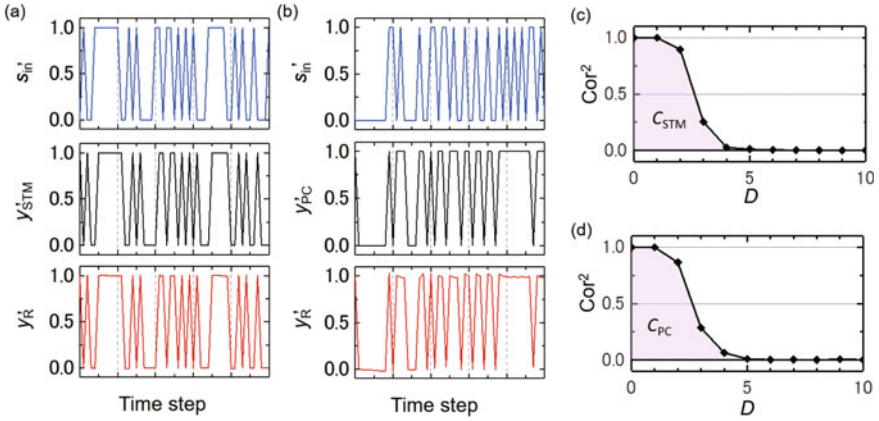


Fig. 7 **a** Input s'_{in}, test data y'_{STM} [Eq. (8) with $D = 1$], and reconstructed data y'_{R} used for evaluating the short-term memory task. **b** Input s'_{in}, test data y'_{PC}, (Eq. (9) with $D = 1$), and reconstructed data y'_{R} used for evaluating parity check task. **c** Correlation using Eq. (10); integrated values are defined as short-term memory capacity (C_{STM}). **d** Correlation using Eq. (10); integrated values are defined as the parity check capacity (C_{PC}). Input-voltage pulse width was 20 ns, and number of virtual nodes was 50 (After Furuta et al. 2018)

shown in Fig. 6c, the virtual nodes $x_{k,1}$ to $x_{k,N}$ during the kth binary input are defined, where output signal $x_{k,i}$ in the figure is the resistance with suffix $i = 1, 2, \ldots, N$ corresponding to the node number. These virtual nodes are further defined as a node vector $\mathbf{x}_k$.

Figure 6d depicts the DC bias voltage dependence of the static MTJ device resistance. Under DC bias voltage conditions, the resistance was measured after the magnetization dynamics were damped. Under positive bias voltage conditions, the spin-polarized current flowed from the free layer to the pinned layer, and the STT induced auto-oscillation in **m**. The relative magnetization angle between **p** and **m** increased, and an antiparallel-like magnetization configuration was realized. This means that the device resistance increases when a positive bias voltage is applied. Under negative bias voltage conditions, a parallel-like magnetization configuration was induced, and the device resistance decreases. For the input pulse voltage depicted in Fig. 6c, the binary values V_0 and V_1 are defined as voltages that rendered device resistance constant. As shown in Fig. 6d, V_0 and V_1 depended on uniaxial magnetic anisotropy H_{azz}.

4.2 Short-Term Memory and Parity Check Capacities in Single STO

In this section, we present figures-of-merit for reservoir computing using a single MTJ device. The uniaxial magnetic anisotropy of the free layer **m** was fixed: $H_{\text{azz}} =$

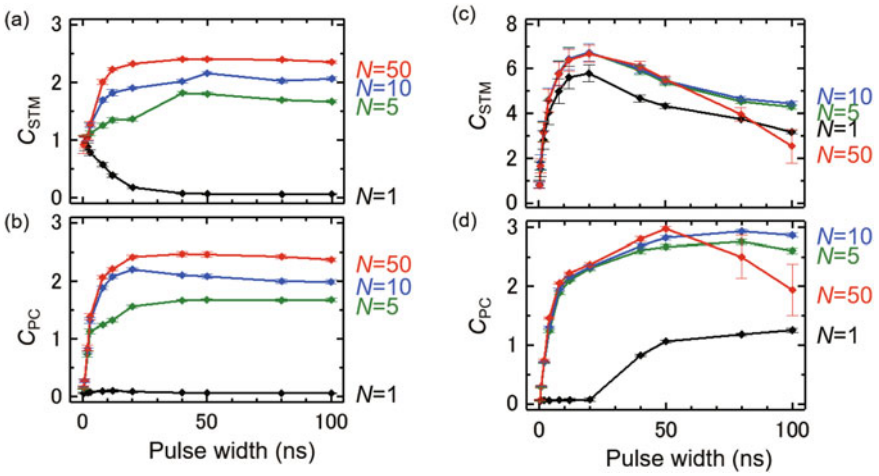


Fig. 8 **a**, **b** Results of reservoir computing using single MTJ for short-term memory capacity (C_{STM}), and parity check capacity (C_{PC}) as a function of input-voltage pulse width; N is number of virtual nodes in MTJ. **c**, **d** Results of reservoir computing using multiple MTJs for C_{STM} and C_{PC} with seven MTJs ($M = 7$) as function of input-voltage pulse width. Uniaxial magnetic anisotropy ratio ($H_{azz,k+1}/H_{azz,k}$) was 1.6 (After Furuta et al. 2018)

1000 Oe. Recall that a positive value of H_{azz} means that the magnetic cell in the MTJ is in-plane magnetized. Figure 7 shows the simulated data used for evaluating the short-term memory and parity check capacities for a single MTJ. The input-voltage pulse width was 20 ns and the number of virtual nodes, N, was 50. Figure 7a shows typical simulation results for input s'_{in}, test data for short-term memory task y'_{STM}, and reconstructed data y'_R as a function of time. Similarly, Fig. 7b shows input s'_{in}, test data for parity check task y'_{PC}, and trained output y'_R. Delay D in both figure was 1. The training and test data for the short-term memory task and parity check task are defined by Eqs. (7) and (12), respectively. The output was calculated using the simulated MTJ resistance (see Fig. 6c) and the weight. The weight was trivially calculated using the definitions given by Eq. (8). Figure 7c, d depicts the correlations [Eq. (10)] between the test and reconstructed data as a function of D. C_{STM} and C_{PC} are defined as the numerical integration of the correlation [see Eq. (11)] and as the capacity using training data for the short-term memory and parity check, respectively.

Figure 8(a) and 8(b) shows the short-term memory C_{STM} and parity check C_{PC} capacities, respectively, as functions of the input-voltage pulse width with virtual nodes N. Both C_{STM} and C_{PC} increased with the pulse width until it reached ~ 20 ns. They then remained nearly constant. When the pulse width is less than 20 ns, the change in the magnetization direction is very small, so the magnetization dynamics cannot work as a reservoir.

4.3 Short-Term Memory and Parity Check Capacities in Multiple STOs

When multiple MTJs are used for reservoir computing, higher figures-of-merit can be obtained. A schematic of a multiple-MTJ circuit for reservoir computing is depicted in Fig. 6b. MTJs are placed in parallel, and an identical pulse voltage is applied to all of them. Spatial multiplexing (Nakajima et al. 2019) is used to construct the nodes for reservoir computing. The node vector has $M \times N$ elements, where M is the number of MTJs and N is the number of virtual nodes in an MTJ:

$$\mathbf{x}_{L,k} = \begin{pmatrix} x_{L,k,1} \\ \vdots \\ x_{L,k,N} \end{pmatrix}, \qquad (L = 1, 2, \ldots, M), \tag{15}$$

$$\mathbf{x}_k \equiv \left(x_{1,k,1} \cdots x_{1,k,N}\ x_{2,k,1} \cdots x_{2,k,N} \cdots x_{M,k,N}\right)^{\mathrm{t}}, \tag{16}$$

where "t" indicates a transposed vector, and $x_{L,k,i}$ is the signal output from the Lth MTJ ($L = 1, 2, \ldots, M$) at the ith node ($i = 1, 2, \ldots, N$) in response to the kth binary data input ($k = 1, 2, \ldots, Z$).

We tested various sets of uniaxial anisotropy $H_{\mathrm{a}zz}$ of ferromagnetic layer-2 in each MTJ (see Table 1). For instance, when four MTJs were used and $H_{\mathrm{a}zz,k}/H_{\mathrm{a}zz,k+1} = 2$, the uniaxial anisotropies of the MTJs were 1000 Oe, 500 Oe, 250 Oe, and 125 Oe. Such variations in anisotropy can be obtained by voltage-controlled magnetic anisotropy in the MTJs (Maruyama et al. 2009). In this study, thermal fluctuation in ferromagnetic layer-2 was neglected. For instance, thermal fluctuation energy at room temperature (26 meV) is negligibly small compared to magnetization energy $M_{\mathrm{s}} H_{\mathrm{a}zz} \mathcal{V}/2$ ($\sim$ 10 eV) when $H_{\mathrm{a}zz}$ = 1000 Oe. Here, M_{s} and $\mathcal{V}$ are the saturation magnetization and volume of the free layer and were assumed to be 1375 emu/c.c. and 23500 nm^3, respectively (Miwa et al. 2014). Therefore, thermal fluctuation is comparable to or less than the magnetization energy of ferromagnetic layer-2 for $H_{\mathrm{a}zz} < 3$ Oe. In this region, simulation assuming the ground state is not very accurate, so a random magnetic field to reproduce the thermal fluctuation (Brown 1963) should be included in the simulation. Similar to the procedure for a single MTJ, the binary values V_0 and V_1, the voltage input to the MTJs, were determined as shown in Fig. 6d. Note that V_0 and V_1 vary as a function of the uniaxial anisotropy field, and the smallest absolute values of the saturation voltages are used for as V_0 and V_1 for reservoir computing with multiple MTJs, i.e., V_0 and V_1 are determined for the MTJ with the smallest uniaxial magnetic anisotropy field.

We characterized C_{STM} and C_{PC} as functions of anisotropy ratio $H_{\mathrm{a}zz,k}/H_{\mathrm{a}zz,k+1}$. In the simulation, the input-voltage pulse width was 20 ns, and the number of virtual nodes for each MTJ was 50 for all MTJs. The maximum value of C_{STM} increased with the number of MTJs (M). Because each MTJ had a different uniaxial magnetic

Table 1 Sets of uniaxial magnetic anisotropy used for reservoir computing with multiple MTJs

$H_{azz}/H_{azz,k+1}$	$H_{azz,1}$	$H_{azz,2}$	$H_{azz,3}$	$\cdots$	$H_{azz,7}$
1.0	1000 Oe	1000 Oe	1000 Oe	$\cdots$	1000 Oe
1.1	1000 Oe	909.1 Oe	826.4 Oe	$\cdots$	564.5 Oe
1.2	1000 Oe	833.3 Oe	694.4 Oe	$\cdots$	334.9 Oe
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
2.9	1000 Oe	344.8 Oe	118.9 Oe	$\cdots$	1.7 Oe
3.0	1000 Oe	333.3 Oe	111.1 Oe	$\cdots$	1.4 Oe

anisotropy field H_{azz}, it had a different response speed to an external voltage/current. This variation in the response speed increased the C_{STM} of the system. In contrast, the increase in C_{PC} was insignificant compared to that in C_{STM} because there was no electric and/or magnetic interaction between the free layers of the MTJs. For instance, we found that $H_{azz,k}/H_{azz,k+1} = 1.6$ was the best condition for maximizing C_{STM} for $M = 7$. As shown in Fig. 8c, the C_{STM} was maximum around a pulse width of 20 ns. When the pulse width was smaller, the change in the magnetization direction due to the STT is too small for performing as a reservoir. When the pulse width was greater than 20 ns, the magnetization dynamics was almost completely damped during each unit time step, and such a condition is not preferable for reservoir computing. As shown in Fig. 8d, the best conditions are not the same for C_{STM} and C_{PC}. This is because a relatively long pulse is required to induce nonlinearity in the magnetization dynamics, in multiple MTJs.

4.4 Comparison with Echo-State Network

We used an echo-state network for comparison (Jaeger and Haas 2004; Jaeger 2001). An echo-state network is also a kind of RNN, in which the input-to-reservoir weight $\mathbf{W}_{\mathrm{in}}$ and internal weights W between nodes are random, whereas the reservoir-to-output weight is optimized. The node vector at time step k of the echo-state network is determined in terms of the node vector at the previous step and the input at time step k. The following function is used as a node vector of the echo-state network,

$$\mathbf{x}_{\mathrm{ESN},k} = \tanh\left(\mathsf{W}\mathbf{x}_{k-1} + \mathbf{W}_{\mathrm{in}} s_{\mathrm{in},k}\right). \tag{17}$$

The hyperbolic tangent function is used for component-wise projection. Weights W and $\mathbf{W}_{\mathrm{in}}$ are given in a matrix and vector, respectively, in which the components are time-independent random values from -1 to $+1$. We normalized W by dividing each component of W by the spectral radius, which is the largest absolute value of the eigenvalues (singular value) of the weight matrix (Verstraeten et al. 2007). Weight $\mathbf{W}_{\mathrm{in}}$ is also normalized by its spectral radius. The C_{STM} and C_{PC} obtained

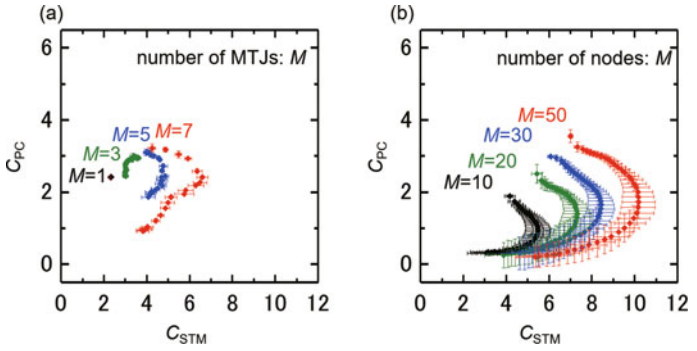


Fig. 9 Plots showing C_{STM} versus C_{PC} in **a** MTJ system and **b** echo-state network. In MTJ system, pulse width and number of virtual nodes (N) of each MTJ were fixed to 20 ns and 50, respectively (After Furuta et al. 2018)

using a multiple MTJ system are plotted in Fig. 9a; the pulse width was 20 ns and the number of virtual node was 50 for each MTJ. The data points from top to bottom correspond to increase in $H_{\mathrm{azz},k}/H_{\mathrm{azz},k+1}$ from 1.1 to 3.0. The C_{STM} and C_{PC} obtained using the echo-state network are shown in Fig. 9b. The data points from top to bottom correspond to increase the spectrum radius of the weight from 0.05 to 2.0. The results shown in Fig. 9 indicate that high-performance reservoir computing, similar to that of an echo-state network with 20–30 nodes, can be obtained for reservoir computing using 5–7 MTJs. In terms of the total number of virtual nodes in the system ($M \times N$), 35 nodes (7×5) in an MTJ system are comparable to 20–30 nodes in an echo-state network. Although C_{PC} increased slightly with M, we can obtain a large C_{PC} if there are magnetic and/or electrical interactions between the free layers in each MTJ.

5 Conclusion

In this chapter, we have described advances in brain-inspired computing devices based on spintronics technology. We have reviewed recent experimental and numerical work and reported our efforts in investigating the applicability of spintronics auto-oscillators (spin-torque oscillators, STOs) to recurrent neural networks. Spintronics devices have several attractive features for brain-inspired computing, such as low power consumption, applicability to high-density structures, a large output signal, and highly nonlinear and fast magnetization dynamics. Therefore, spintronics technology is promising for further development of artificial neural networks. However, the basic properties of spintronics devices, from the viewpoint of brain-inspired computing, have not been fully revealed yet. To prove the applicability of spintronics technology to computing, we need a deep understanding of magnetization dynamics in nanomagnets.

We have experimentally evaluated the short-term memory capacity of a reservoir computing with an STO. We have demonstrated the feasibility of learning and testing from output voltage by applying a time sequence of random voltage pulses to the STO. The short-term memory capacity obtained from the average output voltage was maximized to 1.8 under optimum conditions. Although this value includes the contribution not only from the STO but also from the other circuit components, the oscillator was shown to have a finite memory functionality. This indicates that it is important to reduce the amplitude noise in an STO applied to reservoir computing. The short-term memory capacity was increased by using a short pulse duration of 20 ns, which is comparable to the relaxation time (10–100 ns) of the oscillator. This indicates that it is also important to set the duration of the input pulses appropriately.

Using macromagnetic simulation, we demonstrated reservoir computing using the magnetization dynamics in MTJs. With reservoir computing using 5–7 MTJs, we can obtain performance similar to that of an echo-state network using hyperbolic tangent function with 20–30 nodes. Higher performance can be obtained by enabling magnetic and/or electrical interactions between the free layers in each MTJ.

Acknowledgements The authors are thankful to Julie Grollier at CNRS/Thales for valuable discussions. S. T., T. T., and H. K. are grateful to Kay Yakushiji, Akio Fukushima, and Shinji Yuasa at the National Institute of Advanced Industrial Science and Technology (AIST) for their supports. S. M., K. F., S. T., and H. K. are grateful to Taishi Furuta, Eiiti Tamura, Minori Goto, Shota Hasebe, Koki Shimose, and Yoshishige Suzuki at the Osaka University for their discussions. T. T. is grateful to Takahide Kubota, Atsushi Sugihara, Satoshi Iba, Aurelie Spiesser, Hiroki Maehara, and Ai Emura for their support and encouragement. K. N., S. T., and T. T. are supported from a project subsidized by the New Energy and Industrial Technology Development Organization (NEDO). S. M. is supported by JSPS KAKENHI (No. JP18H03880, JP26103002). K. F. is supported by JSPS KAKENHI No. 16H02211, JST ERATO Grant No. JPMJER1601, JST CREST Grant No. JPMJCR1673, and JST PRESTO Grants No. JPMJPR15E7 and No. JPMJPR1668. K. N. is supported by JST PRESTO Grant Number JPMJPR15E7, Japan, and JSPS KAKENHI Grant Numbers JP18H05472, JP16KT0019, and JP15K16076.

References

L. Appeltant, M.C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C.R. Mirasso, I. Fischer, Nat. Commun. **2**, 468 (2011)

H. Arai, H. Imamura, Phys. Rev. Appl. **10**, 024040 (2018)

A.A. Awad, P. Dürrenfeld, A. Houshang, M. Dvornik, E. Iacocca, R.K. Dumas, J. Akerman, Nat. Phys. **13**, 292 (2017)

M.N. Baibich, J.M. Broto, A. Fert, F. Nguyen van Dan, F. Petroff, P. Etienne, G. Creuzet, A. Friederich, J. Chazelas, Phys. Rev. Lett. **61**, 2472 (1988)

J. Bardeen, L.N. Cooper, J.R. Schrieffer, Phys. Rev. **108**, 1175 (1957)

J. Bass, W.P. Pratt Jr., J. Phys.: Condens. Matter **19**, 183201 (2007)

L. Berger, Phys. Rev. B **54**, 9353 (1996)

G. Bertotti, C. Serpico, I.D. Mayergoyz, A. Magni, M. d'Aquino, R. Bonin, Phys. Rev. Lett. **94**, 127206 (2005)

G. Bertotti, C. Serpico, I.D. Mayergoyz, R. Bonin, M. d'Aquino, J. Magn. Magn. Mater. **316**, 285 (2007)

G. Bertotti, I. Mayergoyz, C. Serpico, *Nonlinear Magnetization Dynamics in Nanosystems* (Elsevier, Oxford, 2009)
G. Binasch, P. Grünberg, F. Saurenbach, W. Zinn, Phys. Rev. Lett. **63**, 664 (1989)
W.A. Borders, H. Akima, S. Fukami, S. Moriya, S. Kurihara, Y. Horio, S. Sato, H. Ohno, Appl. Phys. Express **10**, 013007 (2017)
W.F. Brown Jr., Phys. Rev. **130**, 1677 (1963)
W.F. Brown Jr., A.E. LaBonte, J. Appl. Phys. **36**, 1380 (1965)
D. Brunner, M.C. Soriano, C.R. Mirasso, I. Fischer, Nat. Commun. **4**, 1364 (2013)
S. Chandrasekhar, Rev. Mod. Phys. **56**, 137 (1984)
M.-C. Chen, A. Sengupta, K. Roy, IEEE Trans. Magn. **54**, 1500270 (2018)
D. Chiba, M. Sawicki, Y. Nishiani, Y. Nakatani, F. Matsukura, H. Ohno, Nature **455**, 515 (2008)
B. Dieny, R.B. Goldfarb, K.-J. Lee (eds.), *Introduction to Magnetic Random-Access Memory* (Wiley-IEEE Press, Hoboken, 2016)
P.A.M. Dirac, Proc. Roy. Soc. Lond. A **114**, 243 (1927)
P.A.M. Dirac, Proc. Roy. Soc. Lond. A **117**, 610 (1928)
D.D. Djayaprawira, K. Tsunekawa, M. Nagai, H. Maehara, S. Yamagata, N. Watanabe, S. Yuasa, Y. Suzuki, K. Ando, Appl. Phys. Lett. **86**, 092502 (2005)
A. Dussaux, B. Georges, J. Grollier, V. Cros, A.V. Khvalkovskiy, A. Fukushima, M. Konoto, H. Kubota, K. Yakushiji, S. Yuasa, K.A. Zvezdin, K. Ando, A. Fert, Nat. Commun. **1**, 8 (2010)
A. Dussaux, A.V. Khvalkovskiy, P. Bortolotti, J. Grollier, V. Cros, A. Fert, Phys. Rev. B **86**, 014402 (2012)
M.I. Dyakonov, V.I. Perel, Phys. Lett. A **35**, 459 (1971)
K. Fujii, K. Nakajima, Phys. Rev. Appl. **8**, 024030 (2017)
A. Fukushima, T. Taniguchi, A. Sugihara, K. Yakushiji, H. Kubota, S. Yuasa, AIP Adv. **8**, 055925 (2018)
T. Furuta, K. Fujii, K. Nakajima, S. Tsunegi, H. Kubota, Y. Suzuki, S. Miwa, Phys. Rev. Appl. **10**, 034063 (2018)
W. Gerstner, W.M. Kistler, R. Naud, L. Paninski, *Neuronal Dynamics* (Cambridge University Press, Cambridge, 2014)
T.L. Gilbert, IEEE Trans. Magn. **40**, 3443 (2004)
I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning* (MIT Press, Cambridge, 2017)
E. Grimaldi, A. Dussaux, P. Bortolotti, J. Grollier, G. Pillet, A. Fukushima, H. Kubota, K. Yakushiji, S. Yuasa, V. Cros, Phys. Rev. B **89**, 104404 (2014)
J. Grollier, D. Querlioz, M.D. Stiles, Proc. IEEE **104**, 2024 (2016)
J. Grollier, D. Querlioz, K.Y. Camsari, K. Everschor-Sitte, S. Fukami, M.D. Stiles, Nat. Electron. **3**, 360 (2020)
K.Y. Guslienko, Appl. Phys. Lett. **89**, 022510 (2006)
N. Hayashi, E. Goto, Jpn. J. Appl. Phys. **10**, 128 (1971)
W.J. Heisenberg, Z. Phys. **49**, 619 (1928)
S. Hine, T. Shinjo, T. Takada, J. Phys. Soc. Jpn. **47**, 767 (1979)
J.E. Hirsch, Phys. Rev. Lett. **83**, 1834 (1999)
D. Houssameddine, U. Ebels, B. Delaët, B. Rodmacq, I. Firastrau, F. Ponthenier, M. Brunet, C. Thirion, J.-P. Michel, L. Prejbeanu-Buda, M.-C. Cyrille, O. Redon, B. Dieny, Nat. Mater. **6**, 447 (2007)
Y. Huang, W. Kang, X. Zhang, Y. Zhou, W. Zhao, Nanotechnology **28**, 08LT02 (2017)
A. Hubert, R. Schäfer, *Magnetic Domains* (Springer, Berlin, 1998)
S. Ikeda, J. Hayakawa, Y. Ashizawa, Y.M. Lee, K. Miura, H. Hasegawa, M. Tsunoda, F. Matsukura, H. Ohno, Appl. Phys. Lett. **93**, 082508 (2008)
S. Ikeda, K. Miura, H. Yamamoto, K. Mizunuma, H.D. Gan, M. Endo, S. Kanai, J. Hayakawa, F. Matsukura, H. Ohno, Nat. Mater. **9**, 721 (2010)
J.D. Jackson, *Classical Electrodynamics*, 3rd edn. (John Wiley and Sons Inc, Hoboken, 1999)
H. Jaeger, GMD Report **148**, 13 (2001)
H. Jaeger, GMD Report **152**, 60 (2002)

H. Jaeger, H. Haas, Science **304**, 78 (2004)
A.S. Jenkins, R. Lebrun, E. Grimaldi, S. Tsunegi, P. Bortolotti, H. Kubota, K. Yakushiji, A. Fukushima, G. de Loubens, O. Klein, S. Yuasa, V. Cros, Nat. Nanotechnol. **11**, 360 (2016)
M. Julliere, Phys. Lett. A **54**, 225 (1975)
S. Kaka, M.R. Pufall, W.H. Rippard, T.J. Silva, S.E. Russek, J.A. Katine, Nature **437**, 389 (2005)
R. Karplus, J.M. Luttinger, Phys. Rev. **95**, 1154 (1954)
J.A. Katine, F.J. Albert, R.A. Buhrman, E.B. Myers, D.C. Ralph, Phys. Rev. Lett. **84**, 3149 (2000)
Y.K. Kato, R.C. Myers, A.C. Gossard, D.D. Awschalom, Science **306**, 1910 (2004)
G. Khalsa, M.D. Stiles, J. Grollier, Appl. Phys. Lett. **106**, 0242402 (2015)
A.V. Khvalkovskiy, J. Grollier, A. Dussaux, K.A. Zvezdin, V. Cros, Phys. Rev. B **80**, 140401 (2009)
S.I. Kiselev, J.C. Sankey, I.N. Krivorotov, N.C. Emley, R.J. Schoelkopf, R.A. Buhrman, D.C. Ralph, Nature **425**, 380 (2003)
H. Kubota, A. Fukushima, Y. Ootani, S. Yuasa, K. Ando, H. Maehara, K. Tsunekawa, D.D. Djayaprawira, N. Watanabe, Y. Suzuki, Jpn. J. Appl. Phys. **44**, L1237 (2005)
H. Kubota, S. Ishibashi, T. Saruya, T. Nozaki, A. Fukushima, K. Yakushiji, K. Ando, Y. Suzuki, S. Yuasa, J. Appl. Phys. **111**, 07C723 (2012)
H. Kubota, K. Yakushiji, A. Fukushima, S. Tamaru, M. Konoto, T. Nozaki, S. Ishibashi, S. Saruya, S. Yuasa, T. Taniguchi, H. Arai, H. Imamura, Appl. Phys. Express **6**, 103003 (2013)
K. Kudo, T. Morie, Appl. Phys. Express **10**, 043001 (2017)
K. Kudo, T. Nagasawa, K. Mizushima, H. Suto, R. Sato, Appl. Phys. Express **3**, 043002 (2010)
A. Kundt, Ann. Phys. (Berlin) **285**, 257 (1893)
L. Landau, E. Lifshitz, Phys. Z. Sowjet **8**, 153 (1935)
K.-J. Lee, A. Deac, O. Redon, J.-P. Noziéres, B. Dieny, Nat. Mater. **3**, 877 (2004)
E.M. Lifshitz, L.P. Pitaevskii, *Statistical Physics Part 2, Course of Theoretical Physics*, vol. 9 (Butterworth-Heinemann, Oxford, 1980)
Y. Liu, H. He, Z. Zhang, Appl. Phys. Lett. **91**, 242501 (2007)
N. Locatelli, V. Cros, J. Grollier, Nat. Mater. **13**, 11 (2014)
N. Locatelli, A. Hamadeh, F.A. Araujo, A.D. Belanovsky, P.N. Skirdkov, R. Lebrun, V.V. Naletov, K.A. Zvezdin, M. Munoz, J. Grollier, O. Klein, V. Cross, G. de Loubens, Sci. Rep. **5**, 17039 (2015)
W. Maass, T. Natschläger, H. Markram, Neural Comput. **14**, 2531 (2002)
S. Maekawa (ed.), *Concepts in Spin Electronics* (Oxford Science Publications, Oxford, 2006)
S. Maekawa, U. Gäfvert, IEEE Trans. Magn. **18**, 707 (1982)
S. Maekawa, T. Shinjo (eds.), *Spin Dependent Transport in Magnetic Nanostructures* (CRC Press, Boca Raton, 2002)
S. Maekawa, S.O. Valenzuela, E. Saitoh, T. Kimura (eds.), *Spin Current* (Oxford Science Publications, Oxford, 2012)
F.B. Mancoff, N.D. Rizzo, B.N. Engel, S. Tehrani, Nature **437**, 393 (2005)
D.P. Mandic, J.A. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability* (Wiley, 2001)
D. Marković, N. Leroux, M. Riou, F.A. Araujo, J. Torrejon, D. Querlioz, A. Fukushima, S. Yuasa, J. Trastoy, P. Bortolotti, J. Grollier, Appl. Phys. Lett. **114** (2019)
T. Maruyama, Y. Shiota, T. Nozaki, K. Ohta, N. Toda, M. Mizuguchi, A.A. Tulapurkar, T. Shinjo, M. Shiraishi, S. Mizukami, Y. Ando, Y. Suzuki, Nat. Nanotechnol. **4**, 158 (2009)
T.R. McGuire, R.I. Potter, IEEE Trans. Magn. **11**, 1018 (1975)
W.H. Meiklejohn, C.P. Bean, Phys. Rev. **102**, 1413 (1956)
S. Miwa, S. Ishibashi, H. Tomita, T. Nozaki, E. Tamura, K. Ando, N. Mizuochi, T. Saruya, H. Kubota, K. Yakushiji, T. Taniguchi, H. Imamura, A. Fukushima, S. Yuasa, Y. Suzuki, Nat. Mater. **13**, 50 (2014)
T. Miyazaki, N. Tezuka, J. Magn. Magn. Mater. **139**, L231 (1995)
S. Mizukami, Y. Ando, T. Miyazaki, Phys. Rev. B **66**, 104413 (2002)
J.S. Moodera, L.R. Kinder, T.M. Wong, R. Meservey, Phys. Rev. Lett. **74**, 3273 (1995)
S. Murakami, N. Nagaosa, S.-C. Zhang, Science **301**, 1348 (2003)

N. Nagaosa, J. Sinova, S. Onoda, A.H. MacDonald, N.P. Ong, Rev. Mod. Phys. **82**, 1539 (2010)
T. Nagasawa, H. Suto, K. Kudo, K. Mizushima, R. Sato, J. Appl. Phys. **109**, 07C907 (2011)
K. Nakajima, Jpn. J. Appl. Phys. **59**, 060501 (2020)
K. Nakajima, H. Hauser, T. Li, R. Pfeifer, Sci. Rep. **5**, 10487 (2015)
K. Nakajima, H. Hauser, T. Li, R. Pfeifer, Soft Robot. **5**, 339 (2018)
K. Nakajima, K. Fujii, M. Negoro, K. Mitarai, M. Kitagawa, Phys. Rev. Appl. **11**, 034021 (2019)
R. Nakane, G. Tanaka, A. Hirose, IEEE Access **6**, 4462 (2018)
Y. Nakatani, Y. Uesaka, N. Hayashi, Jpn. J. Appl. Phys. **28**, 2485 (1989)
Y. Nakatani, A. Thiaville, J. Miltat, Nat. Mater. **2**, 521 (2003)
H. Nomura, T. Furuta, K. Tsujimoto, Y. Kuwabiraki, F. Peper, E. Tamura, S. Miwa, M. Goto, R. Nakatani, Y. Suzuki, Jpn. J. Appl. Phys. **58**, 070901 (2018)
T. Nozaki, Y. Shiota, M. Shiraishi, T. Shinjo, Y. Suzuki, Appl. Phys. Lett. **96**, 022506 (2010)
M. Oogane, T. Wakitani, S. Yakata, R. Yilgin, Y. Ando, A. Sakuma, T. Miyazaki, Jpn. J. Appl. Phys. **45**, 3889 (2006)
S.S.P. Parkin, C. Kaiser, A. Panchula, P.M. Rice, B. Hughes, M. Samant, S.-H. Yang, Nat. Mater. **3**, 862 (2004)
W. Pauli, Phys. Rev. **58**, 716 (1940)
A. Pikovsky, M. Rosenblum, J. Kurths, *Synchronization: A Universal Concept in Nonlinear Sciences* (Cambridge University Press, Cambridge, 2003)
D. Pinna, F.A. Araujo, J.-V. Kim, V. Cros, D. Querlioz, P. Bessiere, J. Droulez, J. Grollier, Phys. Rev. Appl. **9**, 064018 (2018)
W.P. Pratt Jr., S.F. Lee, J.M. Slaughter, R. Loloee, P.A. Schroeder, J. Bass, Phys. Rev. Lett. **66**, 3060 (1991)
E.M. Pugh, N. Rostoker, Rev. Mod. Phys. **25**, 151 (1953)
D.C. Ralph, M.D. Stiles, J. Magn. Magn. Mater. **320**, 1190 (2008)
M. Riou, J. Torrejon, B. Garitaine, F.A. Araujo, P. Bortolotti, V. Cros, S. Tsunegi, K. Yakushiji, A. Fukushima, H. Kubota, S. Yuasa, D. Querlioz, M.D. Stiles, J. Grollier, Phys. Rev. Appl. **12**, 024049 (2019)
W.H. Rippard, M.R. Pufall, S. Kaka, S.E. Russek, T.J. Silva, Phys. Rev. Lett. **92**, 027201 (2004)
W. Rippard, M.R. Pufall, A. Kos, Appl. Phys. Lett. **103**, 182403 (2013)
M. Romera, P. Talatchian, S. Tsunegi, F.A. Araujo, V. Cros, P. Bortolotti, J. Trastoy, K. Yakushiji, A. Fukushima, H. Kubota, S. Yuasa, M. Ernoult, D. Vodenicarevic, T. Hirtzlin, N. Locatelli, D. Querlioz, J. Grollier, Nature **563**, 230 (2018)
T. Shinjo (ed.), *Nanomagnetism and Spintronics* (Elsevier, Amsterdam, 2009)
Y. Shiota, T. Nozaki, F. Bonell, S. Murakami, T. Shinjo, Y. Suzuki, Nat. Mater. **11**, 39 (2012)
R.H. Silsbee, A. Janossy, P. Monod, Phys. Rev. B **19**, 4382 (1979)
N.A. Sinitsyn, J. Phys.: Condens. Matter **20**, 023201 (2008)
A. Slavin, V. Tiberkevich, IEEE Trans. Magn. **45**, 1875 (2009)
J.C. Slonczewski, Phys. Rev. B **39**, 6995 (1989)
J.C. Slonczewski, J. Magn. Magn. Mater. **159**, L1 (1996)
J.C. Slonczewski, Phys. Rev. B **71**, 024411 (2005)
H. Sukegawa, S. Mitani, T. Ohkubo, K. Inomata, K. Hono, Appl. Phys. Lett. **103**, 142409 (2013)
H. Suto, T. Nagasawa, K. Kudo, K. Mizushima, R. Sato, Appl. Phys. Express **4**, 013003 (2011)
T. Taniguchi, Phys. Rev. B **91**, 104406 (2015)
T. Taniguchi, H. Arai, S. Tsunegi, S. Tamaru, H. Kubota, H. Imamura, Appl. Phys. Express **6**, 123003 (2013)
T. Taniguchi, J. Grollier, M.D. Stiles, Phys. Rev. Appl. **3**, 044001 (2015)
T. Taniguchi, T. Ito, S. Tsunegi, H. Kubota, Y. Utsumi, Phys. Rev. B **96**, 024406 (2017)
A.A. Thiele, Phys. Rev. Lett. **30**, 230 (1973)
W. Thomson, Proc. R. Soc. A **8**, 546 (1856)
J. Torrejon, M. Riou, F.A. Araujo, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, K. Yakushiji, A. Fukushima, H. Kubota, S. Yuasa, M.D. Stiles, J. Grollier, Nature **547**, 428 (2017)
Y. Tserkovnyak, A. Brataas, G.E.W. Bauer, Phys. Rev. Lett. **88**, 117601 (2002)

S. Tsunegi, K. Yakushiji, A. Fukushima, S. Yuasa, H. Kubota, Appl. Phys. Lett. **109**, 252402 (2016a)
S. Tsunegi, E. Grimaldi, R. Lebrun, H. Kubota, A.S. Jenkins, K. Yakushiji, A. Fukushima, P. Bortolotti, J. Grollier, S. Yuasa, V. Cros, Sci. Rep. **6**, 26849 (2016b)
S. Tsunegi, T. Taniguchi, S. Miwa, K. Nakajima, K. Yakushiji, A. Fukushima, S. Yuasa, H. Kubota, Jpn. J. Appl. Phys. **57**, 120307 (2018a)
S. Tsunegi, T. Taniguchi, K. Yakushiji, A. Fukushima, S. Yuasa, H. Kubota, Appl. Phys. Express **11**, 053001 (2018b)
S. Tsunegi, T. Taniguchi, R. Lebrun, K. Yakushiji, V. Cros, J. Grollier, A. Fukushima, S. Yuasa, H. Kubota, Sci. Rep. **8**, 13475 (2018c)
S. Tsunegi, T. Taniguchi, K. Nakajima, S. Miwa, K. Yakushiji, A. Fukushima, S. Yuasa, H. Kubota, Appl. Phys. Lett. **114**, 164101 (2019)
S. Urazhdin, V.E. Demidov, R. Cao, B. Divinskiy, V. Tyberkevych, A. Slavin, A.B. Rinkevich, S.O. Demokritov, Appl. Phys. Lett. **109**, 162402 (2016)
T. Valet, A. Fert, Phys. Rev. B **48**, 7099 (1993)
D. Verstraeten, B. Schrauwen, M. D'Haene, D. Stroobandt, Neural Netw. **20**, 391 (2007)
D. Vodenicarevic, N. Locatelli, F.A. Araujo, J. Grollier, D. Querlioz, Sci. Rep. **7**, 44772 (2017)
K. Watanabe, B. Jinnai, S. Fukami, H. Sato, H. Ohno, Nat. Commun. **9**, 663 (2018)
S. Weinberg, *The Quantum Theory of Fields*, vol. 1 (Cambridge University Press, Cambridge, 2005)
M. Weisheit, S. Fähler, A. Marty, Y. Souche, C. Poinsignon, D. Givord, Science **315**, 349 (2007)
S. Wiggins, *Introduction to Applied Nonlinear Dynamical Systems and Chaos* (Springer, New York, 1990)
J. Xiao, A. Zangwill, M.D. Stiles, Phys. Rev. B **70**, 172405 (2004)
S. Yakata, H. Kubota, Y. Suzuki, K. Yakushiji, A. Fukushima, S. Yuasa, K. Ando, J. Appl. Phys. **105**, 07D131 (2009)
K. Yakushiji, K. Noma, T. Saruya, H. Kubota, A. Fukushima, T. Nagahama, S. Yuasa, K. Ando, Appl. Phys. Express **3**, 053003 (2010)
T. Yamaguchi, N. Akashi, S. Tsunegi, H. Kubota, K. Nakajima, T. Tanigcuhi, Phys. Rev. Res. **2**, 023389 (2020)
S. Yuasa, T. Nagahama, A. Fukushima, K. Ando, Y. Suzuki, Jpn. J. Appl. Phys. **43**, L588 (2004a)
S. Yuasa, T. Nagahama, A. Fukushima, Y. Suzuki, K. Ando, Nat. Mater. **3**, 868 (2004b)
S. Zhang, P.M. Levy, J. Appl. Phys. **69**, 4786 (1991)
Y. Zhou, J. Akerman, Appl. Phys. Lett. **94**, 112503 (2009)
Y. Zhou, V. Tiberkevich, G. Consolo, E. Iacocca, B. Azzerboni, A. Slavin, J. Akerman, Phys. Rev. B **82**, 012408 (2010)

Reservoir Computing with Dipole-Coupled Nanomagnets

Hikaru Nomura, Hitoshi Kubota, and Yoshishige Suzuki

Abstract An idea to use a magnetic nano-dots array for a reservoir computing is introduced. The mechanism of how the nonlinear calculation is carried out in the magnetic system is explained by showing the simplest case with three nano-dots system. The first trial to prove calculation ability and fabrication ability of the system is demonstrated. Since the proposed reservoir computing system may utilize integration technology of the magnetic random access memory (MRAM), it possesses a possibility to realize a large-scale reservoir computing system.

1 Spin-Glass Model and Spin-Glass Reservoir Computing

The Hopfield model (Hopfield 1982) and the Boltzmann machine (Ackley et al. 1985) are well-known mathematical models of recurrent neural networks (RNN). They are based on a physical model of magnetic material with randomness (Amit and Gutfreund 1985) called a spin glass. Classical view of the spin is a rotation of the electron itself that causes magnetic moment of the atoms. In a spin glass,

H. Nomura · Y. Suzuki (✉)
Graduate School of Engineering Science, Osaka University, Machikaneyama 1-4, Toyonaka, Osaka 560-8531, Japan
e-mail: suzuki-y@mp.es.osaka-u.ac.jp

Center for Spintronics Research Network, Osaka University, Machikaneyama 1-4, Toyonaka, Osaka 560-8531, Japan

H. Nomura
e-mail: nomura@mp.es.osaka-u.ac.jp

H. Nomura
Graduate School of Engineering, Osaka University, Yamada-oka 1-1, Suita, Osaka 565-0871, Japan

H. Kubota
National Institute of Advanced Industrial Science and Technology (AIST), 1-1-1 Umezono, Tsukuba, Ibaraki 305-8568, Japan

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_15

each atom acts as a magnet and interacts with each other via a quantum mechanical exchange interaction, J. In terms of neural concepts, the direction of magnetic dipole moment of an atom can be considered as the electrochemical potential of a neuron and the interaction J as the synaptic weight. Thus, magnetic materials can be treated as models of RNN and sophisticated techniques from statistical physics can be applied to analyze RNNs. In particular, the "infinite range model" (Fig. 1), where an infinite number of atoms (neurons/nodes) with all inter-atomic exchange interactions (inter-neuron/inter-node connections), is considered, which simplifies the problem tremendously because the mean-field approximation from statistical physics can provide an exact solution for the model. In addition, the Ising spin model, in which the direction of the magnetic moment is constrained along +z- or

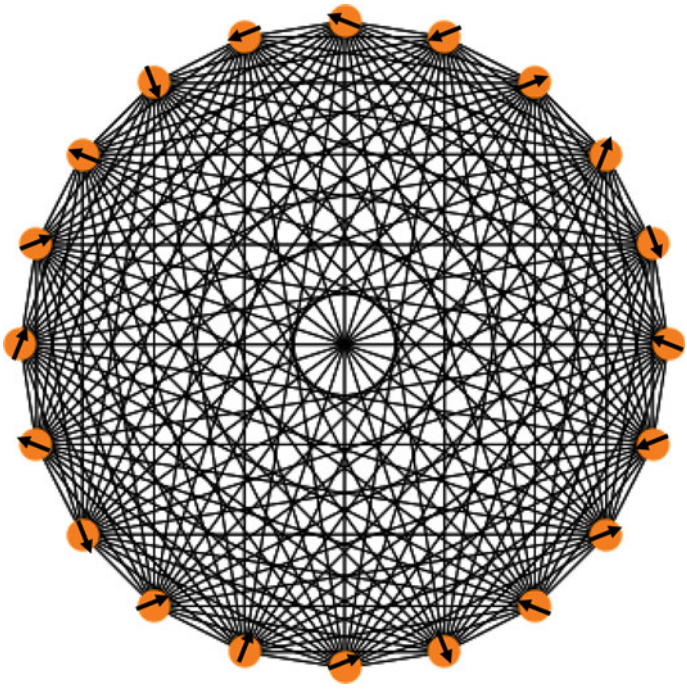


Fig. 1 Infinite range model of a spin glass. Orange disks are atoms with magnetic momentum vectors (arrows). Black lines express exchange interactions, J, between two atoms. Depending on the sign of J, connected two magnetic moments prefer parallel or antiparallel configurations. Since all atoms are connected by lines, the model is regarded as an "infinite range model." This is an idealization of the real spin glass, in which the atoms align in a solid and the exchange interactions are limited between neighboring atoms. In terms of neuromorphic computation, the atoms represent neurons (nodes), and lines represent axons/synapses complex (undirected inter-connections and synaptic weights). In the spin-glass system, because of a randomness in the atom positions, J is distributed. As a consequence, the system has many energy minima. This means that the system has many quasi-equilibrium states with complicated alignments of magnetic moments. In this chapter, we aim to use such a system as a reservoir

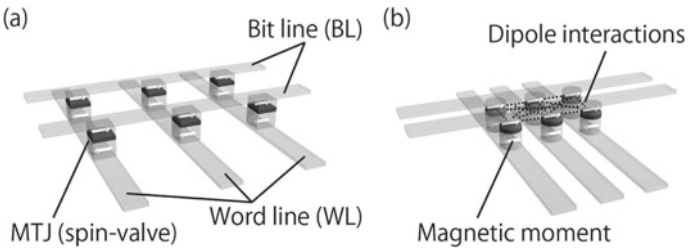


Fig. 2 **a** MRAM. The MRAM comprises magnetic tunnel junctions (MTJs) and selection transistors underneath (not shown). As an example, the MTJ can have a magnetic free layer of 20 nm × 20 nm × 2 nm. The direction of the magnetic moment in the free layer, which is illustrated by the white arrow in the black layer, corresponds to one bit of binary information. Since the resistance of the MTJ depends on the relative angle between the free layer magnetic moment and the reference layer (white arrow in the gray color layer) (Yuasa et al. 2004; Parkin et al. 2004), one may read out individual bit information electrically. **b** Magnetic reservoir made of MTJ array. The magnetic dipole interaction between the magnetic free layers allows linear/nonlinear operations among the stored information and makes the system an effective reservoir

–z-directions, can also simplify the problem without losing its essential calculation ability. These generalizations and simplifications of the spin-glass model resulted in the Hopfield model and Boltzmann machine.

Therefore, one may expect that magnetic materials can serve as good natural systems to realize RNNs. However, limited researches have been done to produce RNNs using the spin glasses. This is because individual atomic magnets cannot be easily controlled. On the other hand, recent developments in the field of electronics have made it possible to integrate large numbers of nano-sized magnets (hereafter called "nanomagnets") to construct a solid-state magnetic random access memory (MRAM) (Bhatti et al. 2017) (Fig. 2a). In MRAM, one may read out each bit information, which is stored as the direction of the magnetic moment in a cell, using the magnetoresistive effect (Yuasa et al. 2004; Parkin et al. 2004). The cell is called as a magnetic tunnel junction (MTJ), in which the resistance is dependent on the relative angle between two magnetic dipole moments in the junction, i.e., of free layer and reference layer. One may also write-in the information by applying either a current (Myers et al. 1999) or a voltage (Maruyama et al. 2009). To construct a stable memory, the interaction between the nanomagnets is removed in the MRAM. In contrast, here, we intentionally use the interaction between the nanomagnets to allow the MRAM to work as a spin-glass system and perform as an RNN. Since nanomagnets are electronically separated in the MRAM, the exchange interaction between the cells does not exist. However, magnetic dipole interaction exists that allows nanomagnets to perform linear/nonlinear calculations, as it will be explained later in this chapter (Fig. 2b). The strengths of the magnetic dipole interactions are determined from the size of magnetic dipole moments and distance between the magnets. Therefore, they cannot be modified after the device has been fabricated. This means that the system can be considered as an RNN with fixed synaptic weights.

As a consequence, the framework of reservoir computing (Jaeger and Haas 2004) is needed to perform machine learning.

As magnetic systems have rich dynamics, usage of dynamic states can increase the performance of magnetic reservoirs. For example, spin-torque oscillators offer a relatively high performance when used as a reservoir working at high frequency (see the chapters by Grollier and Tsunegi). In this chapter, we show that static magnetic systems have also shown relatively high performance by increasing the number of nanomagnets. In particular, we show that the voltage control of magnetic anisotropy (VCMA) offers a way to realize a large-scale, high-performance, and energy-efficient magnetic RNN.

2 Historical Design of the Magnetic Boolean Calculator

A Boolean operation element using dipole-coupled nanomagnets is called a nanomagnet logic (NML) (Cowburn and Welland 2000). NML comprises nanomagnets with single magnetic domain states. The binary state is defined by the polarity of the magnetic moment of the nanomagnet that is similar to MRAM. The nanomagnets communicate each other via dipole interaction. The magnetic moment of a nanomagnet aligns parallel to the stray field from surrounding nanomagnets. Since the direction of the stray field is determined by the major polarity of the surrounding magnets, the NML gate is basically a majority gate. A transmission line (Cowburn and Welland 2000), a NAND/NOR gate (Imre et al. 2006; Hikaru and Ryoichi 2011), a shift register (Hikaru et al. 2017), etc. have been previously realized (Orlov et al. 2008). Figure 3a, b shows a scanning electron microscope (SEM) image of the NML-NAND/NOR logic gate (Hikaru and Ryoichi 2011) and magnetic force microscope images of a typical operation results, respectively. Until now, NMLs have been designed to perform only Boolean operations. However, if the analog value of the magnetic moment vector is used, the NML becomes an attractive candidate for a physical reservoir.

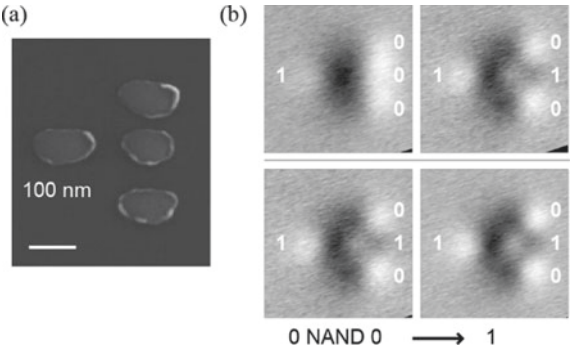


Fig. 3 **a** SEM image of the NML NAND/NOR gate. **b** Magnetic force microscope images of a typical operation of the NML NAND/NOR gate (Hikaru and Ryoichi 2011)

3 Linear and Nonlinear Calculation Using Nanomagnets

The energy (Hamiltonian: H) of a single nanomagnet with uniaxial anisotropy under an external magnetic field is written as follows:

$$\begin{cases} H = -\dfrac{1}{\left|\vec{M}\right|^2}\vec{M}^t\hat{K}\vec{M} - \mu_0\vec{M}\cdot\vec{H}_{ext} \\ \hat{K} = K_u V\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{cases}, \tag{1}$$

where $\vec{M}$ is the magnetic moment vector, $\hat{K}$ is the anisotropy tensor, $K_u > 0$ is the uniaxial anisotropy energy par unit volume, V is the volume of the nanomagnet, μ_0 is the magnetic susceptibility of vacuum, and $\vec{H}_{\text{ext}}$ is the external magnetic field vector. If we can neglect formation of a magnetic domain inside the nanomagnet, the size of the magnetic moment, $\left|\vec{M}\right|$, is the constant whereas it may change the direction. Here, we assumed a disk-like nanomagnet, where the z-axis is perpendicular to the plane of the disk (Fig. 4a). Note that $\vec{M}$ is stable if its direction minimizes H.

In Fig. 4b, the magnetic hysteresis curves for an external field parallel to $\vec{\mathbf{e}}_x + \vec{\mathbf{e}}_z$ (45° from the normal line of the disk) are shown. The solid and dotted curves show the x- and z-components of $\vec{M}$, respectively. When there is no external field, because of the uniaxial magnetic anisotropy, there are two equilibrium points, which are indicated as (i) and (i') in Fig. 4b, i.e., $\vec{M} = \pm M\vec{\mathbf{e}}_z$, where $\vec{\mathbf{e}}_z$ is the unit vector in the

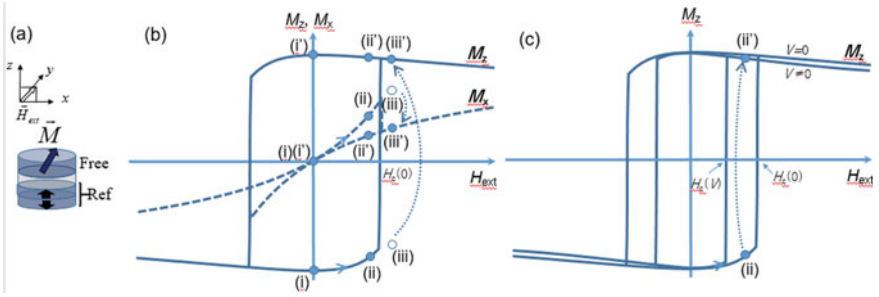


Fig. 4 **a** Schematic diagram of MTJ with perpendicular anisotropy. The free and reference layers are made of ferromagnetic materials and are separated by an insulator layer such as MgO. Employing a very thin insulator (about 1 nm), one may get a tunneling current through the insulator that is dependent on the relative angle between the two magnetic moments at both sides of the insulator layer. To avoid a stray magnetic field from the reference layer, the reference layer comprises two magnetic layers with opposite magnetic moments. **b** Magnetic moment as a function of external magnetic field, which is applied at an angle of 45° from the symmetrical axis of the MTJ. Further details are provided in the text. **c** Application of a voltage on the MTJ reduces the magnetic anisotropy energy and coercive force, H_c, through the VCMA

z-direction. Now, we trace the hysteresis starting from a state with $\vec{M} = -M\vec{\mathbf{e}}_z$. It means that the x- and z-components of $\vec{M}$ are 0 and $-M$, respectively (Fig. 4b (i)). When the external magnetic field increases, $\vec{M}$ begins to tilt toward the x-direction. Therefore, both M_{x} and M_{z} increase (Fig. 4b (ii)). Here, although the energy at point (ii') is smaller than that at point (ii), the state remains at point (ii), which corresponds to a local minimum in the Hamiltonian. When the magnetic field reaches the coercive force, H_{c}, the saddle point that separates the two minima disappears. Then, the state does not stay at (iii) but jumps to (iii'). As a consequence, M_{z} becomes positive and a small jump occurs for M_{x}. Although a small jump exists, we can assume that M_{x} is a sigmoid-like function of H_{ext}. In an array of nanomagnets, the magnetic field exerted on a nanomagnet is the sum of all dipole fields made by the other nanomagnets. Therefore, the nanomagnet calculates a sigmoid-like function of the sum of the dipole fields. Hence, the nanomagnet has a nonlinear calculation power. In contrast to M_{x}, M_{z} shows a large hysteresis. The jump from point (iii) to point (iii') corresponds to the firing of a neuron that has a certain threshold. After firing, the nanomagnet stores the information about the sign of the previous external magnetic field until another large external field is applied. Therefore, a nanomagnet has a memory function. In Fig. 4c, the effect of voltage application on the hysteresis curve is shown. The application of a voltage reduces the magnetic anisotropy and correspondingly reduces the coercive force, H_c. As a consequence, state at point (ii) may jump to point (ii'). By using these phenomena, the information about the sum of the dipole fields can be written to the nanomagnet.

The Hamiltonian of an array of nanomagnets is written as follows:

$$H_{array} = \mu_0 \left(-\sum_{i,j} \vec{M}_i^t \hat{J}_{ij} \vec{M}_j - \sum_i \vec{H}_{ext} \cdot \vec{M}_i \right)$$
$$\hat{J}_{ij} \equiv \begin{cases} \dfrac{\hat{K}_i}{\mu_0 \left| \vec{M}_i \right|^2}; \ i = j \\ \dfrac{1}{2} \hat{D}_{dipole,ij}; \ i \neq j \end{cases}, \tag{2}$$

where

$$\begin{cases} \hat{K}_i = K_{u,i} V_i \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ \hat{D}_{dipole,ij} = \dfrac{1}{4\pi r_{ij}^5} \begin{pmatrix} 3r_{x,ij}^2 - r_{ij}^2 & 3r_{x,ij}r_{y,ij} & 3r_{x,ij}r_{z,ij} \\ 3r_{x,ij}r_{y,ij} & 3r_{y,ij}^2 - r_{ij}^2 & 3r_{y,ij}r_{z,ij} \\ 3r_{x,ij}r_{z,ij} & 3r_{y,ij}r_{z,ij} & 3r_{z,ij}^2 - r_{ij}^2 \end{pmatrix} \\ \vec{r}_{ij} = \vec{r}_j - \vec{r}_i = \begin{pmatrix} r_{x,ij} \\ r_{y,ij} \\ r_{z,ij} \end{pmatrix}. \end{cases} \tag{3}$$

Here, $\vec{M}_i$, $\hat{K}_i$, $K_{u,i}$, and V_i are the magnetic moment vector, anisotropy tensor, uniaxial anisotropy energy par unit volume, and volume of the nanomagnet at site i, respectively. $\hat{D}_{dipole,ij}$, and $\vec{r}_{ij}$ are the dipole matrix, and relative position vector at site i with respect to a nanomagnet at site j, respectively.

By observing the first line in Eq. (2), one may find that the Hamiltonian is similar to that of the spin glass although it is a classical Hamiltonian with dipole interaction.

From Eq. (2), one can obtain effective magnetic field, $\vec{H}_{effct,k}$, exerted on the kth nanomagnet as follows:

$$\begin{aligned} \vec{H}_{\text{effect},k} &= -\frac{1}{\mu_0} \frac{\partial H_{\text{array}}}{\partial \vec{M}_k} \\ &= \vec{H}_{\text{ani},k} + \sum_{i \neq k} \vec{H}_{\text{dipole},ki} + \vec{H}_{\text{ext}} \end{aligned}$$

where, . (4)

$$\vec{H}_{\text{ani},k} = \frac{2}{\mu_0} \frac{\hat{K}_k}{\left|\vec{M}_k\right|^2} \vec{M}_k : \text{Anisotropy filed}$$

$$\vec{H}_{\text{dipole},ki} = \hat{D}_{\text{dipole},ki} \vec{M}_i : \text{Dipole field}$$

The effective field is a sum of anisotropy field, dipole field, and external field. The kth magnetic moment is stable if it is parallel to the effective field at the site. Therefore, it may calculate a sigmoid-like function of the sum of those fields.

A schematic diagram of a dipole magnetic field around a nanomagnet is shown in Fig. 5a. Because of the field, two nanomagnets with perpendicular magnetic moments couple antiparallel with each other when the dipole field is smaller than the anisotropy field. Under large dipole fields, the magnetic moments turn to in-plane and align parallel to the line that connects the two nanomagnets. If there are three nanomagnets on a plane, e.g., at the vertices of an equilateral triangle (see Fig. 5b), "frustration" happens. Then, for a large dipole field, all the magnetic moments turn into the plane and align circularly; this is called the vortex state.

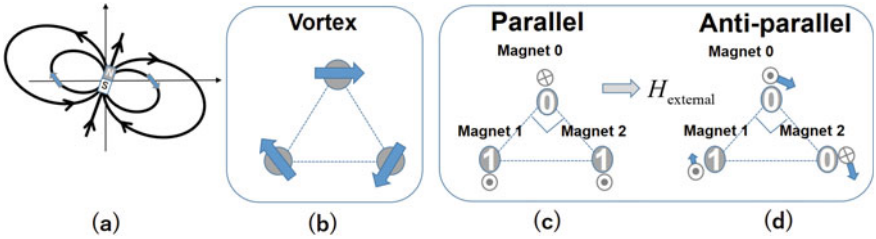


Fig. 5 **a** Magnetic dipole field distribution around a nanomagnet. **b** A system with three nanomagnets, which are placed at the vertices of an equilateral triangle. **c** A system with three nanomagnets, where the magnetic moments of magnets 1 and 2 are parallel. **d** A system with three nanomagnets, where the magnetic moments of magnets 1 and 2 are antiparallel. Here, magnets 1 and 2 are used for the input of information and magnet 0 is used for the output. The values 0 and 1 correspond to the magnetic moments in the −z− and +z-directions, respectively. The arrows beside the disk represent the magnetic moment vectors

Now, we discuss a realistic case that employs a system with three nanomagnets (see Fig. 5c, d). Three nanomagnets are placed on a plane and arranged at the vertices of an isosceles right triangle. A small magnetic field toward +x-direction is applied to break the time-reversal symmetry (see Fig. 5). For this model, we can find a stable set of $\left\{\vec{M}_i\right\}$ by finding the minima in H_{array}. Figure 5c, d shows two stable alignments of the magnetic moments. The z and in-plane components of the magnetic moments are also shown. Here, as for an example, we employ nanomagnets with radius, thickness, and saturation magnetization as 20 nm, 2 nm, and 1.3 MA/m, respectively. Magnetic anisotropy field for perpendicular magnetization without voltage application is about 3.1 mT (2.5 kA/m). Distance between magnet 1 and 2 is $50 \times \sqrt{2}$ nm. In Fig. 5c, the magnetic moments of both magnets 1 and 2 point toward the $+z$-direction, which corresponds to the "1" state. As a consequence, the magnetic moment of magnet 0 points toward the –z-direction ("0" state) because of dipole coupling. The state is obtained using the following protocol. First, the magnetic moments of magnets 0, 1, and 2 are set to point toward the $+x$-, $+z$-, and $+z$-directions, respectively. Then, the system is relaxed to find a minimum in H_{array} near the initial state. After relaxation, a voltage is applied to reduce the perpendicular anisotropy of magnet 0. The voltage is increased slowly compared with the relaxation time to keep the system at the energy minimum. After reaching zero anisotropy, the voltage is slowly removed until it becomes zero. Using this protocol, the information about the dipole field at the position of magnet 0 is written to magnet 0. The alignment shown in Fig. 5c is natural because the distances between magnets 0 and 1 and magnets 0 and 2 are smaller than that of magnets 1 and 2. Figure 5d shows a case where the magnetic moments of magnets 1 and 2 are almost antiparallel. For this case, the frustration of the system makes the perpendicular magnetic moments unstable and all the moments tilt to make a vortex-like in-plane component. The magnetic moment of magnet 0 prefers the $+z$-direction as a consequence of the delicate balance of the two dipole couplings and the external field.

Table 1 Normalized output of the system with three nanomagnets (Fig. 5b, c) and linear calculations between them. The system provides enough information to calculate AND, OR, and XOR functions using only linear calculus

M1 = a	M2 = b	$M_z = c$	$M_x = d$	AND = $\frac{1}{2}(a + b - d)$	OR = $\frac{1}{2}(a + b + d)$	XOR = d
0	0	1	0	0	0	0
0	1	−0.8	1	0	1	1
1	0	0.8	1	0	1	1
1	1	−1	0	1	1	0
M1 = a	M2 = b	$M_x = c$	AND = $\frac{1}{2}(a + b - c)$	OR = $\frac{1}{2}(a + b + c)$	XOR = c	
0	0	0	0	0	0	
0	1	1	0	1	1	
1	0	1	0	1	1	
1	1	0	1	1	0	

Without an external magnetic field, the time-reversed states are also stable and degenerate in energy. Note that, by time reversal, all the magnetic moments are reversed. With a small external magnetic field, the time-reversal symmetry is broken, and degeneracy of the time-reversed states is released. As a consequence, the time-reversed state of the antiparallel alignment of magnets 1 and 2 (Fig. 5d) is not stable. Instead, a state with reversed perpendicular moments and non-reversed in-plane moments with respect to the state shown in Fig. 5d becomes stable. The configurations of the magnetic moments of all the states are listed in Table 1. The 0 and 1 in the first and second columns represent the magnetic moments pointing toward the z- and $-z$-directions, respectively. The third and fourth columns show the normalized values of the z- and x-components of the magnetic moment of magnet 0, respectively. By using those four values, results for the AND, OR, and XOR operations can be obtained using only linear calculus as shown in the fifth, sixth, and seventh columns of Table 1, respectively. This means that the system with three nanomagnets has linear/nonlinear calculation ability and can be used as a reservoir for reservoir computation. Note that such property based on the time-reversal breaking can be obtained without an external field if a system with four nanomagnets is employed. The breaking of the time-reversal symmetry can also be incorporated by taking into account magnetization dynamics with energy dissipation.

4 First Trial of the Dipole-Coupled Nanomagnet Reservoir

As for a first trial, an array system with $8 \times 8 \times 2$ (=128) nanomagnets was tested using a micromagnetic simulation. Figure 6a, b shows the schematic diagram of reservoir computing with dipole-coupled nanomagnet reservoir and the schematic

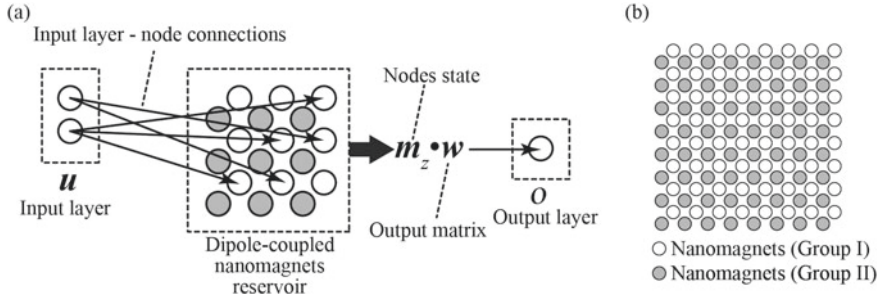


Fig. 6 **a** Schematic diagram of reservoir computing with dipole-coupled nanomagnets. **b** Schematic diagram of the top view of the reservoir with dipole-coupled nanomagnets

diagram of the top view of the reservoir with perpendicular magnetic anisotropy film, respectively. The radius and thickness of the nanomagnets were 20 nm and 0.3 nm, respectively. The gap between the nearest neighbor dots was 10 nm. The z-components of the normalized static magnetic moments of the nanomagnets were used as the state of the nodes denoted by m_z. The nodes (magnetic moments of the nanomagnets) are connected via magnetic dipole interactions. As mentioned previously, the static directions of the magnetic moments of the nanomagnets were determined from the previous direction of the magnetic moments and the dipole interaction between the entire nanomagnets in the reservoir. Therefore, the reservoir can be considered as a recurrent neural network. Some nodes were connected to the input layer u in the computer, and every nodes were connected to the output matrix w made in the computer. A dot product of the node state and the output matrix was stored in the output layer as the output of reservoir computing ($o = u \bullet w$). Binary and analog data were stored in the input layer and the output matrix, respectively.

First, when new data were set in the input layer, they were written to the input nodes (stage 1). To evaluate the system, as will be explained later, we randomly chose 48(=8 × 6) nodes from the 64 nodes of Group I (Fig. 6b) to be the input nodes and connected them to the input layer. The data in the input layer corresponding to the directions of magnetic moment were written using a writing method for MRAM. When the value in the input layer is 0/1, the direction of the magnetic moment of the input node is set to point toward the $-z/+z$-direction.

Next, the state of the node was updated. To update the node state, we changed the magnetic anisotropy of the nanomagnet using the VCMA method. After writing the data to the input nodes, a bias voltage was applied to the Group II nanomagnets (stage 2) and then removed (stage 3). By applying an appropriate bias voltage to the nanomagnets, the magnetic anisotropy of the nanomagnets disappeared. With this change of magnetic anisotropy, the magnetic moments of the nanomagnets transitioned to the next state. During this transition, linear/nonlinear operations occurred as the directions of the magnetic moments of the nanomagnets changed.

The node state in stage 3 was connected to the output matrix. The node state (i.e., the z-components of the magnetic moments of the nanomagnets) was measured using

a reading method for MRAM (e.g., a reading method using tunneling magnetoresistive effect). A dot product of the node state and the output matrix was stored in the output layer using an external hardware. Then, a bias voltage was applied to the Group I nanomagnets (stage 4) and then removed (stage 5). The state of the node was updated by repeating the processes from stage 1 to stage 5.

In order to evaluate the behavior of the magnetic reservoir, we performed micromagnetic simulations. We assumed that the nanomagnets take single-domain states and we used a single cell for a single nanomagnet in the simulations. The static direction of the magnetic moment was calculated by solving the Landau–Lifshitz–Gilbert (LLG) equation using the fourth-order Runge–Kutta method. The LLG equation at 0 K is as follows:

$$\frac{d\vec{M}_i}{dt} = -\gamma_{LL}\vec{M}_i \times \vec{H}_{eff,i} - \frac{\alpha\gamma_{LL}}{\left|\vec{M}_i\right|}\vec{M}_i \times \left(\vec{M}_i \times \vec{H}_{eff,i}\right), \tag{5}$$

α, γ_{LL}, and M_s are the damping constant, gyromagnetic ratio in the Landau–Lifshitz form, and saturation magnetization of the nanomagnets, respectively. The simulation parameters used were as follows: $\alpha = 0.5$, $\gamma_{\mathrm{LL}} = 2.211/\left(1+\alpha^2\right) = 1.7688$ m/As, and $\left|\vec{M}_i\right|/V_i = 1.3$ MA/m. Here, we employed large damping constant to shorten calculation time for the simulation. The large value of α employed here does not affect the following results since we only use static states for the RC. The typical value of the α is from 0.001 to 0.1 for metals.

We performed the NARMA10 task (Atiya and Parlos 2000) to evaluate the performance of the reservoir, which was previously to evaluate the performance of reservoirs (Appeltant et al. 2011; Nakajima et al. 2015). In the NARMA10 task, the output of step y_{s+1} is obtained from the previous inputs u_{s-k} and outputs y_{s-k} by the following equation:

$$y_{s+1} = 0.3y_s + 0.05y_s\left(\sum\nolimits_{k=0}^{9} y_{s-k}\right) + 1.5u_s u_{s-9} + 0.1. \tag{6}$$

For the input u_s, random values ranging from 0 to 0.5 were used. The input data were normalized to an 8-bit unsigned integer before storing in the input layer. In each step, the 8-bit data was written to the 48 input nodes in the reservoir. This means that each data bit was simultaneously written to six different nodes. Since only the current input was written in a single step, the reservoir is requested to have up to 10 short memory to fulfill the NARMA10 task.

We trained the output matrix using 2408 steps of training data with the least-squares method. The performance of reservoir computing was evaluated using NRMSE (normalized root-mean-square error) expressed by the following equation:

$$\mathrm{NRMSE} = \sqrt{\frac{\sum_{s=1}^{N_{\mathrm{test}}}(o_s - y_s)^2}{N_{\mathrm{test}}\sigma^2(y_s)}}, \tag{7}$$

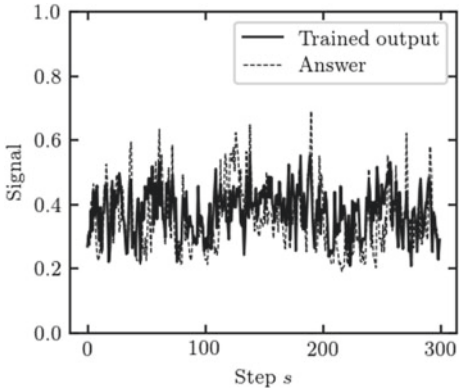


Fig. 7 Typical output result of reservoir computing with dipole-coupled nanomagnets. The solid and dotted lines show the output data of reservoir computing with trained matrix and the answer data with the NARMA10 function, respectively

where N_{test} is the number of test data, o_{s} and y_s are the output data and answer at step s, respectively, and $\sigma^2(y_k)$ is the variance of the answer.

Figure 7 shows a typical output result of reservoir computing with dipole-coupled nanomagnets. The solid line shows the output data of reservoir computing with trained matrix and the dotted line shows the answer data with the NARMA10 function. The reservoir with the nanomagnet radius of 40 nm, the thickness of 10 nm, the nearest inter-dot distance of 10 nm, K_{u} of 0 or 20 kJ/m^3, was used in Fig. 7. The output data of the trained reservoir computing show similar values as compared to the answer data. We calculated the NRMSE of the reservoir with 602 steps of the input data. The results show that the NRMSE between the output and answer data is 0.85, which is not low enough compared with those obtained by other methods. This is because the dipole-coupled nanomagnet reservoir used for the first trial (Fig. 6b) cannot hold sufficient old input values for the NARMA10 task. By changing the position of the node connected to the input layer and the updating procedure of the node state, the short-term memory capacity (Nomura et al. 2018) can be achieved with an NRMSE of 0.23 in the NARMA10 task with dipole-coupled nanomagnet reservoir (Zhu et al. 2018).

Figure 8 shows the SEM image of the prototype reservoir with dipole-coupled nanomagnets. The prototype reservoir comprises 8 × 8 nanomagnets with 2 × 2 contact holes. The distance between the nanomagnet of the prototypic reservoir was 12 nm. The micromagnetic simulations confirm that this structure can be used as a dipole-coupled nanomagnet reservoir.

5 Guiding Principle for the Future

We showed that the static magnetic moment vector in a dipole-coupled nanomagnet array behaves as a reservoir. The randomness of the inter-layer/inter-node coupling may enhance the performance of the magnetic reservoir. In the first trial, we randomly

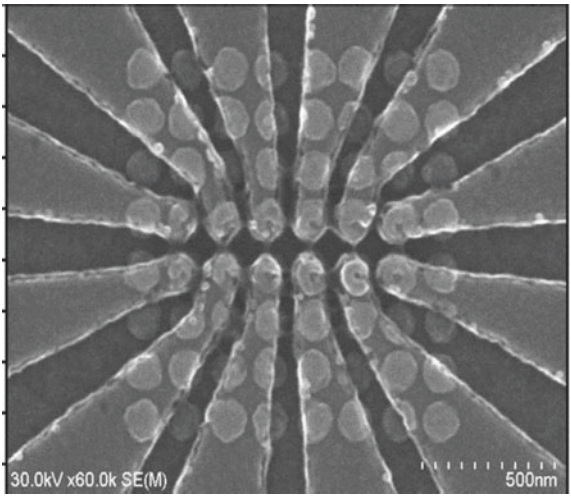


Fig. 8 SEM image of the prototype reservoir

connected the input layer and the nodes in the reservoir. However, inter-node randomness was not utilized. Such randomness can be implemented by randomly changing the size and arrangement of the nanomagnets. Depending on the arrangement of the nanomagnets, the dipole-coupled nanomagnet array can behave as a spin glass or spin ice (Jensen et al. 2018). These arrangements can be used as a reservoir.

As described above, the MRAM reading/writing technology can be used to read the node state. Moreover, the inter-node connection using magnetic dipole coupling solves the wiring problem in realizing the hardware for RNN. The dipole-coupled nanomagnet reservoir, which is capable of creating interfaces with semiconductor technology by various existing technologies, is a candidate for practical use of physical reservoir computing. With the MRAM technology, one may fabricate giga-nodes class reservoir in principle. The guiding principle, however, to obtain better performance in a very large reservoir has not yet established. Nonetheless, explorations on the human brain class (21 giga-neurons) reservoir computing are not a dream.

Acknowledgments This work was supported by the Ministry of Internal Affairs and Communications, JAPAN.

References

D.H. Ackley, G.E. Hinton, T.J. Sejnowski, A learning algorithm for boltzmann machines. Cogn. Sci. **9**(1), 147–169 (1985)

D.J. Amit, H. Gutfreund, Spin-glass models of neural networks. Phys. Rev. A **32**(2), 1007–1018 (1985)

L. Appeltant et al., Information processing using a single dynamical node as complex system. Nat. Commun. **2**, 468 (2011)

A.F. Atiya, A.G. Parlos, New results on recurrent network training: unifying the algorithms and accelerating convergence. IEEE Trans. Neural Netw. **11**(3), 697–709 (2000)
S. Bhatti et al., Spintronics based random access memory: a review. Mater. Today **20**(9), 530–548 (2017)
R.P. Cowburn, M.E. Welland, Room temperature magnetic quantum cellular automata. Science **287**(5457), 1466–1468 (2000)
N. Hikaru, N. Ryoichi, NAND/NOR logical operation of a magnetic logic gate with canted clock-field. Appl. Phys. Express **4**(1), (2011)
N. Hikaru et al., Controlling operation timing and data flow direction between nanomagnet logic elements with spatially uniform clock fields. Appl. Phys. Express **10**(12), (2017)
J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. USA-Biol. Sci. **79**(8), 2554–2558 (1982)
A. Imre et al., Majority logic gate for magnetic quantum-dot cellular automata. Science **311**(5758), 205–208 (2006)
H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)
J.H. Jensen, E. Folven, G. Tufte, Computation in artificial spin ice, in *The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)* (2018), pp. 15–22
T. Maruyama et al., Large voltage-induced magnetic anisotropy change in a few atomic layers of iron. Nat. Nanotechnol. **4**(3), 158–161 (2009)
E.B. Myers et al., Current-induced switching of domains in magnetic multilayer devices. Science **285**(5429), 867–870 (1999)
K. Nakajima et al., Information processing via physical soft body. Sci. Rep. **5**, 10487 (2015)
H. Nomura et al., Reservoir computing with dipole-coupled nanomagnets (2018), arXiv:1810.13140
A. Orlov et al., Magnetic quantum-dot cellular automata: recent developments and prospects. J. Nanoelectron. Optoelectron. **3**, 1–14 (2008)
S.S.P. Parkin et al., Giant tunnelling magnetoresistance at room temperature with MgO (100) tunnel barriers. Nat. Mater. **3**(12), 862–867 (2004)
S. Yuasa et al., Giant room-temperature magnetoresistance in single-crystal Fe/MgO/Fe magnetic tunnel junctions. Nat. Mater. **3**(12), 868–871 (2004)
L. Zhu et al., Remarkable problem-solving ability of unicellular amoeboid organism and its mechanism. R. Soc. Open Sci. **5**(12) (2018)

Part VI
Physical Implementations: Photonic Reservoir Computing

Performance Improvement of Delay-Based Photonic Reservoir Computing

Kazutaka Kanno and Atsushi Uchida

Abstract The techniques for performance improvement of delay-based reservoir computing with photonic systems are proposed and summarized. A chaos input mask signal is introduced to improve the performance of a time-series prediction task. A photonic integrated circuit is used to miniaturize the reservoir computing system. The performance of reservoir computing is compared between a single electro-optic system and a mutually coupled electro-optic system.

1 Introduction

Delay-based photonic reservoir computing has been proposed as a simplified configuration of reservoir computing (RC) (Appeltant et al. 2011). A single nonlinear device with a time-delayed feedback loop is used, and its output within the delayed loop is sampled at a constant interval. The outputs are considered virtual nodes, and the linear sum of the virtual node states are used as output. This configuration does not require a real network configuration and involves easy hardware implementation. There have been many reports on delay-based RC using photonic systems (Larger et al. 2012; Paquot et al. 2012; Duport et al. 2012; Brunner et al. 2013).

In this chapter, three techniques of the performance improvement of delay-based photonic RC are introduced and explained. The first topic provides a method to improve an input mask signal. For delay-based RC, the input mask signal is required for obtaining different node states for the same input signal. The design of the input mask signal can enhance the performance of RC. A chaos mask signal is introduced, and the performance is evaluated by a time-series prediction task. A semiconductor laser with optical feedback and injection is used as a reservoir.

K. Kanno (✉) · A. Uchida
Department of Information and Computer Sciences, Saitama University, 255 Shimo-okubo, Sakura-ku, Saitama City, Saitama 338-8570, Japan
e-mail: kkanno@mail.saitama-u.ac.jp

A. Uchida
e-mail: auchida@mail.saitama-u.ac.jp

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_16

The next topic involves miniaturizing RC systems. A photonic integrated circuit (PIC) with a semiconductor laser and an optical feedback loop is introduced, and a technique to achieve a large number of virtual node states is proposed. A technique for the use of past input signals is also proposed to perform n-step-ahead prediction.

The final topic is the use of mutually coupled systems. A variety of virtual node states can be enhanced by introducing mutually coupled systems. Asymmetric configuration of feedback delay times is crucial for improving the performance of RC. Electro-optic systems are used to show the performance comparison between single and mutually coupled systems.

2 Performance Improvement of Reservoir Computing Using an Input Chaos Mask Signal

In this section, a semiconductor laser with optical feedback and injection is used as a reservoir of delay-based RC. A chaos mask signal is introduced as an input mask signal, and the performance of a time-series prediction task is evaluated by comparison with different types of input mask signals in numerical simulations.

2.1 Scheme

The scheme of an RC using a semiconductor laser subjected to optical delayed feedback and injection is shown in Fig. 1 (Nakayama et al. 2016). RC is composed of three parts: the input layer, the reservoir, and the output layer. Input data is discretized and denoted as $u(n)$, where n is a discrete-time index, as a preprocessing step in the input layer. Input data $u(n)$ is held for a time T and a temporal mask is applied for each duration of T. The value of the mask is set to vary at each interval θ, corresponding

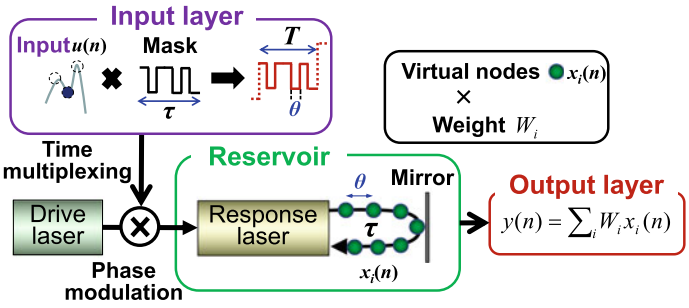


Fig. 1 Schematics of RC using a semiconductor laser (Response) with time-delayed optical feedback and injection from another semiconductor laser (Drive). Adapted with permission from (Nakayama et al. 2016). © The Optical Society

to the virtual node interval in the reservoir. The value of θ is set to be smaller than the transient response of the nonlinear system (i.e., the relaxation oscillation frequency of the laser) so that the system can generate a complex behavior. The feedback delay time τ in the reservoir is set nearly equal to the input holding time T, which is determined by the product of N nodes and node interval θ (i.e., $T = N\theta$). The input sampling time T is set to 40.0 ns, and the reservoir is composed of 400 virtual nodes ($N = 400$), with a node interval of $\theta = 0.1$ ns. The input holding time T and the feedback delay time $\tau = 40.1$ ns are slightly mismatched (i.e., $\tau = T + \theta$) for performance improvement (Paquot et al. 2012).

Two semiconductor lasers are used, referred to as the drive laser and the response laser. The drive laser is used to achieve consistent output from the response laser, as well as to convert the input signal into an optical injection signal. The dynamics of the response laser is used as the reservoir. The virtual nodes are determined from the transient response of the laser system for each interval θ within the feedback delay time τ. The reservoir is composed of virtual nodes $x_i(n)$, ($i = 1, 2,\ldots, N$) for the nth input data, and individual virtual nodes indicate different values to achieve high-dimensional space mapping.

For post-processing in the output layer, the output $y(n)$ for the nth input data is calculated as a linear combination of virtual nodes $x_i(n)$, with output weights W_i for every temporal-mask periodicity T as follows:

$$y(n) = \sum_{i=1}^{N} W_i x_i(n). \tag{1}$$

The output weights W_i are optimized by minimizing the mean-square error between the target function $\bar{y}(n)$ and the RC output $y(n)$ as follows:

$$\frac{1}{L}\sum_{n=1}^{L}(y(n) - \bar{y}(n))^2 \Rightarrow \min. \tag{2}$$

The linear least-squares method is used for learning W_i with the training data.

2.2 Numerical Model

The scheme shown in Fig. 1 uses drive and response semiconductor lasers. The dynamics of the response laser (reservoir) is calculated by using the Lang–Kobayashi equations as follows (Nakayama et al. 2016):

$$\begin{aligned}\frac{dE_r(t)}{dt} = \frac{1+i\alpha}{2}\left\{\frac{G_N(N_r(t)-N_0)}{1+\varepsilon|E_r(t)|^2} - \frac{1}{\tau_p}\right\}E_r(t) + \xi(t) \\ + \kappa E_r(t-\tau)\exp(-i\omega_r\tau) + \kappa_{\mathrm{inj}}E_d(t)\exp(i\Delta\omega t),\end{aligned} \tag{3}$$

$$\frac{dN_r(t)}{dt} = J_r - \frac{N_r(t)}{\tau_s} - \frac{G_N(N_r(t) - N_0)}{1 + \varepsilon |E_r(t)|^2} |E_r(t)|^2, \tag{4}$$

where E_d and E_r are the amplitudes of the electric field of the drive and response lasers, respectively; N_r is the carrier density of the response laser; α is the linewidth enhancement factor; G_N is the gain coefficient; N_0 is the carrier density at transparency; ε is the saturation coefficient; τ_p and τ_s are the photon and carrier lifetimes, respectively; κ is the feedback strength of the response laser; κ_{inj} is the injection strength from the drive to the response laser; $\Delta\omega = \omega_d - \omega_r$ is the angular frequency detuning between the drive and response lasers, where ω_d and ω_r are the optical angular frequencies of the drive and response lasers, respectively; J_r is the injection current of the response laser; J_{th} is the injection current at the lasing threshold, and τ is the feedback delay time. ξ(t) is the white Gaussian noise including spontaneous emissions. The parameters κ and $\Delta\omega$ are selected so that injection locking can be achieved between the drive and response lasers. These parameter values are summarized in Table 1.

The relaxation oscillation frequency of the response laser with drive injection is 5.7 GHz. The time scale of RC is determined by the inverse of the relaxation oscillation frequency (~0.18 ns) of the response laser under drive injection. The transient response is crucial for RC, and the node interval $\theta = 0.1$ ns is selected to maintain the transient response between node dynamics.

Table 1 Laser parameter values used in numerical simulations (Nakayama et al. 2016)

Symbol	Parameter	Value
α	Linewidth enhancement factor	3.0
G_N	Gain coefficient	8.4×10^{-13} m^{3}s^{-1}
N_0	Carrier density at transparency	1.4×10^{24} m^{-3}
ε	Gain saturation	2.0×10^{-23}
τ_p	Photon lifetime	1.927 ps
τ_s	Carrier lifetime	2.04 ns
κ	Feedback strength	15.53 ns^{-1}
κ_{inj}	Injection strength	12.43 ns^{-1}
ω_d	Optical angular frequency	1.23×10^{15} rad/s
$\Delta f = \Delta\omega/(2\pi)$	Frequency detuning	−4.0 GHz
τ	Feedback delay time	40.1 ns
J_r	Injection current of the response laser	1.05 J_{th}
J_{th}	Injection current at threshold	9.89×10^{32} m^{-3}s^{-1}

In the input layer, the masked input signal $M(t)$ is generated by multiplying the input data $u(n)$, the mask signal $mask(t)$ with periodicity T, and the scaling factor γ, as follows:

$$M(t) = mask(t) \times u(n) \times \gamma. \tag{5}$$

The phase of the electric field of the drive laser is modulated by using $M(t)$. After phase modulation, the electric field of the drive laser is described as follows:

$$E_d(t) = \sqrt{I_d} \exp(i\pi\, M(t)), \tag{6}$$

where E_d is the electric field of the drive laser and I_d is the light intensity of the steady state of the drive laser. This drive signal is injected into the response laser to obtain the transient outputs for RC.

2.3 Results of Chaos Mask Signal

The Santa Fe time-series prediction task (Weigend and Gershenfeld 1993) is used to evaluate the performance of RC. This task aims to perform a single-point prediction of chaotic data. These chaotic data are generated from a far-infrared laser. A total of 3000 steps are used for training, and 1000 steps are used for testing. The amplitude of the Santa Fe time-series is normalized so that the input signal $u(n)$ of the Santa Fe time-series ranges from 0 to 1.

A digital binary mask signal and an analog chaos mask signal are used to compare the performances of RC. The binary mask signal is composed of a piecewise constant function for each interval θ, with a randomly modulated binary sequence $\{-1, 1\}$ with equal probabilities (Appeltant et al. 2011; Larger et al. 2012). The standard deviation of the binary mask signal is set to 1. The chaos mask signal is generated from the response laser with optical feedback but without drive injection. The RF spectrum of the chaos mask signal is selected to be similar to that of the response laser output so that complex dynamics can be induced in the response laser cavity. The amplitude of the chaos mask signal is rescaled so that the standard deviation of the chaos mask signal is set to 1, and the mean value is set to 0.

The temporal waveforms of the masked input signals and the output of the response laser for the binary and chaos mask signals are shown in Fig. 2a and 2b, respectively. In the case of the binary mask signal in Fig. 2a, some nodes in the response signal show similar values when the mask value is constant. In contrast, the response laser generates a complex response in the case of the chaos mask signal in Fig. 2b. A variety of node states can be obtained in the complex and fast dynamical response signal in Fig. 2b.

The performance of the time-series prediction task using the binary and chaos mask signals is compared, as shown in Fig. 2c and 2d. In both cases, the prediction

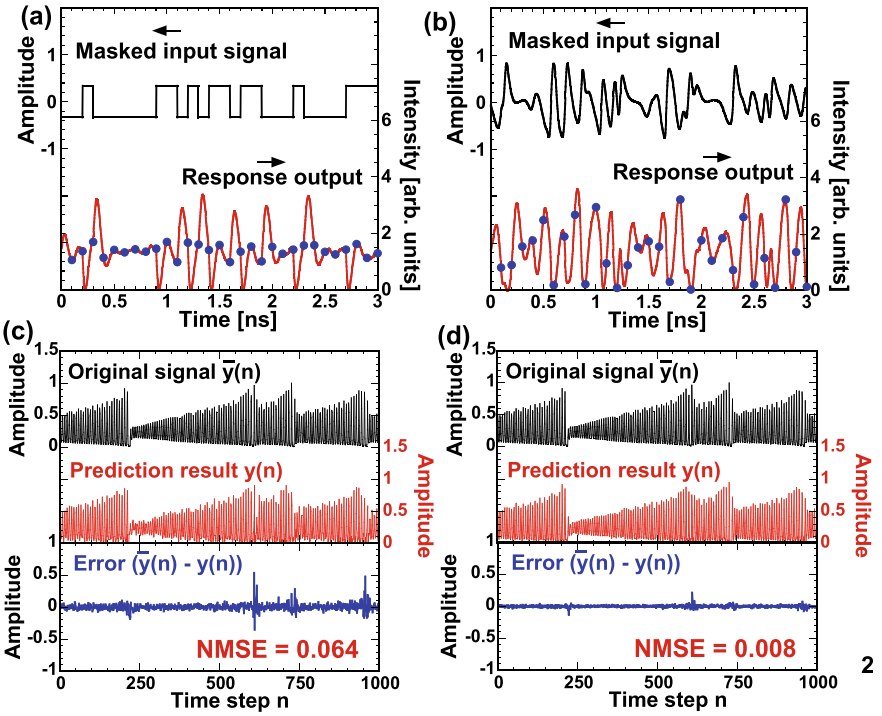


Fig. 2 Masked input signal and response laser output for **a** the binary mask signal and **b** the chaos mask signal for the first input signal. Blue dots indicate the node states. Prediction results of the original signal (black curve) and the RC prediction signal (red curve) for **c** the binary mask signal and **d** the chaotic mask signal. The error signals (blue curves) between the original signal and the RC prediction signal are shown at the bottom of **c** and **d**. Adapted with permission from (Nakayama et al. 2016). © The Optical Society

results are similar to the original signals. However, smaller errors are observed in the case of the chaos mask signal in Fig. 2d. The performance of the time-series prediction task is quantitatively evaluated by using the normalized mean-square error (NMSE) as follows (Appeltant et al. 2011; Larger et al. 2012):

$$NMSE = \frac{1}{L} \frac{\sum_{n=1}^{L} (\bar{y}(n) - y(n))^2}{\mathrm{var}(\bar{y})}, \tag{7}$$

where n is the index of the input data, L is the total number of data, y is the RC output that is compared to the original value $\bar{y}$ as a target, and var represents the variance.

The NMSEs are 0.064 and 0.008 for the binary and chaos mask signals in Fig. 2c and 2d, respectively. Therefore, better performance of the time-series prediction task can be achieved by using the chaos mask signal. The variety of the node states for the chaos mask signal is larger than that for the binary mask signal. The variety of

the node states results in higher dimensional mapping of the input signal and higher performance of RC using the chaos mask signal.

2.4 Comparison of Digital and Analog Mask Signals

Several digital mask signals for RC are introduced and compared with the chaos mask signal in this subsection. The values of the digital mask signal are changed randomly with constant interval θ, similar to the binary mask signal (Appeltant et al. 2011). In addition to the binary mask, a six-level mask signal $\{\pm 1.0, \pm 0.6, \pm 0.3\}$ and a random-level mask signal $\{-1 \sim +1\}$ are also used for comparison.

The performance of the time-series prediction task is shown in Fig. 3a with respect to the feedback strength κ of the response laser for the digital and chaos mask signals. Consistent output (reproducible output with respect to the same input signal (Uchida et al. 2004)) of the response laser is achieved in the region of $0 \leq \kappa \leq 19\ \mathrm{ns}^{-1}$ under injection locking. The consistency region is estimated from the cross-correlation value of temporal waveforms between two response lasers with the same optical injection, under the condition of different initial values and different noise signals. For larger κ, consistency is not achieved as the optical frequencies are mismatched between the two lasers. The NMSEs for all the mask signals are decreased as the feedback strength is increased within the consistency region. Smaller NMSEs are obtained at the vicinity of the consistency region ($\kappa \sim 16\ \mathrm{ns}^{-1}$), which is close to the area of neutral stability of the laser dynamical system (also known as the edge of chaos). Smaller NMSEs are obtained for the chaos mask signal in comparison with that of the digital mask signal.

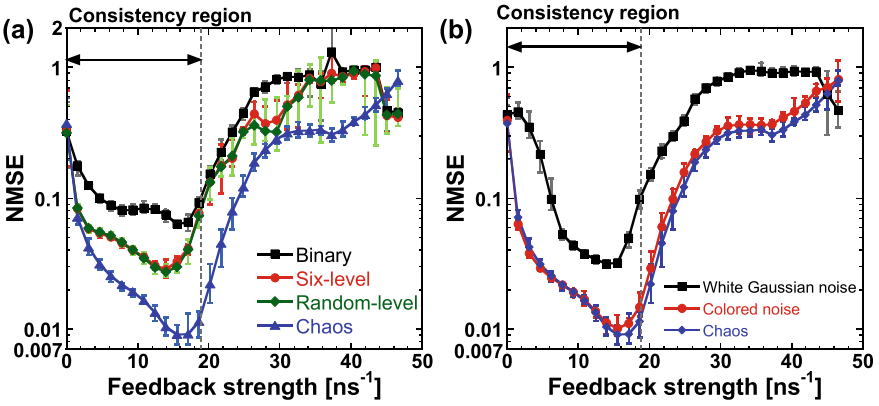


Fig. 3 NMSE of the time-series prediction task as a function of the feedback strength κ, **a** for the binary, six-level, random-level, and chaos mask signals, **b** for the white Gaussian noise, colored-noise, and chaos mask signals. Adapted with permission from (Nakayama et al. 2016). © The Optical Society

Analog noise mask signals (white Gaussian noise and colored-noise signals) are also introduced for comparison with the chaos mask signal. White Gaussian noise is calculated from the Box–Muller transform. Colored noise is calculated from the Ornstein–Uhlenbeck process using white Gaussian noise and a low-pass filter in numerical simulations (Fox et al. 1988). The cut-off frequency of the colored-noise mask signal is set near the relaxation oscillation frequency of the response lasers under optical injection (5.70 GHz), to enhance the resonance between the mask signal and the laser dynamical response. The amplitudes of these noise mask signals are rescaled so that the standard deviation of the mask signal is set to 1, and the mean value is set to 0.

The performance of the time-series prediction task is shown in Fig. 3b with respect to the feedback strength κ of the response laser for the analog mask signals. The consistency region is the same as seen in Fig. 3a ($0 \leq \kappa \leq 19$ ns^{-1}), where small NMSEs are obtained. The minimum error is obtained at the vicinity of the consistency region. The NMSEs for the white Gaussian noise mask are larger than those for the colored-noise and chaos mask signals. However, the NMSEs for the colored-noise mask signal are very similar to those for the chaos mask signal in Fig. 3b. This result indicates that the colored-noise mask signal is as effective as the chaos mask signal in inducing a complex dynamical response for RC. Complex analog mask signals would be suitable for RC input temporal masks. It is speculated that the analog property of the chaos and colored-noise mask signals could be useful for generating smooth and complex temporal waveforms of the response laser output, compared with the digital mask signals, which consist of square waveforms.

To clarify the conditions of the analog mask signals for better RC performance, the characteristic frequency of the mask signals is changed, and the performance of the time-series prediction task is investigated. It is found that the performance of RC can be improved by using the chaos mask signal with the peak frequency near the relaxation oscillation frequency of the response laser (Nakayama et al. 2016).

These findings showing the superiority of the chaos mask signal have been confirmed by experiment, as described in Kuriki et al. (2018).

3 Miniaturization of Reservoir Computing with a Photonic Integrated Circuit

Photonic integrated circuits (PICs) are good candidates for implementing RC at the millimeter scale (Takano et al. 2018). In this section, a method is proposed for implementing delay-based RC using a PIC with a semiconductor laser and a short external cavity in experiment. A time-series prediction task is used to evaluate the performance of RC. A method is introduced for enhancing the performance of the n-step-ahead prediction task using past input signals.

3.1 Scheme

The structure of the PIC for a reservoir is shown in Fig. 4a. The PIC consists of a distributed-feedback (DFB) semiconductor laser, a semiconductor optical amplifier (SOA), a phase modulator (PM), a passive waveguide, and an external mirror for optical feedback (Takano et al. 2018). The DFB laser with the external mirror (referred to as the external cavity) corresponds to the reservoir with the time-delayed feedback loop. The lengths of the DFB laser, SOA, and PM are 0.3, 0.44, and 0.3 mm, respectively. The structures of the DFB laser, SOA, and PM are based on ridge-waveguide-type structures. The optical output port is placed on the left side of the DFB laser and coated with an anti-reflection (AR) coating. It is connected to an optical fiber through a lens. The modulation signal is injected into the optical output port on the left side of the DFB laser. The output signal from the DFB laser (right side) is reflected with the use of an external mirror. The external mirror is constructed

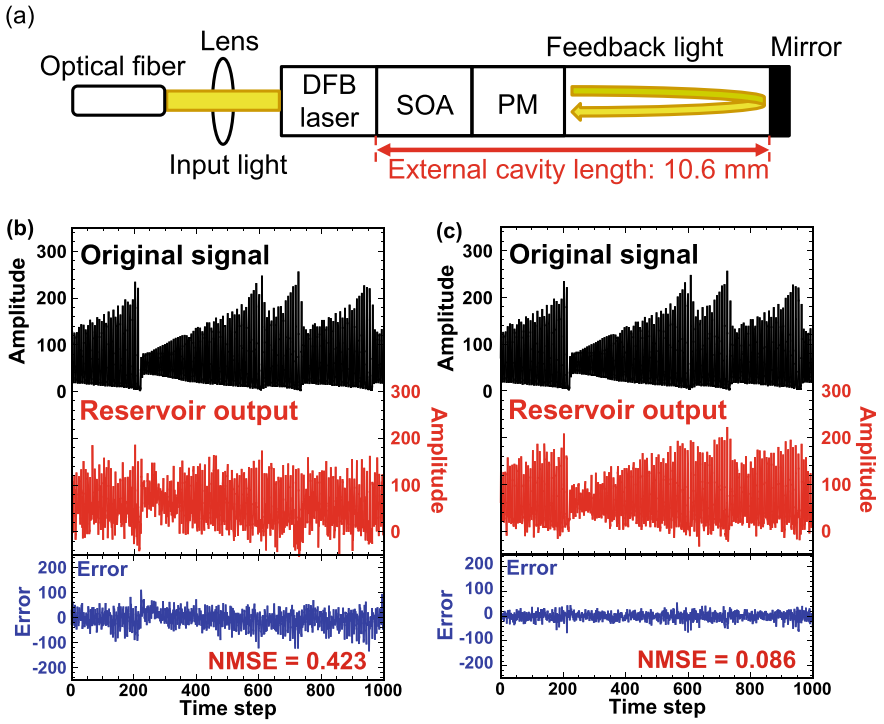


Fig. 4 **a** Structure of photonic integrated circuit. DFB laser, distributed-feedback semiconductor laser; PM, phase modulator; SOA, semiconductor optical amplifier. **b**, **c** Experimental results of time-series prediction task for different node intervals θ and different number of delay times k used as virtual nodes. **b** $\theta = 40$ ps, $k = 1$, and **c** $\theta = 10$ ps, $k = 5$. The numbers of virtual nodes are **b** $N = 6$ and **c** $N = 124$, respectively. The mask interval is set to $\theta_M = 40$ ps. Adapted with permission from (Takano et al. 2018). © The Optical Society

with a high-reflection (HR) coating on the right edge of the PIC. The external cavity length between the right facet of the DFB laser and the external mirror is 10.6 mm. This length corresponds to a round-trip delay time of $\tau = 254$ ps in the external cavity, and the external cavity frequency is 3.9 GHz (refractive index = 3.6). The strength of the optical feedback light is adjusted by the SOA injection current.

A semiconductor laser is used to feed the modulation signal into the PIC reservoir. The phase of the semiconductor laser is modulated by a phase modulator with the modulation signal and injected into the PIC. Injection locking is achieved between the optical wavelengths of the semiconductor laser and the DFB laser in the PIC. The temporal waveform of the DFB laser in the PIC is detected using a photodetector and amplified by an electric amplifier. This electronic signal is sampled using a digital oscilloscope (the sampling frequency of 100 Giga-samples/s) at N nodes at every node interval θ ($\tau \approx N\theta$) in experiment. The sampled outputs are treated as virtual node states and are used to calculate the output signal. The output signal is obtained from the sum of the weighted values of all the virtual node states. The learning of the weight values is conducted using the linear least-squares method with the use of training data, as shown in Eq. (2).

Two methods are proposed to increase the number of virtual nodes in the PIC. The first method is the reduction of the node interval. In the previous method of delay-based RC, the mask interval θ_M and the node interval θ are matched (Appeltant et al. 2011; Larger et al. 2012). However, the node interval θ can be decreased because an analog temporally varying waveform with continuous variation is used to represent the virtual node states. The second method uses the signal output based on the use of multiple delay times. A temporally varying waveform with $k\tau$ multiple delay times (k is a positive integer) of the feedback loop can be used as the set of virtual node states for a single input signal. The configuration of the PIC is unchanged with a single delay loop τ, and a virtual network from a temporal waveform with length $k\tau$ is constructed. The number of virtual nodes N can be increased with an increase in k.

3.2 Experimental Results of Time-Series Prediction Task

The Santa Fe time-series prediction task is used for performance evaluation, as described in Sect. 2.3. A random binary mask signal is applied to the input signal for the PIC reservoir. The experimental results of the time-series prediction task are shown in Fig. 4b and 4c when the node interval and the number of multiple delay times are changed. The node intervals θ are set to 40 and 10 ps, respectively, and the number of multiple delay times k is set to 1 and 5, respectively, for Fig. 4b and 4c. These parameter settings correspond to the number of virtual nodes of $N = 6$ and 124, respectively. Prediction is not successful in Fig. 4b, and the NMSE is 0.423. On the contrary, a significant improvement in the prediction errors is exhibited in Fig. 4c. A small NMSE value of 0.086 is obtained in Fig. 4c, and the reservoir output closely matches the original signal.

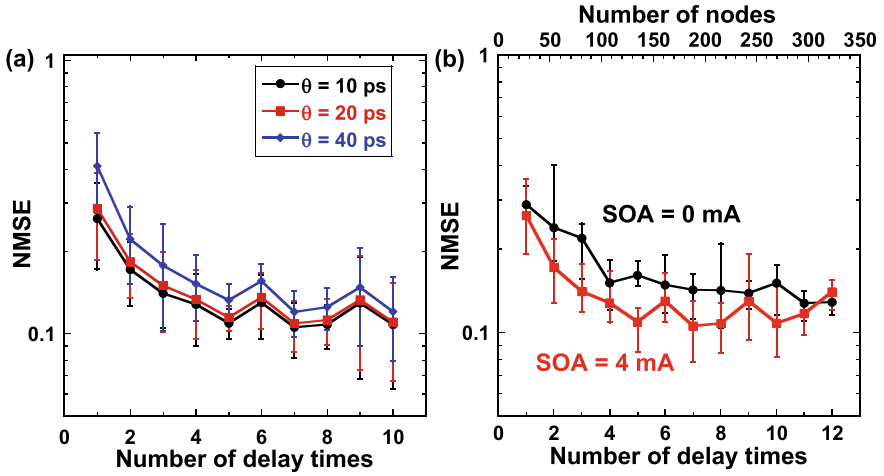


Fig. 5 **a** Prediction errors (NMSEs) as a function of **a** the number of the multiple delay times k. The node intervals θ are set to 10 ps (black circles), 20 ps (red squares), and 40 ps (blue diamonds). **b** NMSEs with (red curve) and without (black curve) optical feedback in the PIC as a function of the number of the multiple delay times k. The SOA injection currents with and without the optical feedback are, respectively, set to 4 and 0 mA. The node interval is set to $\theta = 10$ ps. Adapted with permission from (Takano et al. 2018). © The Optical Society

The prediction errors are investigated when the number of multiple delay times k is changed, as shown in Fig. 5a. The prediction error decreases with increases in k and saturates at $k = 5$. The increase in k is effective for improving the prediction error. Different node intervals are used at $\theta = 10$, 20, and 40 ps that yield similar curves in Fig. 5a. Smaller errors are obtained for smaller node intervals. The best NMSE result of 0.109 is obtained with $k = 5$ and $\theta = 10$ ps.

Next, the prediction errors are investigated for the PICs with and without optical feedback. The comparison of the variations of NMSEs is shown in Fig. 5b with and without optical feedback in the PIC for different SOA injection currents as a function of the number of multiple delay times k. The node interval is set to $\theta = 10$ ps. Smaller NMSE values are obtained in the presence of optical feedback for most of the cases associated with different k. The optical feedback can enhance the memory effect (the ability to hold past results in the reservoir), and it plays a crucial role in improving the errors of the time-series prediction task. Saturation in the values of NMSEs occurs at $k = 5$ with optical feedback, as shown in Fig. 5b. The number of nodes is large enough to perform the time-series prediction task for $k \geq 5$.

3.3 N-*Step-Ahead Prediction Task*

A method for enhancing the memory effect in the PIC reservoir is proposed in this subsection. Past input signals are introduced to the current input signal to improve the memory. A weighted sum of the past input signals is included in the current input signal $U(t)$ as follows:

$$U(t) = \sum_{i=0}^{P} g_i \, m_i(t) \, u(n-i), \tag{8}$$

$$g_i = 1 - \frac{i}{P+1}, \tag{9}$$

where g_i is the weight for the ith past input signal of $u(n-i)$, and $m_i(t)$ is the mask signal for $u(n-i)$. Input signal $u(n)$ evolves in discrete-time n and $u(t)$ is a piecewise constant function of continuous time t: $u(t) = u(n)$, $nT \leq t \leq (n+1)T$, where $T = k\tau$ is the mask length. The input mask $m_i(t) = m_i(t+T)$ is a periodic function of period T and is piecewise constant over the mask interval θ_M, where the constant values are randomly chosen from -1 or 1 (i.e., a binary mask signal).

The current input signal $U(t)$ is constructed so that the effect of the past input signal $u(n-i)$ decreases linearly with an increase in i, indicating fading past input signals. The number of past inputs P is changed, and the NMSE of the time-series prediction task is evaluated. The number of multiple delay times for the virtual nodes is set to $k = 5$, and the node interval is set to $\theta = 10$ ps.

The more difficult task of n-step-ahead time-series prediction ($n > 1$) is introduced in this subsection. The NMSE is shown in Fig. 6a as a function of past input signals P for the n-step-ahead prediction task, using the PIC with optical feedback. The NMSE decreases from $P = 0$ to 6 for the prediction step of $n = 2$ and 3 and is saturated for larger values of P. On the contrary, the NMSE is almost constant for $n = 1$. While the prediction task becomes more difficult for larger values of n, the inclusion of several past input data improves the NMSE.

Next, the NMSE is measured as a function of prediction step n for different numbers of past input P. These results obtained with optical feedback are shown in Fig. 6b. The NMSE values fluctuate periodically in Fig. 6b. This periodic oscillation becomes small when P is large, and the prediction error increases with an increase in n. In fact, this periodic curve corresponds to the auto-correlation function of the original chaotic input data used for the time-series prediction. It is found that n-step-ahead prediction for a large value of n is more difficult because of the sensitive dependence on the initial conditions of the chaotic input data. The inclusion of past input signals is effective in assisting the memory effect of the PIC with a short external cavity, such that n-step-ahead prediction can be successful.

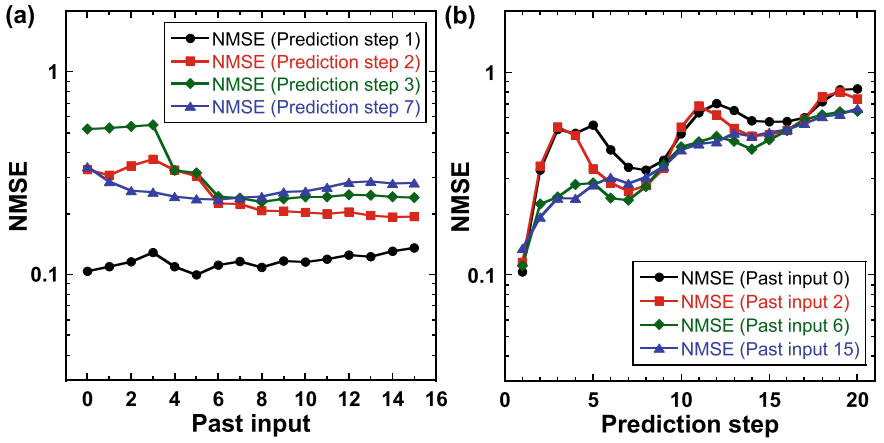


Fig. 6 **a** NMSE as a function of the number of past input signals P with optical feedback for different prediction steps n. The prediction steps include $n = 1$ (black circles), $n = 2$ (red squares), $n = 3$ (green diamonds), and $n = 7$ (blue triangles). **b** NMSE as a function of the prediction step n with optical feedback for different past input signals P. The numbers of past input signals include $P = 0$ (black circles), $P = 2$ (red squares), $P = 6$ (green diamonds), and $P = 15$ (blue triangles). Adapted with permission from (Takano et al. 2018). © The Optical Society

4 Mutually Coupled Electro-Optic System

4.1 RC Based on Electro-Optic Feedback System

An electro-optic system is another implementation for delay-based photonic RC. The schematic diagram of the electro-optic system is shown in Fig. 7a (Larger et al. 2012; Paquot et al. 2012; Larger and Dudley 2010). The system is composed of a semiconductor laser (LD), an electro-optic intensity modulator (Mach–Zehnder modulator, MZM), an optical fiber for delayed feedback (τ is the delay time), a photodetector (PD), an electronic amplifier (AMP), and an electronic bandpass filter (BPF). The MZM is used to provide a nonlinear transformation and the optical fiber

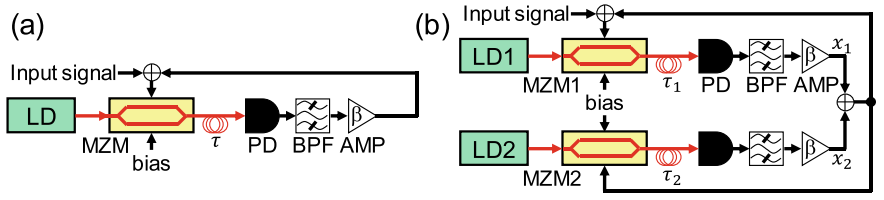


Fig. 7 **a** Schematics of the single electro-optic system and **b** mutually coupled electro-optic system. LD, semiconductor laser; MZM, Mach–Zehnder modulator; PD, photodetector; BPF, bandpass filter; and AMP, electronic amplifier. The input signal for RC is injected into MZM. Adapted with permission from Tezuka et al. (2016). © 2016 The Japan Society of Applied Physics

is used to implement a time-delayed feedback loop as the reservoir. The optical output of the laser is injected into the MZM. The optical signal that passes through the MZM is delayed by the optical fiber. The delayed signal is transformed to an electronic signal by the PD. The electronic signal is fed back to the MZM after the signal passes through the BPF and the AMP.

The MZM provides a nonlinear transfer function ($\cos^2(\cdot)$ function). Various dynamics can be observed in the electro-optic system by changing the feedback strength, which can be controlled by the detection power at the PD of the MZM. For RC, the electro-optic system operates in a steady state at a sufficiently small feedback strength to obtain a consistent response of the system outputs with respect to the same input signal (Uchida et al. 2004).

4.2 Scheme for Mutually Coupled Electro-Optic System

In this section, the RC scheme based on a mutually coupled electro-optic system is demonstrated (Tezuka et al. 2016) and compared with the single electro-optic system. The schematic diagram of the mutually coupled electro-optic system is shown in Fig. 7b. The outputs of two MZMs are detected by photodetectors and amplifiers with bandpass filters after they are delayed with delay times τ_1 and τ_2 via optical fibers. The converted signals are added electrically, and the added signal is fed back to the bias voltage of the two MZMs. The feedback delay times are set to be asymmetric ($\tau_1 \neq \tau_2$) to produce complex response outputs. The two MZMs and the two delayed feedback loops are used as the reservoir.

The mutually coupled electro-optic system shown in Fig. 7b can be modeled by the following delay differential equations (Tezuka et al. 2016; Murphy et al. 2010):

$$\tau_L \frac{dx_1(t)}{dt} = -\left(1 + \frac{\tau_L}{\tau_H}\right) x_1(t) - y_1(t) + \beta \cos^2[\kappa x_1(t - \tau_1) + (1 - \kappa) x_2(t - \tau_2) + \phi_1 + M(t)] + \xi_1(t), \tag{10}$$

$$\tau_H \frac{dy_1(t)}{dt} = x_1(t), \tag{11}$$

$$\tau_L \frac{dx_2(t)}{dt} = -\left(1 + \frac{\tau_L}{\tau_H}\right) x_2(t) - y_2(t) + \beta \cos^2[\kappa x_1(t - \tau_1) + (1 - \kappa) x_2(t - \tau_2) + \phi_2] + \xi_2(t), \tag{12}$$

$$\tau_H \frac{dy_2(t)}{dt} = x_2(t), \tag{13}$$

Table 2 Parameter values for the mutually coupled electro-optic system in numerical simulations (Tezuka et al. 2016)

Symbol	Parameter	Value
τ_L	Time constant for low cut-off frequency of BPF	32 μs
τ_H	Time constant for high cut-off frequency of BPF	0.0016 μs
τ_1	Delay time for MZM1	3.6 μs
τ_2	Delay time for MZM2	9.8 μs
θ	Node interval	0.2 μs
N	Number of virtual nodes	50
β	Coupling and feedback strength	1.0
κ	Coupling ratio between x_1 and x_2	0.5
$\phi_{1,2}$	Bias points of MZM1 and MZM2	-0.25π

where $x_{1,2}$ are the normalized outputs of the BPFs; τ_L and τ_H are time constants corresponding to the low and high cut-off frequencies ($1/(2\pi\tau_L) = 5$ kHz and $1/(2\pi\tau_H) = 100$ MHz) of the BPFs, respectively; β is the dimensionless coupling and feedback strength; $\phi_{1,2}$ represents the bias points of the MZM1 and MZM2, respectively; κ is the coupling ratio between x_1 and x_2; $\tau_{1,2}$ are the delay times for the outputs of MZM1 and MZM2, respectively; $M(t)$ is the preprocessed input signal injected into the reservoir (see Eq. (5) in Sect. 2.2); $\xi_{1,2}$ are the normalized white Gaussian noise with the properties $\xi_{1,2}(t) = 0$ and $\xi_{1,2}(t)\xi_{1,2}(t-\tau) = \delta(t-t_0)$, where $\langle\cdot\rangle$ denotes the ensemble average; and $\delta(t)$ is Dirac's delta function. The outputs of MZM1 and MZM2 are fed back to the two MZMs with the delay times τ_1 and τ_2, and they are coupled before being injected in the amplifiers. The parameter values in Eqs. (10)–(13) are given in Table 2.

In RC based on time-delayed dynamical systems, virtual nodes are assumed in the temporal output of the dynamical system. In the mutually coupled electro-optic system, N virtual nodes are considered within the temporal output $x_1(t)$. The states of these virtual nodes are defined from $x_1(t)$, separated in time by node interval θ. The number of virtual nodes $N = 50$ and the node interval $\theta = 0.2$ μs are used. The delay time has been selected to be equal to $N\theta$ in many studies (Larger et al. 2012), because the virtual nodes are assumed within a time-delayed loop. On the other hand, the unsynchronized scheme (Paquot et al. 2012) is used in this section, where the delay time τ_2 is set to $(N-1)\theta = 9.8$ μs, instead of $N\theta$. The other delay time τ_1 is set to 3.6 μs.

An input signal and the feedback signal are added, and the mixed signal is injected into one of the MZMs (MZM1) of the mutually coupled system. Before injecting the input signal, an input mask $m(t)$ is multiplied to the input signal to obtain a rich variety of node states. The mask signal has a step waveform with the mask interval θ_{M} and a period of $T = N\theta$. The input signal with the input mask is sent to the reservoir. The output of the reservoir is defined as a weighted linear combination of the node states. The weights are optimized by the least-squares method in the training procedure.

The performance of the mutually coupled electro-optic system is compared with that of the single electro-optic system. The numerical model of the dynamics of the single system is given as (Tezuka et al. 2016; Murphy et al. 2010):

$$\tau_L \frac{dx(t)}{dt} = -\left(1 + \frac{\tau_L}{\tau_H}\right)x(t) - y(t) + \beta \cos^2[x(t-\tau) + \phi + I(t)] + \xi(t), \quad (14)$$

$$\tau_H \frac{dy(t)}{dt} = x(t). \quad (15)$$

The feedback delay time τ is set to 9.8 μs and the other parameter values in these equations are set to be the same as those in the mutually coupled electro-optic system (Table 2).

4.3 Results

A longer mask interval θ_M is useful because a low-cost arbitrary waveform generator with lower frequency bandwidth can be used in experiment for fast information processing. However, the slowly modulated mask signal would not produce a variability of the virtual node states. The loss of the variability may result in the degradation of the performance of RC. This degradation could be compensated by using the mutually coupled system, instead of using the single system to enhance complexity.

The single and mutually coupled systems are compared for input mask signals of short and long intervals. A faster mask signal with the mask interval of $\theta_{\mathrm{M}} = \theta = 0.2$ μs is used for the single electro-optic system, as shown in Fig. 8a. In contrast, a slower mask signal with the longer mask interval of $\theta_{\mathrm{M}} = 10\theta = 2.0$ μs is used for the mutually coupled electro-optic system as shown in Fig. 8b.

The Santa Fe time-series prediction task is used for performance evaluation, as described in Sect. 2.3. The results of the time-series prediction using the single and mutually coupled electro-optic systems are shown in Fig. 8c and 8d, respectively. The performance for the time-series prediction task is quantitatively evaluated, using the NMSE. For Fig. 8c and 8d, the values of the NMSE are 0.034 and 0.028, respectively. A similar performance of the time-series prediction can be obtained for the single and mutually coupled systems. It is worth noting that the mutually coupled system exhibits a prediction performance comparable to the single system, even though the slower mask signal is used for the mutually coupled system. This result indicates that the mutually coupled system provides a rich variety of virtual node states, even for using the slower mask signal. The mutually coupled system has two nonlinear elements and two time-delayed loops with different delay times, and the input signal can be transformed into more complex outputs, which generate different virtual node states.

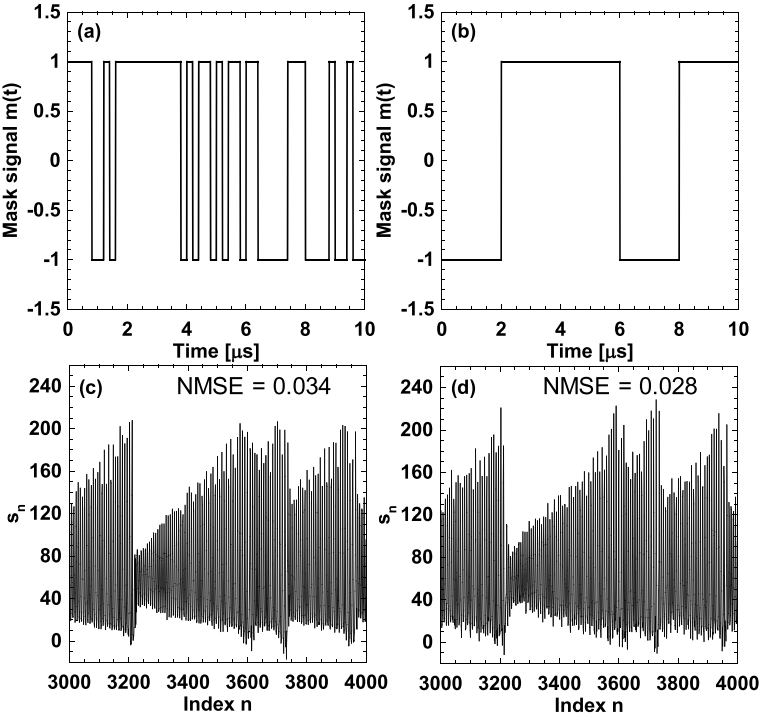


Fig. 8 **a** Fast mask signal with the step length $\theta_M = \theta = 0.2$ μs. **b** Slow mask signal with $\theta_M = 10\theta = 2.0$ μs. The value of the mask signals is randomly taken from 1 or −1(i.e., binary mask signals). The number of steps in these mask signals are $N_M = T/\theta_M = 50$ and 5, where $T = 10$ μs is the period of the mask signal, respectively. **c** Result of chaotic time-series prediction using the single feedback system with the mask signal shown in **a**. **d** Result of chaotic time-series prediction using the mutually coupled system with the mask signal shown in **b**. Adapted with permission from Tezuka et al. (2016). © 2016 The Japan Society of Applied Physics

The influence of the delay time on the prediction accuracy is investigated in the mutually coupled system with the slower mask signal. The NMSE is shown in Fig. 9a as a function of the delay time τ_1. The other delay time τ_2 is fixed at 9.8 μs. Local maxima of NMSE are obtained at $\tau_1 = 2.0$, 4.0, 6.0, 8.0, and 10.0 μs. These values of τ_1 are equal to the multiples of the mask interval $\theta_M = 2.0$ μs. Therefore, a multiple of θ_M needs to be avoided for τ_1 to obtain a high prediction accuracy. It is also found that larger local maxima of NMSE are obtained for larger τ_1. On the other hand, local minima of NMSE are obtained for $\tau_1 = 0.5$, 3.5, and 6.5 μs.

Next, the dependence of RC performance on the normalized feedback strength β is investigated. The dependence of the NMSE on β is shown in Fig. 9b for the single and mutually coupled systems. The black solid and red dashed curves represent the NMSE obtained using the mutually coupled system and the single feedback system, respectively. The slower mask signal shown in Fig. 8b is used to modulate the input signal for both cases. The NMSEs for the mutually coupled system are smaller than

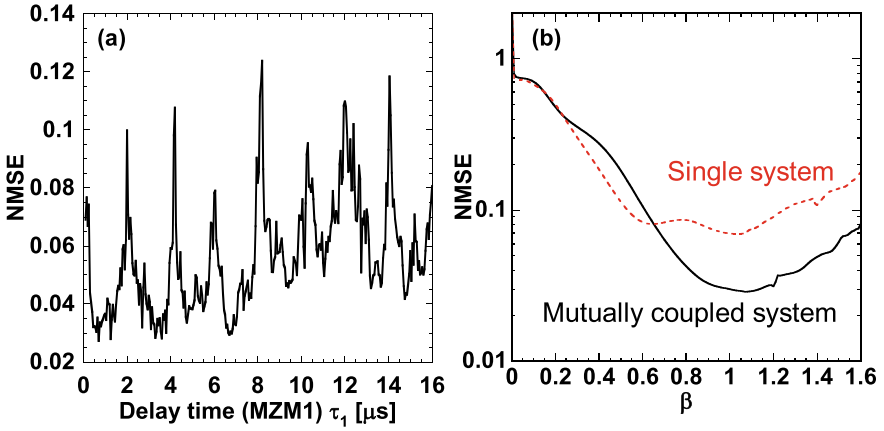


Fig. 9 **a** Prediction errors (NMSEs) as a function of the delay time τ_1 in RC using a mutually coupled system. Delay time τ_2 is fixed at 9.8 μs. **b** NMSE as a function of feedback strength β. The black solid and red dashed curves represent the mutually coupled system and the single system, respectively. The slower mask signal shown in Fig. 8b is used for preprocessing ($\theta_M = 2.0$ μs). Adapted with permission from Tezuka et al. (2016). © 2016 The Japan Society of Applied Physics

those for the single feedback system for $\beta > 0.65$. Thus, the mutually coupled system exhibits higher performance than the single system for $\beta > 0.65$. The best (smallest) NMSE of 0.028 is obtained at $\beta = 1.08$. The dynamics of the single and mutually coupled systems are stable for $\beta < 1.0$ when the input signal is not sent to the reservoir. Complex transient response can be obtained for $\beta \approx 1.0$ due to the delayed feedback and coupling when the input signal is sent to the reservoir. For a large value of $\beta > 1.0$, bifurcation appears from a stable fixed point to a periodic state. Therefore, the performance deteriorates with an increase in β for $\beta > 1.0$.

5 Discussions

The chaotic time-series prediction task is used to evaluate the performance of different reservoir systems in Sects. 2, 3, and 4. The prediction errors of these schemes are compared in this section, even though the comparison is not straightforward because the conditions of RC are different (e.g., the number of virtual nodes, type of mask signals, numerical simulations or experiments).

Compared with the schemes of the semiconductor laser with long delay (Sect. 2) and the PIC with short delay (Sect. 3), the values of the prediction errors for the latter (Fig. 4) are worse than the results for the former (Fig. 2). The main reason is the difference in the signal-to-noise ratio (SNR), the values of which are 25 and 10 dB for the numerical (Sect. 2) and experimental (Sect. 3) systems, respectively. A lower SNR is obtained in the experimental results from fast detection noise in the photodetectors and the digital oscilloscope. In fact, the experimental results of the

prediction errors of the two systems are comparable (NMSE ~ 0.1) if multiple delay times are used for the PIC reservoir (Kuriki et al. 2018; Takano et al. 2018).

In Sect. 3, a random binary mask signal is used in the PIC reservoir. An improvement in performance can be expected using a chaos mask signal instead, as described in Sect. 2.

A comparison of the semiconductor laser (Sect. 2) and the electro-optic system (Sect. 4) schemes shows that the values of the prediction errors for the latter (Fig. 8) are better than those for the former (Fig. 2c) in numerical simulations. Even in the experiment, the prediction errors for the electro-optic system are better than those of the semiconductor laser system, because a higher SNR can be obtained in the electro-optic system with low detection noise of the slow photodetectors. Therefore, the electro-optic system is more robust against detection noise. With respect to the processing speed, the delay time for the electro-optic system (microseconds) is longer than that for the semiconductor laser system (nanoseconds), and a faster RC processing speed can be obtained for the semiconductor laser system.

6 Conclusions

The techniques of the performance improvement of delay-based photonic RC are introduced and summarized in this chapter. The use of a chaotic mask signal and colored-noise mask signal results in RC performance improvement. Photonic integrated circuits are useful devices for the miniaturization of RC. The memory effect in RC can be enhanced by using past input signals. Finally, a mutually coupled system can enhance RC performance. These techniques show promise in solving more difficult tasks for RC. A combination of these techniques including the use of analog mask signals, multiple delay times, past input signals, and mutually coupled systems can further improve the performance of RC.

References

L. Appeltant, M.C. Soriano, G. Van der Sande, J. Danckaert, J. Dambre, B. Schrauwen, C.R. Mirasso, I. Fischer, Information processing using a single dynamical node as complex system. Nat. Commun. **2**, 468 (2011)

D. Brunner, M.C. Soriano, C.R. Mirasso, I. Fischer, Parallel photonic information processing at gigabyte per second data rates using transient states. Nat. Commun. **4**, 1364 (2013)

F. Duport, B. Schneider, A. Semerieri, M. Haelterman, S. Massar, All-optical reservoir computing. Opt. Express **20**(20), 22783–22795 (2012)

R.F. Fox, I.R. Gatland, R. Roy, G. Vemuri, Fast, accurate algorithm for numerical simulation of exponentially correlated colored noise. Phys. Rev. A **38**, 5938 (1988)

Y. Kuriki, J. Nakayama, K. Takano, A. Uchida, Impact of input mask signals on delay-based photonic reservoir computing with semiconductor lasers. Opt. Express **26**(5), 5777–5788 (2018)

L. Larger, J.M. Dudley, Nonlinear dynamics: optoelectronic chaos. Nature **465**, 41–42 (2010)

L. Larger, M.C. Soriano, D. Brunner, L. Appeltant, J.M. Gutierrez, L. Pesquera, C.R. Mirasso, I. Fischer, Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing. Opt. Express **20**(3), 3241–3249 (2012)

T.E. Murphy, A.B. Cohen, B. Ravoori, K.R.B. Schmitt, A.V. Setty, F. Sorrentino, C.R.S. Williams, E. Ott, R. Roy, Complex dynamics and synchronization of delayed-feedback nonlinear oscillators. Philos. Trans. R. Soc. A **368**, 343–366 (2010)

J. Nakayama, K. Kanno, A. Uchida, Laser dynamical reservoir computing with consistency: an approach of a chaos mask signal. Opt. Express **24**(8), 8679–8692 (2016)

Y. Paquot, F. Duport, A. Semerieri, J. Dambre, B. Schrauwen, M. Haelterman, S. Massar, Optoelectronic reservoir computing. Sci. Rep. **2**, 287 (2012)

K. Takano, C. Sugano, M. Inubushi, K. Yoshimura, S. Sunada, K. Kanno, A. Uchida, Compact reservoir computing with a photonic integrated circuit. Opt. Express **26**(22), 29424–29439 (2018)

M. Tezuka, K. Kanno, M. Bunsen, Reservoir computing with a slowly modulated mask signal for preprocessing using a mutually coupled optoelectronic system. Jpn. J. Appl. Phys. **55**, 08RE06 (2016)

A. Uchida, R. McAllister, R. Roy, Consistency of nonlinear system response to complex drive signals. Phys. Rev. Lett. **93**, 244102 (2004)

A.S. Weigend, N.A. Gershenfeld, Results of the time series prediction competition at the Santa Fe Institute. IEEE Int. Conf. Neural Netw. **3**, 1786 (1993)

Computing with Integrated Photonic Reservoirs

Joni Dambre, Andrew Katumba, Chonghuai Ma, Stijn Sackesyn, Floris Laporte, Matthias Freiberger, and Peter Bienstman

Abstract The idea of using photonic systems as reservoirs to perform general-purpose computing was first introduced in 2008. Since then, a wide range of systems using either discrete or integrated optical components has been explored. In this chapter, we summarise a decade of research into integrated coherent photonic reservoirs. In these systems, information is carried by the intensity and the phase of light waves. Computations emerge from the way the light propagates inside the system, and the ways in which light that travels along different paths is mixed and transformed. We discuss the computational capabilities of these reservoirs and the trade-offs that can be made to optimise them. We also discuss the technological constraints that are encountered in building such systems and the ways these reflect on their design and training. Finally, we give an overview of recent approaches to combining multiple such reservoirs into larger and computationally more powerful systems.

1 Introduction

Light is commonly used as an information carrier in telecommunication systems. Optical communication has almost entirely replaced electronics for all but very short distances, because electronic interconnect is inherently bandwidth limited and consumes far more power at high data rates. Optical telecommunication systems are now mostly held back by the fact that computation is still performed electronically. The limitations of electronics pop up at the boundaries between the optical and the electronic domain, where optical signals are converted into electronic ones and vice versa. This fact has revived the interest in pushing as much computation as possible into the optical domain. However, the devices and material systems used for integrated photonics do not match well with the typical behaviour of elementary digital building blocks such as logic gates and flipflops. For this reason, the general-purpose divide-and-conquer design methodology used for electronic digital computing systems is not portable to photonics. Also, in contrast to integrated electronics, device

J. Dambre (✉) · A. Katumba · C. Ma · S. Sackesyn · F. Laporte · M. Freiberger · P. Bienstman
Ghent University and imec, Technologiepark 126, 9052 Zwijnaarde, Ghent, Belgium
e-mail: Joni.Dambre@UGent.be

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_17

dimensions are hard coupled to the wavelength of the light that is used, which means that integrated photonics does not scale well. For this reason it is essential to find the sweet spot in the trade-off between computational power and area. Although dedicated analogue optical components are being designed for specific tasks, a design methodology for general-purpose analogue photonic components does not exist. On the positive side, the trade-off between bandwidth and power is generally better for photonic than for electronic devices, raising the hope that integrated photonic computing devices can push computation speeds even higher for digital information processing.

Around the beginning of the twenty-first-century, the principles of what is now known as *reservoir computing* showed up in several scientific publications. Echo state networks (ESN) (Jaeger 2001) and Liquid State Machines (LSM) (Maass et al. 2002) are the most commonly cited ones. What they have in common, is the fact that useful computation can be done by random dynamical systems, provided that the dynamics can be globally tuned to a useful regime: stable, but close to the edge of chaos.[1] Although the systems originally studied were simulated artificial neural networks, operating in discrete time, this fascinating idea soon stimulated other researchers to explore whether physical dynamical systems could also be used as reservoirs in this setup, giving birth to the field of *physical reservoir computing*. Photonic systems have been among the first physical reservoirs that were studied.

Two large families of photonic reservoirs exist. Time multiplexed reservoirs (addressed in other chapters of this book) use a single or very few physical nonlinear processing units. They are time-multiplexed, using delayed feedback to provide dynamics and memory (Brunner et al. 2013; Larger et al. 2012, 2017; Paquot et al. 2012). They have been shown to be capable of solving relatively complex tasks, even image classification (Hermans et al. 2015), at very high bandwidths. They are in principle scalable towards complex tasks, but this requires longer delay lines, making their implementations rather bulky. In contrast, the parallel integrated photonic reservoirs which are the topic of this chapter are much closer to the echo state networks they are based on. They are integrated, and therefore small. They are also optimised for low power and very high bandwidths. The performance of integrated photonic reservoirs has been studied numerically for networks of ring resonators (Andre et al. 2014; Mesaritakis et al. 2013, 2015; Vandoorne et al. 2011; Zhang et al. 2014), networks of SOAs (Vandoorne 2011), and experimentally with networks of delay lines and splitters (Vandoorne et al. 2014). Integrated photonic reservoirs are particularly compelling when implemented in a CMOS-compatible platform as they can take advantage of its associated benefits for technology reuse and mass production.

This chapter describes our vision, the research route we followed and the lessons we learned. We start by giving an overview of the technological hurdles that have been (and are still being) taken to obtain low-power tunable computing modules. We then highlight recent first steps towards a scalable design methodology with such modules. We will not elaborate on state-of-the-art technologies for designing and

[1] Although *edge of chaos* is the common term in dynamical systems theory, the term *edge of stability* has always felt like a more appropriate term in this context.

fabricating the components we use, since that can be found in publications. Instead we focus on the interplay between system optimisation, which starts from known good practices in unconstrained (software) reservoir computing, and the technical possibilities and constraints of the chosen implementation substrate. In this way, we hope that our path and the lessons we learned can also inspire researchers who are exploring other physical substrates and systems.

2 From Ideas to First Prototypes

Early research on integrated photonic reservoir computing aimed to mimic Echo State networks with tanh nonlinearities as closely as possible, while exploiting the potential benefits of integrated photonics: high bandwidth and low power. However, as software models, ESNs are entirely unconstrained with respect to properties like connectivity, connection weight values, neuron non-linearity or time scales. In contrast, in any physical implementation medium, limitations arise from physical and device constraints, or from fabrication issues. In addition, the often misleading near-infinite precision of software simulations is replaced by finite precision due to limited measurement accuracy and various noise sources. The very first conceptual description of a possible integrated photonic reservoir (Vandoorne and Bienstman 2007) proposed to use a regular 2D grid of Semiconductor Optical Amplifiers (SOAs) as reservoir nodes and had a "waterfall" interconnection topology (Fig. 1). Between that very first concept and the time of writing this chapter lies a decade of interdisciplinary research, combining systematic architecture and device optimisation, machine learning methods, targeted technological innovations and prototyping efforts. The result is systems that are now ready to bridge the final stretch between academic research and industrial take-up in the telecommunications sector.

2.1 Coherent Light and Planar Topologies

When using light as an information carrier, there are several options with respect to the way information is encoded into the signal. A first choice to make is to use either incoherent light or coherent light. In the first case, the information is carried only by the light intensity, represented by a positive real number. In the second case, the signal has both a magnitude (still positive) and a phase, and should therefore be modelled as a complex number. Like in our brains, the presence of *inhibition*, i.e., connections with negative weight between neurons, is crucial for computation in artificial neural networks. Early studies (unpublished) have confirmed that ESNs in which all signal values and all weights are positive perform very poorly. However, coherent light has both a phase and a magnitude. Equally, connections introduce both a phase shift and a change of magnitude (usually a loss). This implies that all operations in coherent optical reservoirs need to be modelled as complex-valued operations.

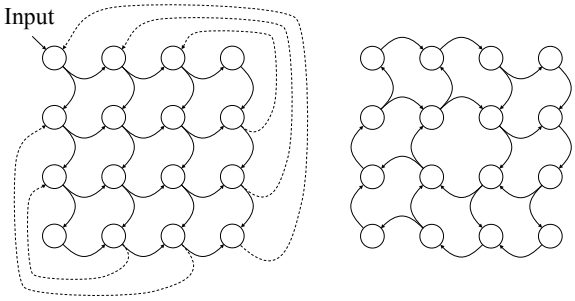


Fig. 1 Planar architectures for integrated photonic reservoirs: (left) the Waterfall topology from Vandoorne and Bienstman (2007) with the input signal inserted in the top-left node and optional feedback connection s(dashed lines) and (right) the Swirl topology that was first introduced in Vandoorne et al. (2008) and is still the basis for most recent work. No input injection is indicated here, since multiple variants have been investigated

As a consequence, even with only positive weights and positive light intensities, the addition of coherent light signals can be subtractive as well as additive w.r.t. to the magnitude of the signals. This is the main reason why coherent platforms were adopted early on for integrated photonic reservoirs.

Because of their integration on a 2D chip, integrated photonic reservoirs are also two-dimensional (planar) structures. Using photonic integrated waveguides for connecting neurons means that sharp bends and interconnect crossings cause prohibitive losses, excluding the stacking of layers of interconnect, as in electronic chips. For this reason, a purely planar connection graph is preferable. The most common approach is one in which each neuron has at most four neighbours. Initially, the input signal was injected into a single node only. After the initial concept that was based on the waterfall architecture (Vandoorne and Bienstman 2007) (Fig. 1, left), subsequent studies as well as first prototypes were based on the Swirl architecture (Vandoorne et al. 2008) (Fig. 1, right), which was designed to introduce optimal mixing of the internal state signals within the constraints of a planar structure. The connections are made by waveguides, which are passive components and therefore introduce no gain (only loss). They can be modelled as complex-valued weights with a magnitude smaller than 1 and a phase shift relative to their length. Each waveguide also has a propagation delay that is again relative to its length.

2.2 Readout and Training

In these reservoirs, a part of the light is split off at the output of each node and converted back into an electronic signal by a photodetector. This conversion of light intensity into an electrical signal non-linearly transforms each reservoir state (essentially quadratically) before linearly combining the states in the readout, trained with ridge regression.

In practice, the fabrication tolerances on the lengths of integrated waveguide connections are such that their loss and propagation delay are very close to the designed values, but the deviations of the phase shifts are so large that it is safest to consider each individual phase shift as random and different for each physical reservoir instance. In fact, due to the otherwise very regular topologies of the integrated photonic reservoirs, the variability of the phase shifts is their main source of richness. All simulation studies and design choices described in this chapter are based on performing simulations for a number of reservoirs with the same design parameters and random phases and reporting the average results.

However, there is also a downside to this uncontrollability: because of the differences between individual reservoirs, weights trained in simulation perform very poorly on the physical devices. Therefore, once they are fabricated, each individual reservoir will always need a calibration stage for training its readout. It is also not possible (or at least not within a reasonable time) to construct an accurate simulation model for individual physical reservoirs, since this would require a procedure to extract the actual values of all interconnection phases. This means that typical neural network training or optimisation approaches based on the application of gradient descent (backpropagation) cannot be used to optimise internal parameters of integrated photonic reservoirs.

2.3 Delays, Non-linearity and Power

The first integrated photonic reservoir architectures that were thoroughly studied and systematically optimised (simulation only) used a 4×4 array of Semiconductor Optical Amplifiers (SOAs) as reservoir nodes with a swirl connection structure. The input signal was initially inserted into a single node only (Vandoorne et al. 2008). The hyperparameters to be optimised for these reservoirs were the scaling and bias for the input signal, the bias current (determining the operating point of the SOAs) and the delay of the inter-node waveguides. The inter-node losses were fixed by the technology and consisted of the splitter losses, combiner losses and the waveguide losses, which were coupled to the waveguide length.

A first important conclusion from this initial work is the fact that tuning the inter-node connection delay is crucial for good performance. The importance of reservoir time scales was already pointed out in Verstraeten (2009) for simulated reservoirs (ESNs). The point of using a reservoir for computation is the fact that it naturally has memory, i.e., current state signals still depend to some extent on past inputs. This memory is weaker for inputs that lie further in the past. In order to solve a task on a given input signal (or signals), the reservoir has to be able to remember any past information from the input that is useful for the task and its bandwidth has to be high enough to respond to changes in the input signal. A physical reservoir's memory is determined by how fast signals (echo's) fade away (the losses) or are non-linearly mixed with other signals. Its bandwidth is affected by the bandwidths of all components, i.e., in photonic reservoirs: the signal generation and input modulation,

the SOAs, and the detector. Non-linear mixing occurs in the nodes (SOAs). The inter-node interconnections in integrated photonic reservoirs act as delay lines, i.e., sources of near-perfect memory. Although their loss is length dependent, it is dominated by splitter and combiner losses. For this reason, changing the inter-node delays directly impacts the memory without strongly affecting the non-linear mixing in the nodes. This means that for each task, inter-node interconnection lengths need to be tuned to match the relevant time scales in the input signal and the task.

A second conclusion was to drop the SOAs in future designs. It turns out that using SOAs nodes would increase the overall power budget above acceptable limits. They were originally introduced into the network to provide non-linearity, and because their static input-output characteristics more or less resemble the upper half of the hyperbolic tangent function that is commonly used in ESNs. It is generally understood that ESNs are in some sense universal approximators for fading memory functions, provided they can be made large enough (although some discussion exists about the precise conditions). Moving further away from ESNs by removing the non-linearity from the nodes and shifting it to the readout reduces the set of functions a reservoir can approximate. However, in that respect, the SOAs turned out to offer only limited advantages: at the optimal parameter settings for several benchmark tasks, they operated in an almost linear regime (Vandoorne 2011). In practice and for tasks that are not strongly non-linear, the non-linearity from the photodetectors alone is sufficient to achieve state-of-the-art performance (Vandoorne et al. 2014).

The two lessons above gave rise to the first actual implementation of an integrated photonic reservoir. Since the SOAs were omitted, it was a 4×4 purely passive Swirl network consisting of waveguides, splitters and combiners. For telecom tasks on bitstream signals, the lengths of the delay lines were optimised relative to the bit period. This made the design easily portable to operate at different bitrates, simply by changing the delay line lengths. Strangely enough, this also decreases the footprint of the reservoir for larger bitrates. For commonly used analogue benchmarks with much slower time scales like spoken digit recognition, the inputs were artificially speed up in order to achieve the optimal relation between signal and reservoir time scales. This first physical implementation contained only the reservoir itself. The readout was implemented in software after driving the reservoir with a long input sequence multiple times, each time measuring one reservoir state with a photodetector and storing it for further processing. The experimental results in Vandoorne et al. (2014) demonstrate that a 4×4 passive integrated photonic reservoir can yield error free performance on the header recognition task for headers up to 3-bit in length. Simulations for larger reservoirs indicated that it should be possible to go up to 8-bit headers (see Fig. 2). We additionally demonstrated that the passive integrated photonic reservoir can be used for bit-level manipulations on digital optical bit streams that could be useful for various tasks in telecommunication. Vandoorne et al. (2014) contains more information about the chip design and fabrication procedure.

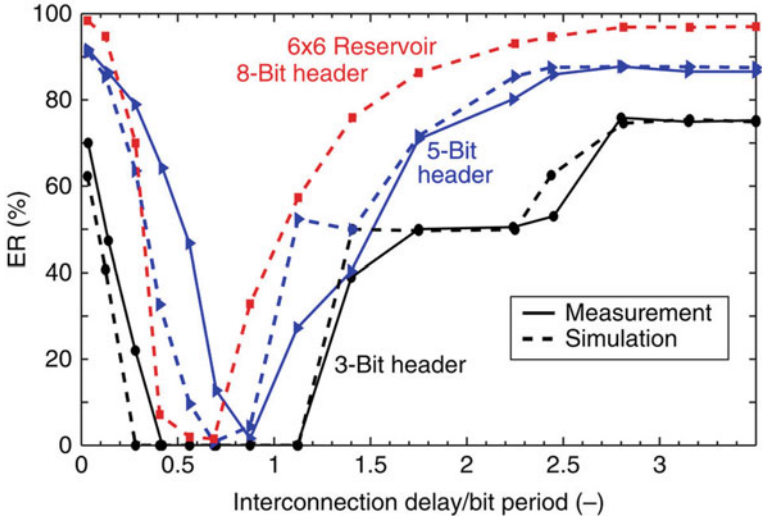


Fig. 2 Simulated and measured performance of a 4×4 swirl passive integrated photonics reservoir on the 3- and 5-bit header recognition task and simulated performance of a 6×6 reservoir on the 8-bit header recognition task (Vandoorne et al. 2014). The plots show the dependency of the error rate on the inter-node delays in the reservoir, relative to the bit period

2.4 Next Generation Reservoir Architectures

From the perspective of power consumption, the step to passive integrated reservoirs was crucial, since it removes all power consumption inside the reservoir. In addition, it allows the reservoir to be implemented in the CMOS-compatible silicon photonics technology, which drastically lowers the technological barrier for industrial take-up. However, the fact that there is no more gain inside the reservoir affects the reservoir performance: although signals still travel in the reservoir, their magnitudes now decrease very rapidly due to losses. When only a single reservoir node is driven by the input signal, the power in many nodes ends up below the detection limit (noise margin) of the photodetectors. Under these circumstances, even mildly scaling up the reservoir to address more difficult tasks makes no sense since this will not increase the number of nodes with detectable power levels. For this reason, follow-up simulation studies over the last few years have focused on designing more power-efficient reservoirs by reducing the losses or by shifting to totally different architectures. In what follows, we summarise recent progress on three lines of research.

Optimisation of the input distribution

Increasing the input power, even if this were desirable, does not help much because the losses accumulate in a multiplicative manner as light travels further from the input node. It turns out to much more effective to split the available input power budget

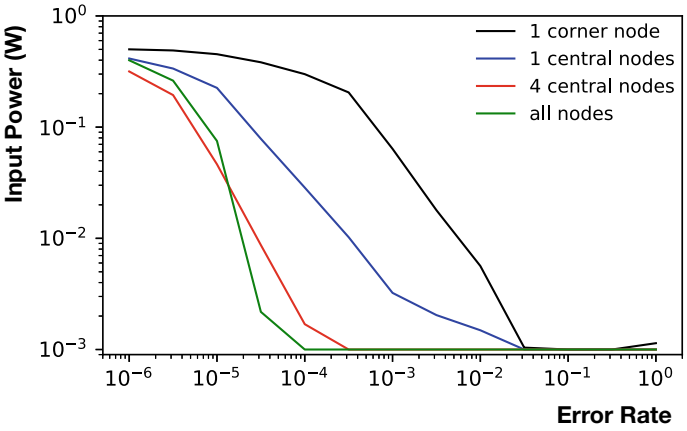


Fig. 3 Error rate for the one-bit-delayed XOR task versus total input power for different injection scenarios in a 4×4 passive Swirl reservoir. The minimum measurable error, given the number of bits used for testing, is 10^{-3} (Katumba et al. 2017). Note that the power budget only includes the power that is actually sent into the reservoir. It does not include the additional power that is consumed by the driving and receiving circuits as these have not been modelled in our simulations

across multiple input nodes (Katumba et al. 2017). Spreading the input power evenly across all nodes yields the best accuracy for a given power budget (Fig. 3 illustrates this).

However, the design and fabrication also become more difficult as the number of nodes that are driven by the input increases. For a 4×4 Swirl architecture, driving only the four central nodes is an excellent compromise between reservoir performance and technological complexity (Fig. 3). For the same accuracy, a reservoir with this input scheme needs about 2 orders of magnitude less power than when only a single node is driven to achieve almost the same performance as a reservoir for which the input power is distributed evenly across all nodes. Clearly, the outcome of this optimisation is closely linked to the size and connectivity of the reservoir. For the 16-node reservoir, the four central nodes form a single connected loop, from which power flows into the undriven boundary nodes. For larger architectures, the ideal trade-off may lie elsewhere. Ideally, for each future architecture, the input connectivity and reservoir connectivity must be jointly optimised.

Thus far, most benchmarks addressed with integrated photonic reservoirs have been relatively simple. However, scaling up to larger reservoirs in order to solve more difficult tasks only makes sense if the additional nodes actually contribute. Spreading the available input power across multiple nodes as in Katumba et al. (2017) makes this possible. However, it also makes the design more complex because the input signal needs to be coupled into more nodes. Also, driving nodes off the boundary of the planar network necessarily leads to crossings or the need for coupling the light into the nodes vertically (from the top). For this reason, it is desirable to keep the number of driven nodes as small as possible. If this can be complemented with

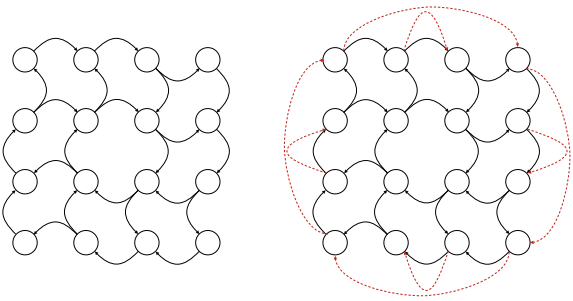


Fig. 4 Extension of swirl architecture (left) to more power-efficient four-port architecture (right): connections that were added are shown in red

technological solutions to reduce the losses, the fraction of nodes that need to be driven to achieve good performance can be further reduced. Two such approaches for loss reduction have been investigated.

Multi-mode reservoirs

The original passive reservoirs were designed to guide only a single mode. Note that the term *mode* in this context refers to the different transverse modes of the waveguide. Since part of the losses are due to light that leaks away to other modes, it could be beneficial to design a reservoir in which some of these other modes are also guided, such that the power that leaks into them is not lost. In this way, using multi-mode reservoirs can help to realise a better power balance.

The use of multi-mode rather than single-mode reservoirs was studied in simulation for passive reservoirs in which only a single node is driven (Katumba et al. 2018a). The successful application of this approach strongly hinges on the design of a novel multi-mode Y-junction with carefully tailored adiabaticity that lowers the losses at combination points in the photonic network constituting the reservoir. It turned out that, for a 36-node (6×6) reservoir, we can gain up 30% in per node power, especially for nodes that are furthest from the input point. Although the power in the boundary nodes is still lower, This extra power boost could be the difference between being below or above the noise floor at a node. In future work, additional benefits of multi-mode reservoirs will be investigated. For example, separate modes can carry different degrees of freedom, which could be beneficial for mixing the input signals.

New architectures

In the swirl architecture (repeated in Fig. 4, left), each node has at most two inputs and two outputs. However, this maximum is not reached for all nodes. The four corner nodes only have a single input and output, which means that no signal mixing occurs in them. All other boundary nodes have three ports (either only one input or only one output). This means they need asymmetrical combiner or splitter devices which suffer from modal radiation losses. These losses are inherent for non-symmetrical reciprocal splitting devices, as on average there is a 50% modal mismatch between the two input channels. By changing the architecture in such a way that each node has exactly two inputs and two outputs, these losses can be avoided. From an architectural perspective,

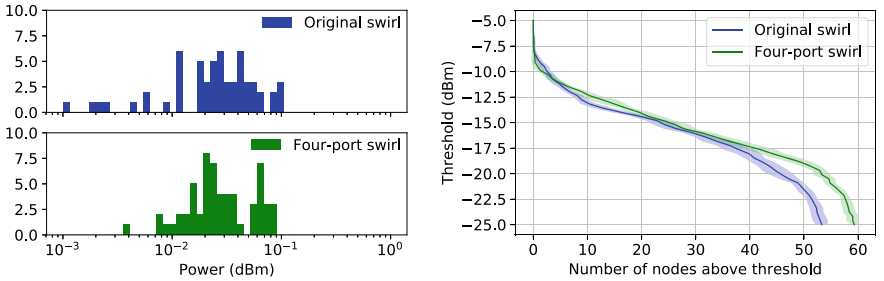


Fig. 5 Comparison between node power levels of 6×10 original swirl architecture and four-port architectures when driven with bitstream signals: (left) histograms of node powers for both architectures; (right) required detection thresholds for being able to detect a given number of nodes

the best way to do this would be to extend the architecture with so-called *wrap-around* connections at the edges. However, a wrap-around architecture is inherently not planar, which means that crossings would be needed in a planar implementation. A planar alternative was proposed in which wrap-around connections are added only for the corner nodes, and all other added boundary connections are non-crossing (Fig. 4, right). Note that the four corner nodes now effectively form a second four-node loop, similar to the inner four nodes. In Sackesyn et al. (2018), this new architecture is compared with the swirl architecture in simulations in terms of loss and in terms of performance on the equalisation of a non-linearly distorted BPSK signal. These simulations (Fig. 5) show that the four-port architecture indeed suffers less losses and thus has a better energy-efficiency as all input power at a node is redirected to one of the two output channels instead of radiating away.

3 Training Reservoirs with Integrated Optical Readouts

3.1 Motivation

To train the readout of a physical reservoir computing system, all states that used in that readout must be observable. In many cases, the reservoir is driven with the training input signal(s), the resulting reservoir states are recorded and one-shot learning through ridge regression is used to obtain the optimal readout weights. However, online learning (using recursive least squares or FORCE optimization Sussillo and Abbott 2009) is also possible. For the integrated photonic reservoirs discussed thus far, it was assumed that a photodetector would be needed for each reservoir state, followed by an AD convertor because training is usually done in the digital domain. Unfortunately, this solution would be challenging when scaling up to reservoirs with more nodes and tuned to operate at higher bandwidths, as high-speed photodetectors tend to be costly in terms of chip footprint.

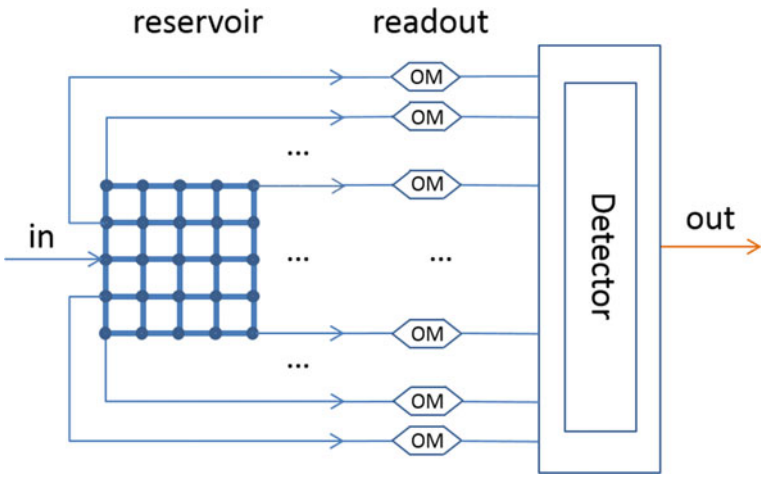


Fig. 6 Schematic of a fully optical readout. Each optical output signal is modulated by an Optical Modulator (OM) implementing the weights. The optical outputs are then sent to a photodiode where all signals are summed and then converted to a final electrical output signal

This chip area, as well as the power consumption of the photodetectors, could be avoided by implementing the readout optically. Since the aim is to reduce costs (chip area) and power consumption, the optical implementation of the readout should equally not involve power-hungry weighting components for each state signal. A straightforward tuneable optical weighting element can take the form of a reverse-biased pn-junction. An even better solution would be to use non-volatile optical weighting elements, such as the ones that are currently being developed by several groups (Abel et al. 2013; Ríos et al. 2015; Van Bilzen et al. 2015). Figure 6 illustrates the concept of a fully optical integrated readout.

From a system designer's perspective, this means we end up with a readout with quite different properties from the original one (Fig. 6). In what follows, we summarise each of these differences and their implications and discuss how they have been taken into account in the designs for the next generation integrated reservoir prototypes.

3.2 Limited Observability

As mentioned before, the fabrication tolerances of integrated waveguides are such that the propagation phase of two nominally identical waveguides could be completely different. By averaging simulations across a number of randomly selected instances, the global architectural parameters can be tuned. However it is not possible to build a simulation model that exactly matches the behaviour of individual physical reservoirs, which means that training the readout weights cannot be done

in simulation. In fact, simulation studies have shown that, fro realistic settings of the phase variability, weights trained in simulation are almost as bad as randomly initialised weights (Freiberger et al. 2017).

As long as the fabrication tolerances do not improve, this implies that device-specific training on the actual hardware will always be necessary. Thus far, no technical solutions have been found that indicate that the actual optimisation of the weight values could also be done optically, on the physical hardware, so opto-electronic conversion and digital training still remain necessary. For this reason, we maintain a single photodetector after the linear readout. However, in order to record all state signals for training, it no longer suffices to send the input sequences through the reservoir once. Remember that the photodetector's output is essentially the squared magnitude of its optical input. Obtaining the state signals implies that we need to know their magnitude, but also their phase, relative to a reference phase. It turns out that this is in fact possible with a single photodetector by driving the reservoir with the same input signals multiple times (just under three times per state signal), each time with different settings of the weights (Freiberger et al. 2018). Under ideal conditions, this non-linearity inversion procedure is exact. Each signal's magnitude is obtained by setting all weights to zero except for the observed signal. To obtain the relative phases, one signals phase is chosen as a reference and two measurements are necessary to obtain the relative phase for each of the other signals. Once this procedure has been executed, training can be done offline. Once the weights have been trained, the reservoir can operate without the need for digital processing, although we expect that re-training or tuning may be necessary at regular intervals to compensate for any causes of drift or non-stationarity in the system.

3.3 Non-linearities and Complex-Valued Regression

With an integrated optical readout, the non-linearity of the single photodetector (now the only non-linearity in the system) is applied *after* the linear combination and the accuracy of the reservoir is evaluated after the photodetector. However, ridge regression is a linear technique. In order to use it in this new setting, the desired signal values after the non-linearity have to be converted back into desired complex-valued signals before the non-linearity. After that, the optical weighted sum at the readout is now a linear combination in the complex domain, which can be optimised using complex-valued ridge regression (Hoerl and Kennard 1970).

A straightforward way to obtain complex-valued target signals is to again invert the non-linearity by taking the square root of the original target signal as magnitude and setting the desired phase to an arbitrary value (e.g., zero). Unfortunately, this procedure yields rather disappointing results for tasks that really require non-linearity. A simple example is the highly non-linear XOR task on bitstream input signals, where the desired output is again a bitstream signal, representing the Boolean XOR of the current bit and the previous bit. This task could easily be solved by the reservoirs

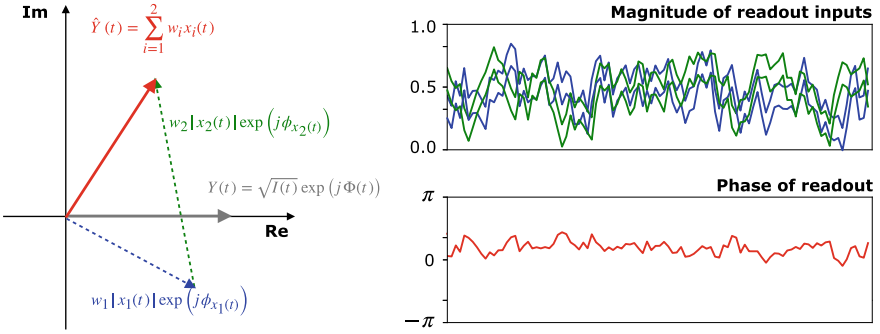


Fig. 7 Mathematics of a complex readout for two states: although each weighted input's phase is constant in time (chosen at $-30°$ and $+100°$ in the figure), the phase of the linear combination varies in time

without optical readout, i.e., with a (non-linear) photodetector for each state signal, but turns out to be much harder with a single photodetector after the readout.

One part of the reason for this is the fact that, purely from a function approximation point of view, taking a weighted sum of non-linearly transformed signals offers more non-linear richness than taking the same non-linear transformation of a weighted sum. However, it also turns out that setting targets for the complex-valued signals *before* the photodetector is far from optimal. This can be explained as follows. First, fixing the magnitude of the signal also fixes the operating point of the output non-linearity. Since the output signal can easily be rescaled and shifted in the electrical domain, this is not necessary. Second, and more importantly, fixing the phase to a single value is overly restrictive. Let's consider the simplest case of combining two states. The weights are constant in time and so is the phase for each of the state signals. The readout is therefore the sum of two complex-valued signals with magnitudes that vary in time and constant phases. Due to the properties of complex addition, the phase of this readout also varies in time, as illustrated in Fig. 7. This simple example shows why trying to enforce a constant phase to the outputs of the readout constrains the solution space unnecessarily. The phase in the complex domain is of no relevance to the signal after its transformation to the electrical domain, so it should be left unconstrained.

Since ridge regression is an optimisation technique for linear models, we cannot use it without setting targets in the complex domain. Instead, we use gradient descent to minimise the error between the electrical output signal and the complex-valued weights directly. We previously explained why using gradient descent is not possible for optimising the entire reservoir due to the uncertainty about inter-node connection phases. However, the phases in the paths between the readout weights and the photodetector are included in the signals obtained from non-linearity inversion procedure and a differentiable model for the photodetector is quite straightforward.

The replacement of ridge regression by gradient descent avoids enforcing any values for the complex-valued signal between the linear combination and the pho-

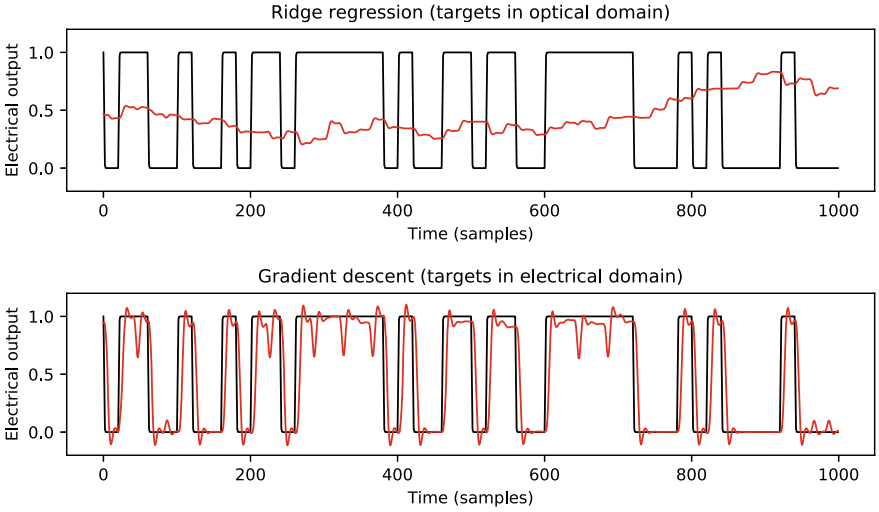


Fig. 8 Comparison of ridge regression with complex-valued targets and gradient descent for the XOR task on bitstream input (reservoir using four-port architecture of 4×4 nodes and input applied to all nodes)

todetector. The trained parameters are the magnitudes and phases of all weights (including the constant weight) as well as the bias and rescaling after the photodetector:

$$\hat{y}(t) = w_{\text{bias}} + w_{\text{scale}} \left(m_0 * e^{j\phi_0} + \sum_{i=1}^{n} m_i * e^{j\phi_i} * x_i \right)^2, \tag{1}$$

where n is the number of state signals used in the readout, w_{bias} and w_{scale} operate in the electrical domain (i.e., after the photodetector) and $m_0 \ldots m_n$ and $\phi_0 \ldots \phi_n$ are the tuneable magnitudes and phases of the optical weights. Figure 8 illustrates the striking the difference between trained outputs obtained using ridge regression with complex-valued targets and using gradient descent directly on the real-valued targets at the output of the photodetector. This example indicates that, despite initial expectations, having a single non-linearity at the readout does not dramatically affect our reservoir performance. This is mostly due to the fact that we are working in the complex domain (with coherent light).

3.4 Limited Precision

Readout weights can be implemented using different approaches. From the perspective of minimising power, there is a large difference between volatile and non-volatile technologies. Volatile approaches, like reverse-biased PN-junctions, can typically be

tuned with fine resolution, but they need power to maintain their set value. In contrast, non-volatile weight elements only consume power when their value is being set, but the physics of the current candidate technologies do not allow for high resolution. A typical example with such a limitation is a weighing element based on barium titanate (BaTiO3) (Abel et al. 2013), an integration of a transition metal oxide material. These elements are typically able to bring only 20 discrete weight levels. Clearly, this limited precision of the tuneable weights needs to be investigated since it can be expected to affect reservoir performance.

The impact of weight precision has been analysed in simulation. For a first study, we assumed the same number of levels for both the magnitudes and the phases of the weights. The readout was trained in the same way as with infinite precision and the weights were quantized only after training, using discretisation at 8, 16, 32 and 64 discrete levels (also referred to as 3-, 4-, 5- and 6-bit quantization, respectively). Figure 9 illustrates the results for a header recognition task, each time sweeping across different values for the inter-node delay (arbitrary units). We can conclude that, without taking any additional measures, the performance degenerates a lot for 3-bit quantisation. For 4 or more bits, the error rate at the optimal inter-node delay is not much higher than for the case with infinite precision but there remains some variation between the results for individual reservoirs. At 6 bits resolution, this variability disappears almost completely. These results are promising, given the fact that no measures have yet been taken to take weight quantisation into account during train-

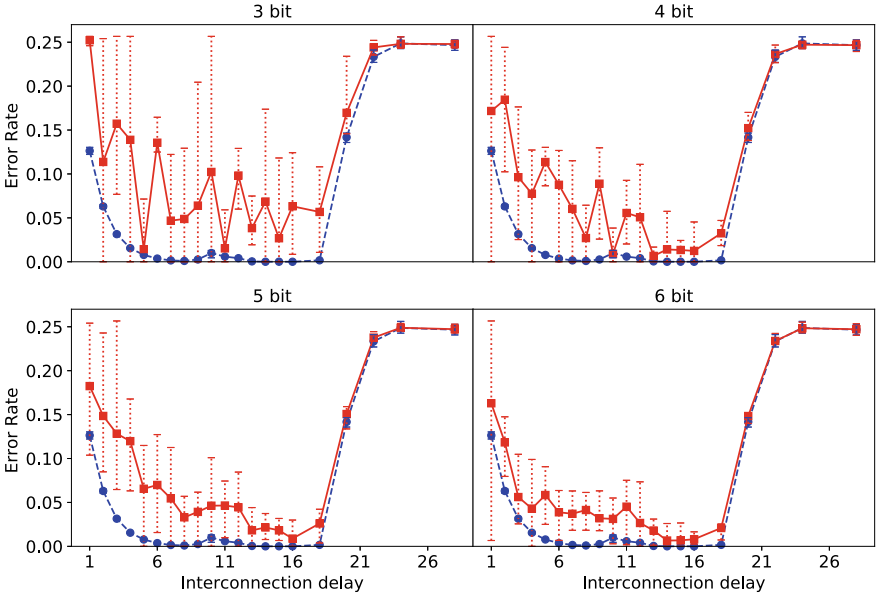


Fig. 9 Error rate when using quantised optical weights (red curves) in a header recognition task, as a function of interconnection delay. The weight resolution (in bits) is indicated at the top of each plot. For comparison, the blue curves show the case of infinite weight resolution

ing. Reduced weight precision in real-valued networks is currently being actively addressed in deep learning research. In view of introducing autonomous deep learning in embedded systems (without needing cloud connectivity) it is crucial to reduce the computational cost of inference in deep neural networks. One way to approach this is to use (extremely) reduced precision (Gupta et al. 2015; Hao et al. 2017). Results have even been proposed that go as far as using binary weights (Bengio and David 2015; Rastegari et al. 2016). By developing approaches for quantised training that are adapted to complex-valued readouts, we expect to be able to relax the precision requirements on integrated optical weights to 16 or even 8 quantisation levels (3 or 4 bits).

4 From Modules to Systems

4.1 The Future of Single Reservoir Modules

Based on simulation results, the architectural improvements mentioned in the previous sections should be sufficient to allow good reservoir performance for small to medium-sized reservoir modules. Second-generation prototypes to evaluate these claims have been designed and are being characterised at the time of writing this chapter. At the same time, more challenging applications in the telecom area are being addressed. Although a lot of optical processing is already being done in that domain, it is mostly linear. For this reason, we are focusing on non-linear tasks on optical telecom signals.

One problem that is being addressed is the restoration of optical signals that have degraded due to non-linear distortions during generation, transmission and reception phases (Djordjevic et al. 2010). Causes for this can be traced back to effects like dispersion, amplified spontaneous emission at amplification points, attenuation and reflections in fibre links, optical nonlinearities in fibres or timing jitter introduced at O/E and E/O points. Today, such restoration is typically done in the digital electronic domain using advanced DSP post-processing, but such an approach consumes a lot of power and chip real estate. Photonic reservoir computing could provide an alternative here, to undo (part of) these signal impairments already in the optical domain. Katumba et al. (2019) shows non-linear compensation in unrepeated metro links of up to 200km that outperform electrical FFE-based equalisers, and ultimately any linear compensation device. For a high-speed short-reach 40Gb/s link based on a Distributed Feedback Laser (DFB) and an Electroabsorptive Modulator (EAM), and considering an HD-FEC limit of 0.2102, the reach can be increased by almost 10km using a reservoir with only 16 nodes. These results show for the first time that integrated photonic reservoir modules can be competitive to the-state-of-the-art solutions in the telecommunications domain.

The next step is to address more complex tasks. Most often, tasks on optically encoded digital signals are performed on data encoded with a binary modulation format, but this can be extended to the much more complex case of computing directly on a PAM-4 signal, a two-bit per symbol amplitude modulation format. This is equivalent to performing Boolean computations in a 4-valued space. In Katumba et al. (2018b), initial results are given for operations on PAM-4 signals modulated at 10 GHz, which translates to a bitrate of 20 GHz. These indicate that small reservoirs like the ones discussed above cannot solve this. Even an 8×8 swirl reservoir has a symbol error rate of approximately 30%. However, the complexity of this task allows us to test how this can improve for increasing reservoir size. In this case, there was a close to linear improvement and for the largest simulated reservoir in that study (20×20 nodes), the symbol error rate had decreased to 5%. Although this is only a first study, it suggests that there is still quite a bit of room for improvement in addressing more complex tasks simply by scaling up the reservoir.

In parallel to the application-directed research, which aims to pave the road to industrial take-up, more fundamentally different reservoir architectures are also being explored. The reservoirs discussed thus far were still based on the earliest ones, which tried to follow the conventional node structure of neural networks quite closely. However, the field of physical reservoir computing has evolved a lot since the early days. In particular, as is clear from the variety of contributions in this book, the dynamical systems used as reservoirs do not have to follow the straitjacket of interconnected nodes. In particular, the inherently parallel nature of photonics allows for architectures in which the light propagates and mixes in free space. A possible design for an integrated free-space photonic reservoir consists of a photonic crystal cavity with a quarter stadium shape (Laporte et al. 2018), depicted in Fig. 10. In this design, the quarter stadium shape makes sure that an input signal gets mixed in a complicated manner (Liu et al. 2015; Sieber et al. 1993; Stöckmann and Stein 1990). The mixed light leaks out of the cavity along the connected waveguides which provide the state signals to the readout. As can be seen in the figure, the mixing of light within the cavity is very rich. In fact, it is richer than in the waveguide-based reservoirs, while at the same time using considerably less chip area. The time scales of the reservoir depend on the time-of-flight inside the cavity (the cavity dimensions) and how well the light is confined inside the cavity. The cavity in Fig. 10 was optimised for optical bitstream signals at a bitrate of around 50 Gbps. Its dimensions are $30\,\mu\text{m} \times 60\,\mu\text{m}$. In theory, cavities as small as $7\,\mu\text{m} \times 7\,\mu\text{m}$ are possible, which would allow bitrates up to 1 Tbps. Based on simulations, this photonic crystal design promises very low loss and excellent performance several benchmark telecom tasks, such as the highly non-linear XOR task and header recognition tasks, while still accepting bitrates in a wide region of operation (Laporte et al. 2018).

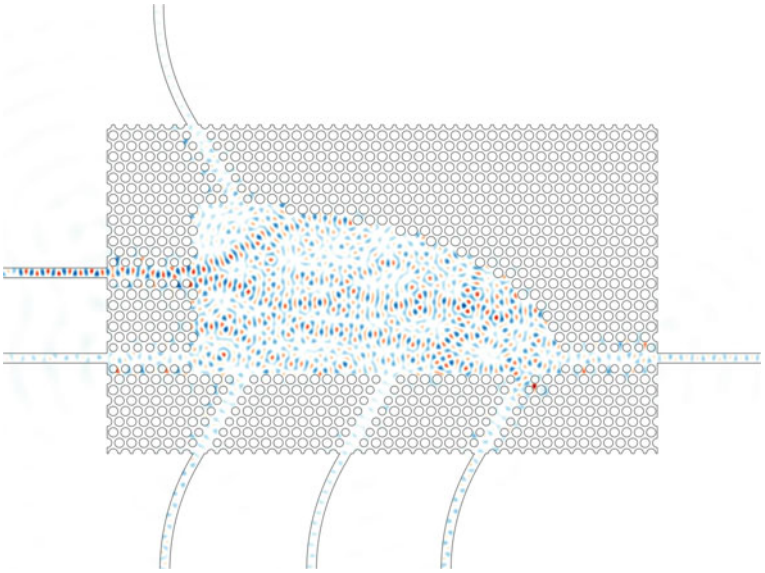


Fig. 10 Optical field profiles in a photonic crystal cavity. In the example, input is applied through a single channel. Thanks to the specific profile of the cavity's boundary, the mixing of the optical fields inside the cavity is very rich. The mixed light leaks out of the cavity along all the waveguides which are routed to a readout

4.2 Making Ensembles of Reservoir Modules

Although initial results indicate that there is still room for improvement with respect to task complexity by scaling up reservoirs, it is well known in the reservoir computing community that there are bounds to what you can achieve simply by making a reservoir larger and using more observed states in the readout. One of the main reasons for this is that, due to the interactions inside the reservoir, the states are usually highly correlated and it becomes more difficult to exploit the residual information that is added by additional states as the state space grows larger. For simulated reservoirs, such as infinite precision echo state networks, this would require an increase in the amount of training data. Physical reservoirs do not have infinite precision due to noise and measurement inaccuracies, so their performance tends to saturate even faster as a function of reservoir size. This limits the applicability of single reservoir modules.

In order to go beyond what can be achieved with a single reservoir module, more extensive training is required by building networks of interconnected reservoirs, in which training occurs only at the inter-module interfaces. Unfortunately, we are again (heavily) constrained by the fact that, due to process variability, the detailed behaviour of individual reservoirs cannot be modelled in simulation. This effectively rules

out backpropagation, the workhorse of deep learning, as a top-down optimisation approach. Instead, targets for each individual reservoir module must be explicitly defined based on the global task targets.

In machine learning, ensembling is an overarching name to achieve exactly that: increasing task performance beyond what can be achieved with a single model. In order to work well, the models in an ensemble must be different, i.e., the mistakes they make must be as uncorrelated as possible. Variation between models of the same type can be achieved by training them on different subsets of the training data or the input features. They can also be trained to correct each other's mistakes (*boosting*, Friedman et al. 2001).

In an initial simulation study of ensembling for photonic reservoirs, we consider ensembles in which each model is of the same type: an integrated 4×8 photonic four-port reservoir with interconnection delays tuned for operation at 30 Gbps. Since the interconnection phases of each reservoir are randomly chosen, each reservoir is already unique and different from the others. Since this work was performed in parallel with the research on optical readouts described in Sect. 3, it focused mostly on reservoirs with electrical readouts, i.e., with the non-linear transformation of a photodetector for each signal that is used in the readout.

We selected a number of approaches of combining four such reservoirs and compared them to the baseline case of a single 16×32 reservoir with the same architecture. The approaches we considered in this study are illustrated in Fig. 11. In the first type (single-stage ensemble), the observed states of all modules in the ensemble are concatenated into a single readout. In the second type, gradient boosting, only the first module is trained on the original task, while each consecutive module is trained to correct the mistakes of its predecessor's outputs. As a third approach, we have also evaluated the paradigm of stacking, which has already been applied in the context of reservoir computing (Keuninckx 2016; Nichele and Molund 2017). In this approach, each reservoir is trained on the original task. Only the first module is driven with the original input signal, while each consecutive module is driven with its predecessor's trained output (converted back into the optical domain. Finally, we introduce a new combination technique inspired by these approaches, which we refer to as chaining. Like in gradient boosting, each module is driven with the original input and like in stacking each readout is trained on the original task. However, from the second module onward, each readout also receives the trained output from the previous reservoir as an input. This offers a different way for the readouts to correct the mistakes of the previous reservoirs.

The architectures described above were evaluated on two benchmark tasks. The first task is the XOR task on bitstream data, but now with 3 bits delay. This means that the desired output is the Boolean XOR of the current binary symbol and that which lies three-bit periods in the past. This task has the same non-linearity requirements as the XOR task used before, but it requires more memory. The second task is the 1 sample ahead prediction Santa Fe task (Weigend and Gershenfeld 1993). Table 1 summarises the error rates obtained for these two tasks at 30 Gbps. Error rates printed in bold face indicate the best performing approach per task, error rates in italic the second best. This table shows that the single-stage ensembling systematically

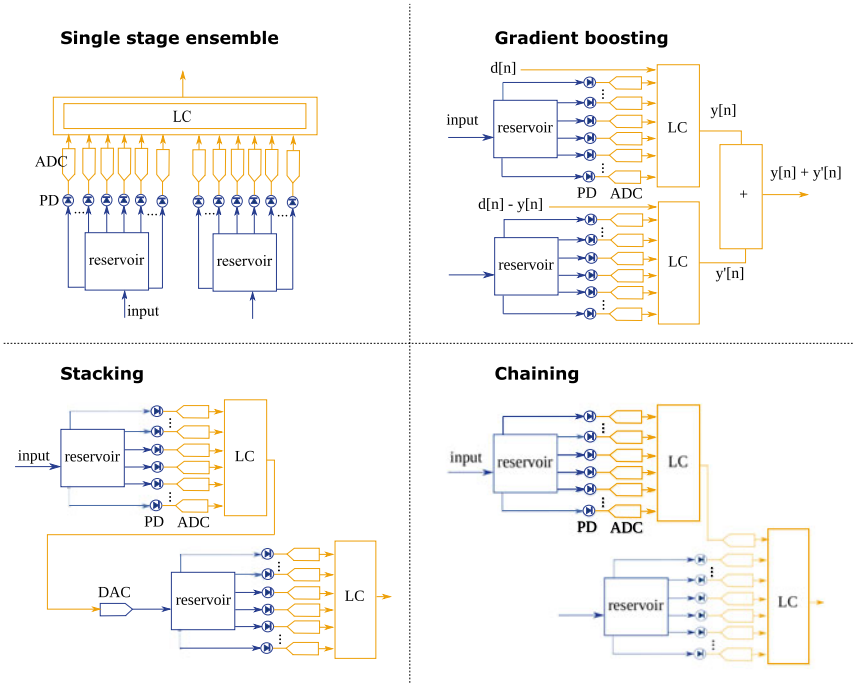


Fig. 11 Types of ensembles of reservoirs with electrical readout considered in the initial study. Components operating in the optical domain are shown in blue, components in the electrical domain in yellow

Table 1 Results for electrical training/coupling at 30 Gbps

Task	Baseline	Ensemble	Boosting	Stacking	Chaining
XOR 3-bit delay (BER)	*0.006*	**0.001**	0.041	0.222	0.038
Santa Fe (NMSE)	0.028	**0.022**	0.038	0.038	*0.027*

outperforms the baseline of a single large reservoir. This is good news, since we expect it will be technologically easier to combine multiple small reservoirs than to make a large one. The main reason for this is that optically routing power from all the nodes to the readout becomes more difficult when the dimensions of the architecture increase. Boosting and stacking are consistently worse than the baseline. This was also the case for experiments at other bandwidths (not shown). For our proposed chaining approach, conclusions are mixed. At 30Gbps, its performance is between the baseline and the ensemble for the Santa Fe task but worse than both for the XOR task. However, at other bandwidths, this gap is closed and chaining again matches the ensemble's performance.

Clearly, this is only an initial study, in which many possibilities for optimisation are yet to be explored. In view of our recent progress on integrated readouts trained with gradient descent, the next study will focus directly on those architectures. It will also cover a more extensive range of combination approaches. For example, the recent approach followed in Gallicchio and Micheli (2017), Gallicchio et al. (2018) is similar to but not the same as the stacking performed above. In those works, each reservoir module is driven with all the states of its predecessor (with untrained weights). The readout is trained on the aggregated states of all reservoirs in the ensemble. This architecture performs better than the one studied here because, as the information flows from reservoir to reservoir, each subsequent reservoir has a memory that reaches further into the past. Translated to integrated photonics technology, it would have to be simplified, e.g., by projecting a random combination of each reservoir's states back into the next reservoir and training the readout on all states in the electrical domain.

5 Conclusion and Perspectives

In this chapter we outlined our research path on integrated photonic reservoir computing, from its first steps in 2007 until today, when second-generation prototypes have been designed are being characterised and at least one application is a promising candidate for industrial take-up.

The focus in this chapter was on the impact of technological limitations on system performance and on the architectural and operational solutions we developed to end up with ever better performing systems. In view of the present results, we are more confident now than a decade ago that this technology will eventually find its way into industrial applications.

The last part of this chapter reported on first steps towards designing multi-reservoir systems. It is clear that the ensembling approaches investigated in this study do not offer sufficient improvement by themselves. Their limitation lies in the fact that all targets for training individual modules are either equal to or directly derived from the original task targets. This is not sufficient for building really powerful multi-reservoir networks. Candidate starting points for exploring architectures as well as optimising them could be based on, e.g., reinforcement learning or large-scale genetic black box optimisation approaches. However, what is really needed is a new automatic and efficient divide-and-conquer design methodology for analogue computing. Only then can reservoir computing with integrated photonic systems, as well as other novel analog computing substrates, leverage general-purpose computation in the optical domain.

References

S. Abel, T. Stferle, C. Marchiori, C. Rossel, M. Rossell, R. Erni, D. Caimi, M. Sousa, A. Chelnokov, B. Offrein, J. Fompeyrine, A strong electro-optically active lead-free ferroelectric integrated on silicon. Nat. Commun. **4**, 1671 (2013)

D. Brunner, M.C Soriano, C.R. Mirasso, I. Fischer, Parallel photonic information processing at gigabyte per second data rates using transient states. Nat. Commun. **4**, 1364 (2013)

M. Courbariaux, Y. Bengio, J.-P. David, Binaryconnect: Training deep neural networks with binary weights during propagations, in *Advances in Neural Information Processing Systems* (2015), pp. 3123–3131

I. Djordjevic, W. Ryan, B. Vasic, *Coding for Optical Channels* (Springer, US, 2010)

M.A.A. Fiers, T. Van Vaerenbergh, F. Wyffels, D. Verstraeten, B. Schrauwen, J. Dambre, P. Bienstman, Nanophotonic reservoir computing with photonic crystal cavities to generate periodic patterns. IEEE Trans. Neural Netw. Learn. Syst. **25**(2), 344–355 (2014)

M. Freiberger, A. Katumba, P. Bienstman, J. Dambre, On-chip passive photonic reservoir computing with integrated optical readout, in *2017 IEEE International Conference on Rebooting Computing (ICRC)* (2017)

M. Freiberger, A. Katumba, P. Bienstman, J. Dambre, Training passive photonic reservoirs with integrated optical readout. IEEE Trans. Neural Netw. Learn. Syst. 1–11 (2018)

J. Friedman, T. Hastie, R. Tibshirani, *The Elements of Statistical Learning*, Springer Series in Statistics, vol. 1. (Springer, New York, 2001)

C. Gallicchio, A. Micheli, Deep echo state network (deepesn): a brief survey (2017), arXiv:1712.04323

C. Gallicchio, A. Micheli, L. Pedrelli, Design of deep echo state networks. Neural Netw. **108**, 33–47 (2018)

S. Gupta, A. Agrawal, K. Gopalakrishnan, P. Narayanan, Deep learning with limited numerical precision, in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning, ICML'15*, vol. 37 (2015), pp. 1737–1746, JMLR.org

M. Hermans, M.C. Soriano, J. Dambre, P. Bienstman, I. Fischer, Photonic delay systems as machine learning implementations. J. Mach. Learn. Res. **16**, 2081–2097 (2015)

A. Hoerl, R. Kennard, Ridge regression: biased estimation for nonorthogonal problems. Technometrics **12**(1), 55–67 (1970)

H. Jaeger, The echo state approach to analysing and training recurrent neural networks–with an Erratum note 1. Technical report, GMD148, Bonn, Germany: German (2001), pp. 1–47

A. Katumba, M. Freiberger, P. Bienstman, J. Dambre, A multiple-input strategy to efficient integrated photonic reservoir computing. Cogn. Comput. **4**, 1–8 (2017)

A. Katumba, J. Heyvaert, B. Schneider, S. Uvin, J. Dambre, P. Bienstman, Low-loss photonic reservoir computing with multimode photonic integrated circuits. Sci. Rep. **8**(1) (2018a)

A. Katumba, M. Freiberger, F. Laporte, A. Lugnan, S. Sackesyn, C. Ma, J. Dambre, P. Bienstman, Neuromorphic computing based on silicon photonics and reservoir computing. IEEE J. Sel. Top. Quantum Electron. **24**, 6 (2018b)

A. Katumba, X. Yin, J. Dambre, P. Bienstman, A neuromorphic silicon photonics nonlinear equalizer for optical communications with intensity modulation and direct detection. J. Light. Technol. (2019)

L. Keuninckx, Electronic systems as an experimental testbed to study nonlinear delay dynamics. PhD thesis, Vrije Universiteit Brussel (2016)

F. Laporte, A. Katumba, J. Dambre, P. Bienstman, Numerical demonstration of neuromorphic computing with photonic crystal cavities. Opt. Express **26**(7), 7955–7964 (2018)

L. Larger, M.C. Soriano, D. Brunner, L. Appeltant, J.M. Gutiérrez, L. Pesquera, C.R. Mirasso, I. Fischer, Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing. Opt. Express **20**(3), 3241–3249 (2012)

L. Larger, A. Baylón-Fuentes, R. Martinenghi, V.S. Udaltsov, Y.K. Chembo, M. Jacquot, High-speed photonic reservoir computing using a time-delay-based architecture: million words per second classification. Phys. Rev. X **7**(1), 011015 (2017)
H. Li, S. De, X. Zheng, C. Studer, H. Samet, T. Goldstein, Training quantized nets: A deeper understanding, in *NIPS* (2017)
C. Liu, R.E.C. Van Der Wel, N. Rotenberg, L. Kuipers, T.F. Krauss, A. Di Falco, A. Fratalocchi, Triggering extreme events at the nanoscale in photonic seas. Nat. Phys. **11**(4), 358–363 (2015)
W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. **2560**, 2531–2560 (2002)
C. Mesaritakis, V. Papataxiarhis, D. Syvridis, Micro ring resonators as building blocks for an all-optical high-speed reservoir-computing bit-pattern-recognition system, in *JOSA B*, October 2013 (2013)
C. Mesaritakis, A. Kapsalis, D. Syvridis, All-optical reservoir computing system based on InGaAsP ring resonators for high-speed identification and optical routing in optical networks, vol. 9370, 2 (2015), p. 937033
S. Nichele, A. Molund, Deep reservoir computing using cellular automata (2017), arXiv:1703.02806
Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, S. Massar, Optoelectronic reservoir computing. Sci. Rep. **2**, 287, 2 (2012)
M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, Xnor-net: Imagenet classification using binary convolutional neural networks, in *European Conference on Computer Vision* (Springer, 2016), pp. 525–542
C. Ríos, M. Stegmaier, P. Hosseini, D. Wang, T. Scherer, C. Wright, H. Bhaskaran, W. Pernice, Integrated all-photonic non-volatile multi-level memory. Nat. Photonics **9**(11), 725–732 (2015)
S. Sackesyn, C. Ma, J. Dambre, P. Bienstman, An enhanced architecture for silicon photonic reservoir computing, in *Cognitive Computing 2018 - Merging Concepts with Hardware* (2018), pp. 1–2
M. Sieber, U. Smilansky, S.C. Creagh, R.G. Littlejohn, Non-generic spectral statistics in the quantized stadium billiard. J. Phys. A: Math. Gen. **26**(22), 6217 (1993)
H.-J. Stöckmann, J. Stein, Quantum chaos in billiards studied by microwave absorption. Phys. Rev. Lett. **64**, 2215–2218 (1990)
D. Sussillo, L.F. Abbott, Generating coherent patterns of activity from chaotic neural networks. Neuron **63**(4), 544–557, 8 (2009)
B. Van Bilzen, P. Homm, L. Dillemans, C. Su, M. Menghini, M. Sousa, C. Marchiori, L. Zhang, J. Seo, J. Locquet, Production of vo 2 thin films through post-deposition annealing of v 2 o 3 and vo x films. Thin Solid Films **591**, 143–148 (2015)
K. Vandoorne, Photonic reservoir computing with a network of coupled semiconductor optical amplifiers. PhD thesis, Ghent University (2011)
K. Vandoorne, P. Bienstman, A photonic implementation of reservoir computing, in *2007 IEEE/LEOS Symposium Benelux Chapter Proceedings* (2007), pp. 195–198
K. Vandoorne, W. Dierckx, B. Schrauwen, D. Verstraeten, R. Baets, P. Bienstman, J. Van Campenhout, Toward optical signal processing using photonic reservoir computing. Opt. Express **16**(15), 11182–11192 (2008)
K. Vandoorne, J. Dambre, D. Verstraeten, B. Schrauwen, P. Bienstman, Parallel reservoir computing using optical amplifiers. IEEE Trans. Neural Netw. **22**(9), 1469–1481, 9 (2011)
K. Vandoorne, P. Mechet, T. Van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, P. Bienstman, Experimental demonstration of reservoir computing on a silicon photonics chip. Nat. Communi. **5**, 3541, 1 (2014)
D. Verstraeten, Reservoir computing: computation with dynamical systems. PhD thesis, Ghent University (2009)
A.S. Weigend, N.A. Gershenfeld, Results of the time series prediction competition at the santa fe institute, in *IEEE International Conference on Neural Networks* (IEEE, 1993), pp. 1786–1793
H. Zhang, X. Feng, B. Li, Y. Wang, K. Cui, F. Liu, W. Dou, Integrated photonic reservoir computing based on hierarchical time-multiplexing structure. Opt. Express **22**(25), 31356–31370, 12 (2014)

Part VII
Physical Implementations: Quantum Reservoir Computing

Quantum Reservoir Computing: A Reservoir Approach Toward Quantum Machine Learning on Near-Term Quantum Devices

Keisuke Fujii and Kohei Nakajima

Abstract Quantum systems have an exponentially large degree of freedom in the number of particles and hence provide a rich dynamics that could not be simulated on conventional computers. Quantum reservoir computing is an approach to use such a complex and rich dynamics on the quantum systems as it is for temporal machine learning. In this chapter, we explain quantum reservoir computing and related approaches, quantum extreme learning machine and quantum circuit learning, starting from a pedagogical introduction to quantum mechanics and machine learning. All these quantum machine learning approaches are experimentally feasible and effective on the state-of-the-art quantum devices.

1 Introduction

Over the past several decades, we have enjoyed exponential growth of computational power, namely, Moore's law. Nowadays even smart phone or tablet PC is much more powerful than super computers in 1980s. People are still seeking more computational power, especially for artificial intelligence (machine learning), chemical and material simulations, and forecasting complex phenomena like economics, weather and climate. In addition to improving computational power of conventional computers, i.e., more Moore's law, a new generation of computing paradigm has been started to be investigated to go beyond Moore's law. Among them, natural computing seeks to exploit natural physical or biological systems as computational resource. Quantum reservoir computing is an intersection of two different paradigms of natural computing, namely, quantum computing and reservoir computing.

K. Fujii (✉)
Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan
e-mail: fujii@qc.ee.es.osaka-u.ac.jp

K. Nakajima
Graduate School of Information Science and Technology, The University of Tokyo, Bunkyo-ku, 113-8656 Tokyo, Japan
e-mail: k_nakajima@mech.t.u-tokyo.ac.jp

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_18

Regarding quantum computing, the recent rapid experimental progress in controlling complex quantum systems motivates us to use quantum mechanical law as a new principle of information processing, namely, quantum information processing (Nielsen and Chuang 2010; Fujii 2015). For example, certain mathematical problems, such as integer factorisation, which are believed to be intractable on a classical computer, are known to be efficiently solvable by a sophisticatedly synthesized quantum algorithm (Shor 1994). Therefore, considerable experimental effort has been devoted to realizing full-fledged universal quantum computers (Barends et al. 2014; Kelly et al. 2015). In the near feature, quantum computers of size > 50 qubits with fidelity $> 99\%$ for each elementary gate would appear to achieve quantum computational supremacy beating simulation on the-state-of-the-art classical supercomputers (Preskill 2018; Boixo et al. 2018). While this does not directly mean that a quantum computer outperforms classical computers for a useful task like machine learning, now applications of such a near-term quantum device for useful tasks including machine learning has been widely explored. On the other hand, quantum simulators (Feynman 1982) are thought to be much easier to implement than a full-fledged universal quantum computer. In this regard, existing quantum simulators have already shed new light on the physics of complex many-body quantum systems (Cirac and Zoller 2012; Bloch et al. 2012; Georgescu et al. 2014), and a restricted class of quantum dynamics, known as adiabatic dynamics, has also been applied to combinatorial optimisation problems (Kadowaki and Nishimori 1998; Farhi et al. 2001; Rønnow et al. 2014; Boixo et al. 2014). However, complex real-time quantum dynamics, which is one of the most difficult tasks for classical computers to simulate (Morimae et al. 2014; Fujii et al. 2016; Fujii and Tamate 2016) and has great potential to perform nontrivial information processing, is now waiting to be harnessed as a resource for more general purpose information processing.

Physical reservoir computing, which is the main subject throughout this book, is another paradigm for exploiting complex physical systems for information processing. In this framework, the low-dimensional input is projected to a high-dimensional dynamical system, which is typically referred to as a *reservoir*, generating transient dynamics that facilitates the separation of input states (Rabinovich et al. 2008). If the dynamics of the reservoir involve both adequate memory and nonlinearity (Dambre et al. 2012), emulating nonlinear dynamical systems only requires adding a linear and static readout from the high-dimensional state space of the reservoir. A number of different implementations of reservoirs have been proposed, such as abstract dynamical systems for echo state networks (ESNs) (Jaeger and Haas 2004) or models of neurons for liquid state machines (Maass et al. 2002). The implementations are not limited to programs running on the PC but also include physical systems, such as the surface of water in a laminar state (Fernando and Sojakka 2003), analogue circuits and optoelectronic systems (Appeltant et al. 2011; Woods and Naughton 2012; Larger et al. 2012; Paquot et al. 2012; Brunner et al. 2013; Vandoorne et al. 2014), and neuromorphic chips (Stieg et al. 2012). Recently, it has been reported that the mechanical bodies of soft and compliant robots have also been successfully used as a reservoir (Hauser et al. 2011; Nakajima et al. 2013a, b, 2014, 2015; Caluwaerts et al. 2014). In contrast to the refinements required by learning algorithms, such as

in deep learning (LeCun et al. 2015), the approach followed by reservoir computing, especially when applied to real systems, is to find an appropriate form of physics that exhibits rich dynamics, thereby allowing us to outsource a part of the computation.

Quantum reservoir computing (QRC) was born in the marriage of quantum computing and physical reservoir computing above to harness complex quantum dynamics as a reservoir for real-time machine learning tasks (Fujii and Nakajima 2017). Since the idea of QRC has been proposed in Fujii and Nakajima (2017), its proof-of-principle experimental demonstration for non-temporal tasks (Negoro et al. 2018) and performance analysis and improvement (Nakajima et al. 2019; Kutvonen et al. 2020; Tran and Nakajima 2020) has been explored. The QRC approach to quantum tasks such as quantum tomography and quantum state preparation has been recently garnering attention (Ghosh et al. 2019a, b, 2020). In this book chapter, we will provide a broad picture of QRC and related approaches starting from a pedagogical introduction to quantum mechanics and machine learning.

The rest of this paper is organized as follows. In Sect. 2, we will provide a pedagogical introduction to quantum mechanics for those who are not familiar to it and fix our notation. In Sect. 3, we will briefly mention to several machine learning techniques like, linear and nonlinear regressions, temporal machine learning tasks and reservoir computing. In Sect. 4, we will explain QRC and related approaches, quantum extreme learning machine (Negoro et al. 2018) and quantum circuit learning (Mitarai et al. 2018). The former is a framework to use quantum reservoir for non-temporal tasks, that is, the input is fed into a quantum system, and generalization or classification tasks are performed by a linear regression on a quantum enhanced feature space. In the latter, the parameters of the quantum system are further fine-tuned via the gradient descent by measuring an analytically obtained gradient, just like the backpropagation for feedforward neural networks. Regarding QRC, we will also see chaotic time series predictions as demonstrations. Section 5 is devoted to conclusion and discussion.

2 Pedagogical Introduction to Quantum Mechanics

In this section, we would like to provide a pedagogical introduction to how quantum mechanical systems work for those who are not familiar to quantum mechanics. If you already familiar to quantum mechanics and its notations, please skip to Sect. 3.

2.1 Quantum State

A state of a quantum system is described by a *state vector*,

$$|\psi\rangle = \begin{pmatrix} c_1 \\ \vdots \\ c_d \end{pmatrix} \tag{1}$$

on a complex d-dimensional system $\mathbb{C}^d$, where the symbol $|\cdot\rangle$ is called *ket* and indicates a complex column vector. Similarly, $\langle\cdot|$ is called *bra* and indicates a complex row vector, and they are related complex conjugate,

$$\langle\psi| = |\psi\rangle^\dagger = \left(c_1^* \ldots c_d^*\right). \tag{2}$$

With this notation, we can write an inner product of two quantum state $|\psi\rangle$ and $|\phi\rangle$ by $\langle\psi|\phi\rangle$. Let us define an orthogonal basis

$$|1\rangle = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix}, \ldots \quad |k\rangle = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{pmatrix}, \ldots \quad |d\rangle = \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ d \end{pmatrix}, \tag{3}$$

a quantum state in the d-dimensional system can be described simply by

$$|\psi\rangle = \sum_{i=1}^{d} c_i |i\rangle. \tag{4}$$

The state is said to be a *superposition state* of $|i\rangle$. The coefficients $\{c_i\}$ are complex, and called *complex probability amplitudes*. If we measure the system in the basis $\{|i\rangle\}$, we obtain the measurement outcome i with a probability

$$p_i = |\langle i|\psi\rangle|^2 = |c_i|^2, \tag{5}$$

and hence the complex probability amplitudes have to be normalized as follows

$$|\langle\psi|\psi\rangle|^2 = \sum_{i=1}^{d} |c_i|^2 = 1. \tag{6}$$

In other words, a quantum state is represented as a normalized vector on a complex vector space.

Suppose the measurement outcome i corresponds to a certain physical value a_i, like energy, magnetization and so on, then the expectation value of the physical valuable is given by

$$\sum_i a_i p_i = \langle\psi|A|\psi\rangle \equiv \langle A\rangle, \tag{7}$$

where we define an hermitian operator

$$A = \sum_i a_i |i\rangle\langle i|, \tag{8}$$

which is called *observable*, and has the information of the measurement basis and physical valuable.

The state vector in quantum mechanics is similar to a probability distribution, but essentially different form it, since it is much more primitive; it can take complex value and is more like a square root of a probability. The unique features of the quantum systems come from this property.

2.2 Time Evolution

The time evolution of a quantum system is determined by a Hamiltonian H, which is a hermitian operator acting on the system. Let us denote a quantum state at time $t = 0$ by $|\psi(0)\rangle$. The equation of motion for quantum mechanics, so-called Schrödinger equation, is given by

$$i\frac{\partial}{\partial t}|\psi(t)\rangle = H|\psi(t)\rangle. \tag{9}$$

This equation can be formally solved by

$$|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle. \tag{10}$$

Therefore, the time evolution is given by an operator e^{-iHt}, which is a unitary operator and hence the norm of the state vector is preserved, meaning the probability conservation. In general, the Hamiltonian can be time dependent. Regarding the time evolution, if you are not interested in the continuous time evolution, but in just its input and output relation, then the time evolution is nothing but a unitary operator U

$$|\psi_{\text{out}}\rangle = U|\psi_{\text{in}}\rangle. \tag{11}$$

In quantum computing, the time evolution U is sometimes called *quantum gate*.

2.3 Qubits

The smallest nontrivial quantum system is a two-dimensional quantum system $\mathbb{C}^2$, which is called *quantum bit* or *qubit*:

$$\alpha|0\rangle + \beta|1\rangle, \quad (|\alpha|^2 + |\beta|^2 = 1). \tag{12}$$

Suppose we have n qubits. The n-qubit system is defined by a tensor product space $(\mathbb{C}^2)^{\otimes n}$ of each two-dimensional system as follows. A basis of the system is defined by a direct product of a binary state $|x_k\rangle$ with $x_k \in \{0, 1\}$,

$$|x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle, \tag{13}$$

which is simply denoted by

$$|x_1 x_2 \cdots x_n\rangle. \tag{14}$$

Then, a state of the n-qubit system can be described as

$$|\psi\rangle = \sum_{x_1, x_2, \ldots, x_n} \alpha_{x_1, x_2, \ldots, x_n} |x_1 x_2 \cdots x_n\rangle. \tag{15}$$

The dimension of the n-qubit system is 2^n, and hence the tensor product space is nothing but a 2^n-dimensional complex vector space $\mathbb{C}^{2^n}$. The dimension of the n-qubit system increases exponentially in the number n of the qubits.

2.4 Density Operator

Next, I would like to introduce operator formalism of the above quantum mechanics. This describes an exactly the same thing but sometimes the operator formalism would be convenient. Let us consider an operator ρ constructed from the state vector $|\psi\rangle$:

$$\rho = |\psi\rangle\langle\psi|. \tag{16}$$

If you chose the basis of the system $\{|i\rangle\}$ for the matrix representation, then the diagonal elements of ρ corresponds the probability distribution $p_i = |c_i|^2$ when the system is measured in the basis $\{|i\rangle\}$. Therefore, the operator ρ is called a density operator. The probability distribution can also be given in terms of ρ by

$$p_i = \mathrm{Tr}[|i\rangle\langle i|\rho], \tag{17}$$

where Tr is the matrix trace. An expectation value of an observable A is given by

$$\langle A\rangle = \mathrm{Tr}[A\rho]. \tag{18}$$

The density operator can handle a more general situation where a quantum state is sampled form a set of quantum states $\{|\psi_k\rangle\}$ with a probability distribution $\{q_k\}$. In this case, if we measure the system in the basis $\{|i\rangle\langle i|\}$, the probability to obtain the measurement outcome i is given by

$$p_i = \sum_k q_k \mathrm{Tr}[|i\rangle\langle i|\rho_k], \tag{19}$$

where $\rho_k = |\psi_k\rangle\langle\psi_k|$. By using linearity of the trace function, this reads

$$p_i = \mathrm{Tr}[|i\rangle\langle i| \sum_k q_k \rho_k]. \tag{20}$$

Now, we interpret that the density operator is given by

$$\rho = \sum_k q_k |\psi_k\rangle\langle\psi_k|. \tag{21}$$

In this way, a density operator can represent classical mixture of quantum states by a convex mixture of density operators, which is convenient in many cases. In general, a positive and hermitian operator ρ being subject to $\mathrm{Tr}[\rho] = 1$ can be a density operator, since it can be interpreted as a convex mixture of quantum states via spectral decomposition:

$$\rho = \sum \lambda_i |\lambda_i\rangle\langle\lambda_i|, \tag{22}$$

where $\{|\lambda_i\rangle\}$ and $\{\lambda_i\}$ are the eigenstates and eigenvectors, respectively. Because of $\mathrm{Tr}[\rho] = 1$, we have $\sum_i \lambda_i = 1$.

From its definition, the time evolution of ρ can be given by

$$\rho(t) = e^{-iHt}\rho(0)e^{iHt} \tag{23}$$

or

$$\rho_{\mathrm{out}} = U\rho_{\mathrm{in}}U^\dagger. \tag{24}$$

Moreover, we can define more general operations for the density operators. For example, if we apply unitary operators U and V with probabilities p and $(1-p)$, respectively, then we have

$$\rho_{\mathrm{out}} = pU\rho U^\dagger + (1-p)V\rho V^\dagger. \tag{25}$$

As another example, if we perform the measurement of ρ in the basis $\{|i\rangle\}$, and we forget about the measurement outcome, then the state is now given by a density operator

$$\sum_i \mathrm{Tr}[|i\rangle\langle i|\rho]|i\rangle\langle i| = \sum_i |i\rangle\langle i|\rho|i\rangle\langle i|. \tag{26}$$

Therefore, if we define a map from a density operator to another, which we call *superoperator*,

$$\mathcal{M}(\cdots) = \sum_i |i\rangle\langle i|(\cdots)|i\rangle\langle i|, \tag{27}$$

the above non-selective measurement (forgetting about the measurement outcomes) is simply written by

$$\mathcal{M}(\rho). \tag{28}$$

In general, any physically allowed quantum operation $\mathcal{K}$ that maps a density operator to another can be represented in terms of a set of operators $\{K_i\}$ being subject to $K_i^\dagger K_i = I$ with an identity operator I:

$$\mathcal{K}(\rho) = \sum_i K_i \rho K_i^\dagger. \tag{29}$$

The operators $\{K_i\}$ are called *Kraus operators*.

2.5 Vector Representation of Density Operators

Finally, we would like to introduce a vector representation of the above operator formalism. The operators themselves satisfy axioms of the linear space. Moreover, we can also define an inner product for two operators, so-called *Hilbert–Schmidt inner product*, by

$$\mathrm{Tr}[A^\dagger B]. \tag{30}$$

The operators on the n-qubit system can be spanned by the tensor product of Pauli operators $\{I, X, Y, Z\}^{\otimes n}$,

$$P(\boldsymbol{i}) = \bigotimes_{k=1}^{n} \sigma_{i_{2k-1} i_{2k}}. \tag{31}$$

where σ_{ij} is the Pauli operators:

$$I = \sigma_{00} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, X = \sigma_{10} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Z = \sigma_{01} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, Y = \sigma_{11} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}. \tag{32}$$

Since the Pauli operators constitute a complete basis on the operator space, any operator A can be decomposed into a linear combination of $P(\boldsymbol{i})$,

$$A = \sum_{\boldsymbol{i}} a_{\boldsymbol{i}} P(\boldsymbol{i}). \tag{33}$$

The coefficient $a_{\boldsymbol{i}}$ can be calculated by using the Hilbert–Schmidt inner product as follows:

$$a_{\boldsymbol{i}} = \mathrm{Tr}[P(\boldsymbol{i})A]/2^n, \tag{34}$$

by virtue of the orthogonality

$$\mathrm{Tr}[P(\boldsymbol{i})P(\boldsymbol{j})]/2^n = \delta_{\boldsymbol{i}\boldsymbol{j}}. \tag{35}$$

The number of the n-qubit Pauli operators $\{P(\boldsymbol{i})\}$ is 4^n, and hence a density operator ρ of the n-qubit system can be represented as a 4^n-dimensional vector

$$\boldsymbol{r} = \begin{pmatrix} r_{00\ldots0} \\ \vdots \\ r_{11\ldots1} \end{pmatrix}, \tag{36}$$

where $r_{00\ldots0} = 1/2^n$ because of $\mathrm{Tr}[\rho] = 1$. Moreover, because $P(\boldsymbol{i})$ is hermitian, $\boldsymbol{r}$ is a real vector. The superoperator $\mathcal{K}$ is a linear map for the operator, and hence can be represented as a matrix acting on the vector $\boldsymbol{r}$:

$$\rho' = \mathcal{K}(\rho) \Leftrightarrow \boldsymbol{r}' = K\boldsymbol{r}, \tag{37}$$

where the matrix element is given by

$$K_{\boldsymbol{i}\boldsymbol{j}} = \mathrm{Tr}[P(\boldsymbol{i})\mathcal{K}(P(\boldsymbol{j}))]/2^n. \tag{38}$$

In this way, a density operator ρ and a quantum operation $\mathcal{K}$ on it can be represented by a vector $\boldsymbol{r}$ and a matrix K, respectively.

3 Machine Learning and Reservoir Approach

In this section, we briefly introduce machine learning and reservoir approaches.

3.1 Linear and Nonlinear Regression

A supervised machine learning is a task to construct a model $f(x)$ from a given set of teacher data $\{x^{(j)}, y^{(j)}\}$ and to predict the output of an unknown input x. Suppose

x is a d-dimensional data, and $f(x)$ is one dimensional, for simplicity. The simplest model is *linear regression*, which models $f(x)$ as a linear function with respect to the input:

$$f(x) = \sum_{i=1}^{d} w_i x_i + w_0. \tag{39}$$

The weights $\{w_i\}$ and bias w_0 are chosen such that an error between $f(x)$ and the output of the teacher data, i.e. *loss*, becomes minimum. If we employ a quadratic loss function for given teacher data $\{\{x_i^{(j)}\}, y^{(j)}\}$, the problem we have to solve is as follows:

$$\min_{\{w_i\}} \sum_j (\sum_{i=0}^{d} w_i x_i^{(j)} - y^{(j)})^2, \tag{40}$$

where we introduced a constant node $x_0 = 1$. This corresponds to solving a superimposing equations:

$$\mathbf{y} = \mathbf{X}\mathbf{w}, \tag{41}$$

where $\mathbf{y}_j = y^{(j)}$, $\mathbf{X}_{ji} = x_i^{(j)}$, and $\mathbf{w}_i = w_i$. This can be solved by using the Moore–Penrose pseudo inverse $\mathbf{X}^+$, which can be defined from the singular value decomposition of $\mathbf{X} = UDV^T$ to be

$$\mathbf{X}^+ = VDU^T. \tag{42}$$

Unfortunately, the linear regression results in a poor performance in complicated machine learning tasks, and any kind of nonlinearity is essentially required in the model. A neural network is a way to introduce nonlinearity to the model, which is inspired by the human brain. In the neural network, the d-dimensional input data x is fed into N-dimensional hidden nodes with an $N \times d$ input matrix W^{in}:

$$W^{\text{in}}x. \tag{43}$$

Then, each element of the hidden nodes is now processed by a nonlinear activation function σ such as tanh, which is denoted by

$$\sigma(W^{\text{in}}x). \tag{44}$$

Finally, the output is extracted by an output weight W^{out} ($1 \times N$ dimensional matrix):

$$W^{\text{out}}\sigma(W^{\text{in}}x). \tag{45}$$

The parameters in W^{in} and W^{out} are trained such that the error between the output and teacher data becomes minimum. While this optimization problem is highly nonlinear, a gradient based optimization, so-called backpropagation, can be employed. To improve a representation power of the model, we can concatenate the linear transformation and the activation function as follows:

$$W^{\text{out}}\sigma\left(W^{(l)}\ldots\sigma\left(W^{(1)}\sigma(W^{\text{in}}x)\right)\right), \tag{46}$$

which is called multi-layer perceptron or deep neural network.

3.2 Temporal Task

The above task is not a temporal task, meaning that the input data is not sequential but given simultaneously like the recognition task of images for hand written language, pictures and so on. However, for a recognition of spoken language or prediction of time series like stock market, which are called temporal tasks, the network has to handle the input data that is given in a sequential way. To do so, the recurrent neural network feeds the previous states of the nodes back into the states of the nodes at next step, which allows the network to memorize the past input. In contrast, the neural network without any recurrency is called a feedforward neural network.

Let us formalize a temporal machine learning task with the recurrent neural network. For given input time series $\{x_k\}_{k=1}^{L}$ and target time series $\{\bar{y}_k\}_{k=1}^{L}$, a temporal machine learning is a task to generalize a nonlinear function,

$$\bar{y}_k = f(\{x_j\}_{j=1}^{k}). \tag{47}$$

For simplicity, we consider one-dimensional input and output time series, but their generalization to a multi-dimensional case is straightforward. To learn the nonlinear function $f(\{x_j\}_{j=1}^{k})$, the recurrent neural network can be employed as a model. Suppose the recurrent neural network consists of m nodes and is denoted by m-dimensional vector

$$\boldsymbol{r} = \begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix}. \tag{48}$$

To process the input time series, the nodes evolve by

$$\boldsymbol{r}(k+1) = \sigma[W\boldsymbol{r}(k) + W^{\text{in}}x_k], \tag{49}$$

where W is an $m \times m$ transition matrix and W^{in} is an $m \times 1$ input weight matrix. Nonlinearity comes from the nonlinear function σ applied on each element of the

nodes. The output time series from the network is defined in terms of a $1 \times m$ readout weights by

$$y_k = W^{\text{out}} \boldsymbol{r}(k). \tag{50}$$

Then, the learning task is to determine the parameters in W^{in}, W, and W^{out} by using the teacher data $\{x_k, \bar{y}_k\}_{k=1}^{L}$ so as to minimize an error between the teacher $\{\bar{y}_k\}$ and the output $\{y_k\}$ of the network.

3.3 Reservoir Approach

While the representation power of the recurrent neural network can be improved by increasing the number of the nodes, it makes the optimization process of the weights hard and unstable. Specifically, the backpropagation-based methods always suffer from the vanishing gradient problem. The idea of reservoir computing is to resolve this problem by mapping an input into a complex higher dimensional feature space, i.e., *reservoir*, and by performing simple linear regression on it.

Let us first see a reservoir approach on a feedforward neural network, which is called *extreme learning machine* (Huang et al. 2006). The input data x is fed into a network like multi-layer perceptron, where all weights are chosen randomly. The states of the hidden nodes at some layer are now regarded as basis functions of the input x in the feature space:

$$\{\phi_1(x), \phi_2(x), \ldots, \phi_N(x)\}. \tag{51}$$

Now, the output is defined as a linear combination of these

$$\sum_i w_i \phi_i(x) + w_0 \tag{52}$$

and hence the coefficients are determined simply by the linear regression as mentioned before. If the dimension and nonlinearity of the the basis functions are high enough, we can model a complex task simply by the linear regression.

The *echo state network* is similar but employs the reservoir idea for the recurrent neural network (Jaeger and Haas 2004; Maass et al. 2002; Verstraeten et al. 2007), which has been proposed before extreme learning machine appeared. To be specific, the input weights W^{in} and weight matrix W are both chosen randomly up to an appropriate normalization. Then, the learning task is done by finding the readout weights W^{out} to minimize the mean square error

$$\sum_k (y_k - \bar{y}_k)^2. \tag{53}$$

This problem can be solved stably by using the pseudo inverse as we mentioned before.

For both feedforward and recurrent types, the reservoir approach does not need to tune the internal parameters of the network depending on the tasks as long as it posses sufficient complexity. Therefore, the system, to which the machine learning tasks are outsourced, is not necessarily the neural network anymore, but any nonlinear physical system of large degree of freedoms can be employed as a *reservoir* for information processing, namely, physical reservoir computing (Fernando and Sojakka 2003; Appeltant et al. 2011; Woods and Naughton 2012; Larger et al. 2012; Paquot et al. 2012; Brunner et al. 2013; Vandoorne et al. 2014; Stieg et al. 2012; Hauser et al. 2011; Nakajima et al. 2013a, b, 2014, 2015; Caluwaerts et al. 2014).

4 Quantum Machine Learning on Near-Term Quantum Devices

In this section, we will see QRC and related frameworks for quantum machine learning. Before going deep into the temporal tasks done on QRC, we first explain how complicated quantum natural dynamics can be exploit as generalization and classification tasks. This can be viewed as a quantum version of extreme learning machine (Negoro et al. 2018). While it is an opposite direction to reservoir computing, we will also see *quantum circuit learning* (QCL) (Mitarai et al. 2018), where the parameters in the complex dynamics is further tuned in addition to the linear readout weights. QCL is a quantum version of a feedforward neural network. Finally, we will explain quantum reservoir computing by extending quantum extreme learning machine for temporal learning tasks.

4.1 Quantum Extreme Learning Machine

The idea of quantum extreme learning machine lies in using a Hilbert space, where quantum states live, as an enhanced feature space of the input data. Let us denote the set of input and teacher data by $\{x^{(j)}, \bar{y}^{(j)}\}$. Suppose we have an n-qubit system, which is initialized to

$$|0\rangle^{\otimes n}. \tag{54}$$

In order to feed the input data into quantum system, a unitary operation parameterized by x, say $V(x)$, is applied on the initial state:

$$V(x)|0\rangle^{\otimes n}. \tag{55}$$

For example, if x is one-dimensional data and normalized to be $0 \leq x \leq 1$, then we may employ the Y-basis rotation $e^{-i\theta Y}$ with an angle $\theta = \arccos(\sqrt{x})$:

$$e^{-i\theta Y}|0\rangle = \sqrt{x}|0\rangle + \sqrt{1-x}|1\rangle. \tag{56}$$

The expectation value of Z with respect to $e^{-i\theta Y}|0\rangle$ becomes

$$\langle Z\rangle = 2x - 1, \tag{57}$$

and hence is linearly related to the input x. To enhance the power of quantum enhanced feature space, the input could be transformed by using a nonlinear function ϕ:

$$\theta = \arccos(\sqrt{\phi(x)}). \tag{58}$$

The nonlinear function ϕ could be, for example, hyperbolic tangent, Legendre polynomial, and so on. For simplicity, below we will use the simple linear input $\theta = \arccos(\sqrt{x})$.

If we apply the same operation on each of the n qubits, we have

$$\begin{aligned} V(x)|0\rangle^{\otimes n} &= (\sqrt{x}|0\rangle + \sqrt{1-x}|1\rangle)^{\otimes n} \\ &= (1-x)^{n/2} \sum_{i_1,\ldots,i_n} \prod_k \sqrt{\frac{x}{1-x}}^{i_k} |i_1,\ldots,i_n\rangle. \end{aligned} \tag{59}$$

Therefore, we have coefficients that are nonlinear with respect to the input x because of the tensor product structure. Still the expectation value of the single-qubit operator Z_k on the kth qubit is $2x - 1$. However, if we measure a *correlated* operator like $Z_1 Z_2$, we can obtain a second-order nonlinear output

$$\langle Z_1 Z_2\rangle = (2x-1)^2 \tag{60}$$

with respect to the input x. To measure a correlated operator, it is enough to apply an entangling unitary operation like CNOT gate $\Lambda(X) = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X$:

$$\langle\psi|\Lambda_{1,2}(X) Z_1 \Lambda_{1,2}(X)|\psi\rangle = \langle\psi|Z_1 Z_2|\psi\rangle. \tag{61}$$

In general, an n-qubit unitary operation U transforms the observable Z under the conjugation into a linear combination of Pauli operators:

$$U^\dagger Z_1 U = \sum_{\boldsymbol{i}} \alpha_{\boldsymbol{i}} P(\boldsymbol{i}). \tag{62}$$

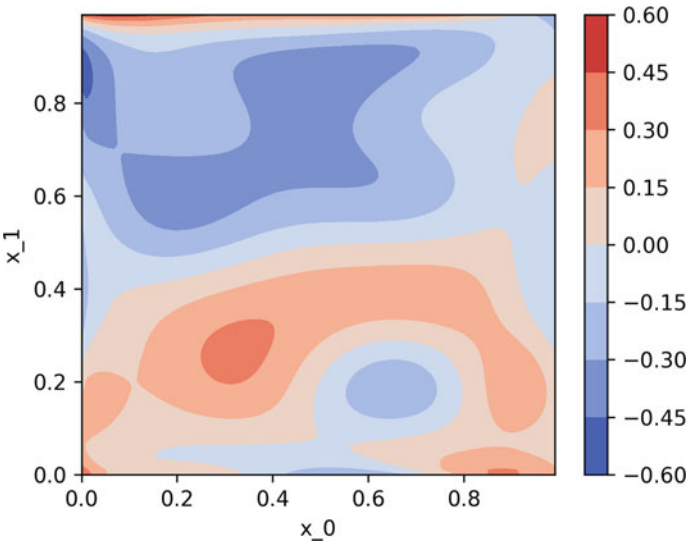


Fig. 1 The expectation value $\langle Z \rangle$ of the output of a quantum circuit as a function of the input (x_0, x_1)

Thus if you measure the output of the quantum circuit after applying a unitary operation U,

$$UV(x)|0\rangle^{\otimes n}, \tag{63}$$

you can get a complex nonlinear output, which could be represented as a linear combination of exponentially many nonlinear functions. U should be chosen to be appropriately complex with keeping experimental feasibility but not necessarily fine-tuned.

To see how the output behaves in a nonlinear way with respect to the input, in Fig. 1, we will plot the output $\langle Z \rangle$ for the input (x_0, x_1) and $n = 8$, where the inputs are fed into the quantum state by the Y-rotation with angles

$$\theta_{2k} = k \arccos(\sqrt{x_0}) \tag{64}$$

$$\theta_{2k+1} = k \arccos(\sqrt{x_1}) \tag{65}$$

on the $2k$th and $(2k + 1)$th qubits, respectively. Regarding the unitary operation U, random two-qubit gates are sequentially applied on any pairs of two qubits on the 8-qubit system.

Suppose the Pauli Z operator is measured on each qubit as an observable. Then, we have

$$z_i = \langle Z_i \rangle, \tag{66}$$

for each qubit. In quantum extreme learning machine, the output is defined by taking linear combination of these n output:

$$y = \sum_{i=1}^{n} w_i z_i . \tag{67}$$

Now, the linear readout weights $\{w_i\}$ are tuned so that the quadratic loss function

$$L = \sum_j (y^{(j)} - \bar{y}^{(j)})^2 \tag{68}$$

becomes minimum. As we mentioned previously, this can be solved by using the pseudo inverse. In short, quantum extreme learning machine is a linear regression on a randomly chosen nonlinear basis functions, which come from the quantum state in a space of an exponentially large dimension, namely quantum enhanced feature space. Furthermore, under some typical nonlinear function and unitary operations settings to transform the observables, the output in Eq. (67) can approximate any continuous function of the input. This property is known as the universal approximation property (UAP), which implies that the quantum extreme learning machine can handle a wide class of machine learning tasks with at least the same power as the classical extreme learning machine (Goto et al. 2020).

Here we should note that a similar approach, quantum kernel estimation, has been taken in Havlicek et al. (2019) and Kusumoto et al. (2021). In quantum extreme learning machine, a classical feature vector $\phi_i(x) \equiv \langle \Phi(x)|Z_i|\Phi(x)\rangle$ is extracted from observables on the quantum feature space $|\Phi(x)\rangle \equiv V(x)|0\rangle^{\otimes n}$. Then, linear regression is taken by using the classical feature vector. On the other hand, in quantum kernel estimation, quantum feature space is fully employed by using support vector machine with the kernel functions $K(x, x') \equiv \langle \Phi(x)|\Phi(x')\rangle$, which can be estimated on a quantum computer. While classification power would be better for quantum kernel estimation, it requires more quantum computational costs both for learning and prediction in contrast to quantum extreme learning machine.

In Fig. 2,we demonstrate quantum extreme learning machine for a two-class classification task of a two-dimensional input $0 \leq x_0, x_1 \leq 1$. Class 0 and 1 are defined to be those being subject to $(x_0 - 0.5)^2 + (x_1 - 0.5)^2 \leq 0.15$ and > 0.15, respectively. The linear readout weights $\{w_i\}$ are learned with 1000 randomly chosen training data and prediction is performed with 1000 randomly chosen inputs. The class 0 and 1 are determined whether or not the output y is larger than 0.5. Quantum extreme learning machine with an 8-qubit quantum circuit shown in Fig. 2a succeeds to predict the class with 95% accuracy. On the other hand, a simple linear regression for (x_0, x_1) results in 39%. Moreover, quantum extreme learning machine with $U = I$, meaning no entangling gate, also results in poor, 42%. In this way, the feature space enhanced by quantum entangling operations is important to obtain a good performance in quantum extreme learning machine.

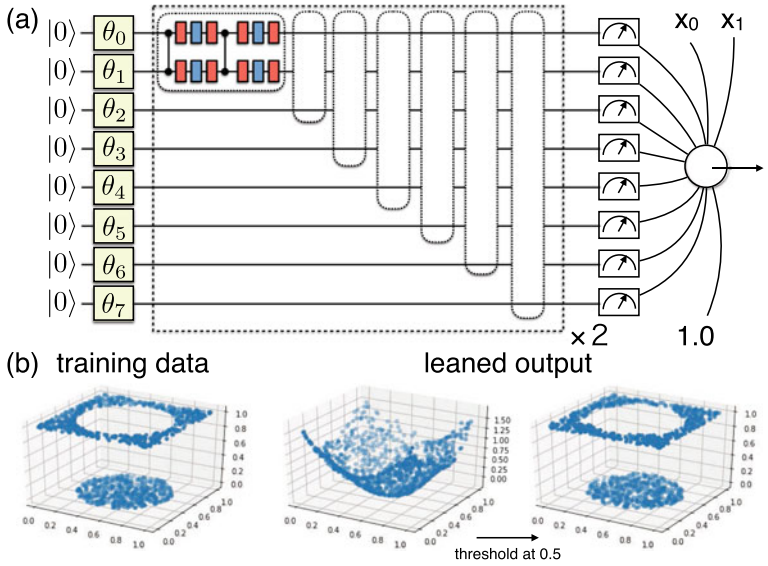


Fig. 2 **a** The quantum circuit for quantum extreme learning machine. The box with $theta_k$ indicates Y-rotations by angles θ_k. The red and blue boxes correspond to X and Z rotations by random angles, Each dotted-line box represents a two-qubit gate consisting of two controlled-Z gates and 8 X-rotations and 4 Z-rotations. As denoted by the dashed-line box, the sequence of the 7 dotted boxes is repeated twice. The readout is defined by a linear combination of $\langle Z_i \rangle$ with constant bias term 1.0 and the input (x_0, x_1). **b** (Left) The training data for a two-class classification problem. (Middle) The readout after learning. (Right) Prediction from the readout with threshold at 0.5

4.2 *Quantum Circuit Learning*

In the split of reservoir computing, dynamics of a physical system is not fine-tuned but natural dynamics of the system is harnessed for machine learning tasks. However, if we see the-state-of-the-art quantum computing devices, the parameter of quantum operations can be finely tuned as done for universal quantum computing. Therefore, it is natural to extend quantum extreme learning machine by tuning the parameters in the quantum circuit just like feedfoward neural networks with backpropagation.

Using parameterized quantum circuits for supervised machine leaning tasks such as generalization of nonlinear functions and pattern recognitions have been proposed in Mitarai et al. (2018), Farhi and Neven (2018), which we call *quantum circuit learning*. Let us consider the same situation with quantum extreme learning machine. The state before the measurement is given by

$$UV(x)|0\rangle^{\otimes n}. \tag{69}$$

In the case of quantum extreme learning machine, the unitary operation for a nonlinear transformation with respect to the input parameter x is randomly chosen. However, the unitary operation U may also be parameterized:

$$U(\{\phi_k\}) = \prod_k u(\phi_k). \tag{70}$$

Thereby, the output from the quantum circuit with respect to an observable A

$$\langle A(\{\phi_k\}, x)\rangle = \langle 0|^{\otimes n} V^\dagger(x) U(\{\phi_k\})^\dagger Z_i U(\{\phi_k\}) V(x)|0\rangle^{\otimes n} \tag{71}$$

becomes a function of the circuit parameters $\{\phi_k\}$ in addition to the input x. Then, the parameters $\{\phi_k\}$ are tuned so as to minimize the error between teacher data and the output, for example, by using the gradient just like the output of the feedforward neural network.

Let us define a teacher dataset $\{x^{(j)}, y^{(j)}\}$ and a quadratic loss function

$$L(\{\phi_k\}) = \sum_j (\langle A(\{\phi_k\}, x^{(j)})\rangle - y^{(j)})^2. \tag{72}$$

The gradient of the loss function can be obtained as follows:

$$\frac{\partial}{\partial \phi_l} L(\{\phi_k\}) = \frac{\partial}{\partial \phi_l} \sum_j (\langle A(\{\phi_k\}, x^{(j)})\rangle - y^{(j)})^2 \tag{73}$$

$$= \sum_j 2(\langle A(\{\phi_k\}, x^{(j)})\rangle - y^{(j)}) \frac{\partial}{\partial \phi_l} \langle A(\{\phi_k\}, x^{(j)})\rangle. \tag{74}$$

Therefore, if we can measure the gradient of the observable $\langle A(\{\phi_k\}, x^{(j)})\rangle$, the loss function can be minimized according to the gradient descent.

If the unitary operation $u(\phi_k)$ is given by

$$u(\phi_k) = W_k e^{-i(\phi_k/2)P_k}, \tag{75}$$

where W_k is an arbitrary unitary, and P_k is a Pauli operator. Then, the partial derivative with respect to the lth parameter can be analytically calculated from the outputs $\langle A(\{\phi_k\}, x^{(j)})\rangle$ with shifting the lth parameter by $\pm\epsilon$ (Mitarai et al. 2018; Mitarai and Fujii 2019):

$$\begin{aligned} &\frac{\partial}{\partial \phi_l} \langle A(\{\phi_k\}, x^{(j)})\rangle \\ &= \frac{1}{2\sin\epsilon} (\langle A(\{\phi_1, \ldots, \phi_l + \epsilon, \phi_{l+1}, \ldots\}, x^{(j)})\rangle - \langle A(\{\phi_1, \ldots, \phi_l - \epsilon, \phi_{l+1}, \ldots\}, x^{(j)})\rangle). \end{aligned} \tag{76}$$

By considering the statistical error to measure the observable $\langle A \rangle$, ϵ should be chosen to be $\epsilon = \pi/2$ so as to make the denominator maximum. After measuring the partial derivatives for all parameters ϕ_k and calculating the gradient of the loss function $L(\{\phi_k\})$, the parameters are now updated by the gradient descent:

$$\theta_l^{(m+1)} = \theta_l^{(m)} - \alpha \frac{\partial}{\partial \phi_l} L(\{\phi_k\}). \tag{77}$$

The idea of using the parameterized quantum circuits for machine learning is now widespread. After the proposal of quantum circuit learning based on the analytical gradient estimation above (Mitarai et al. 2018) and a similar idea (Farhi and Neven 2018), several researches have been performed with various types of parameterized quantum circuits (Schuld et al. 2020; Huggins et al. 2019; Chen et al. 2018; Glasser et al. 2018; Du et al. 2018) and various models and types of machine learning including generative models (Benedetti et al. 2019a; Liu and Wang 2018) and generative adversarial models (Benedetti et al. 2019b; Situ et al. 2020; Zeng et al. 2019; Romero and Aspuru-Guzik 2019). Moreover, an expression power of the parameterized quantum circuits and its advantage against classical probabilistic models have been investigated (Du et al. 2020). Experimentally feasible ways to measure an analytical gradient of the parameterized quantum circuits have been investigated (Mitarai and Fujii 2019; Schuld et al. 2019; Vidal and Theis 2018). An advantage of using such a gradient for the parameter optimization has been also argued in a simple setting (Harrow and John 2019), while the parameter tuning becomes difficult because of the vanishing gradient by an exponentially large Hilbert space (McClean et al. 2018). Software libraries for optimizing parameterized quantum circuits are now developing (Bergholm et al. 2018; Chen et al. 2019). Quantum machine learning on near-term devices, especially for quantum optical systems, is proposed in Steinbrecher et al. (2019), Killoran et al. (2019). Quantum circuit learning with parameterized quantum circuits has been already experimentally demonstrated on superconducting qubit systems (Havlicek et al. 2019; Wilson et al. 2018) and a trapped ion system (Zhu et al. 2019).

4.3 Quantum Reservoir Computing

Now, we return to the reservoir approach and extend quantum extreme learning machine from non-temporal tasks to temporal ones, namely, quantum reservoir computing (Fujii and Nakajima 2017). We consider a temporal task, which we explained in Sect. 3.2. The input is given by a time series $\{x_k\}_k^L$ and the purpose is to learn a nonlinear temporal function:

$$\bar{y}_k = f(\{x_j\}_j^k). \tag{78}$$

To this end, the target time series $\{\bar{y}_k\}_{k=1}^L$ is also provided as teacher.

Contrast to the previous setting with non-temporal tasks, we have to fed input into a quantum system sequentially. This requires us to perform an initialization process during computation, and hence the quantum state of the system becomes mixed state. Therefore, in the formulation of QRC, we will use the vector representation of density operators, which was explained in Sect. 2.5.

In the vector representation of density operators, the quantum state of an N-qubit system is given by a vector in a 4^N-dimensional real vector space, $\boldsymbol{r} \in \mathbb{R}^{4^N}$. In QRC, similarly to recurrent neural networks, each element of the 4^N-dimensional vector is regarded as a *hidden* node of the network. As we seen in Sect. 2.5, any physical operation can be written as a linear transformation of the real vector by a $4^N \times 4^N$ matrix W:

$$\boldsymbol{r}' = W\boldsymbol{r}. \tag{79}$$

Now we see, from Eq. (79), a time evolution similar to the recurrent neural network, $\boldsymbol{r}' = \tanh(W\boldsymbol{r})$. However, there is no nonlinearity such as tanh in each quantum operation W. Instead, the time evolution W can be changed according to the external input x_k, namely W_{x_k}, which contrasts to the conventional recurrent neural network where the input is fed additively $W\boldsymbol{r} + W^{\text{in}}x_k$. This allows the quantum reservoir to process the input information $\{x_k\}$ nonlinearly, by repetitively feeding the input.

Suppose the input $\{x_k\}$ is normalized such that $0 \leq x_k \leq 1$. As an input, we replace a part of the qubits to the quantum state. The density operator is given by

$$\rho_{x_k} = \frac{I + (2x_k - 1)Z}{2}. \tag{80}$$

For simplicity, below we consider the case where only one qubit is replaced for the input. Corresponding matrix S_{x_k} is given by

$$(S_{x_k})_{\boldsymbol{ji}} = \text{Tr}\left\{ P(\boldsymbol{j}) \frac{I + (2x_k - 1)Z}{2} \otimes \text{Tr}_{\text{replace}}[P(\boldsymbol{i})] \right\} / 2^N, \tag{81}$$

where $\text{Tr}_{\text{replace}}$ indicates a partial trace with respect to the replaced qubit. With this definition, we have

$$\rho' = \text{Tr}_{\text{replace}}[\rho] \otimes \rho_{x_k} \Leftrightarrow \boldsymbol{r}' = S_{x_k}\boldsymbol{r}. \tag{82}$$

The unitary time evolution, which is necessary to obtain a nonlinear behavior with respect to the input valuable x_k, is taken as a Hamiltonian dynamics $e^{-iH\tau}$ for a given time interval τ. Let us denote its representation on the vector space by U_τ:

$$\rho' = e^{-iH\tau}\rho e^{iH\tau} \Leftrightarrow \boldsymbol{r}' = U_\tau \boldsymbol{r}. \tag{83}$$

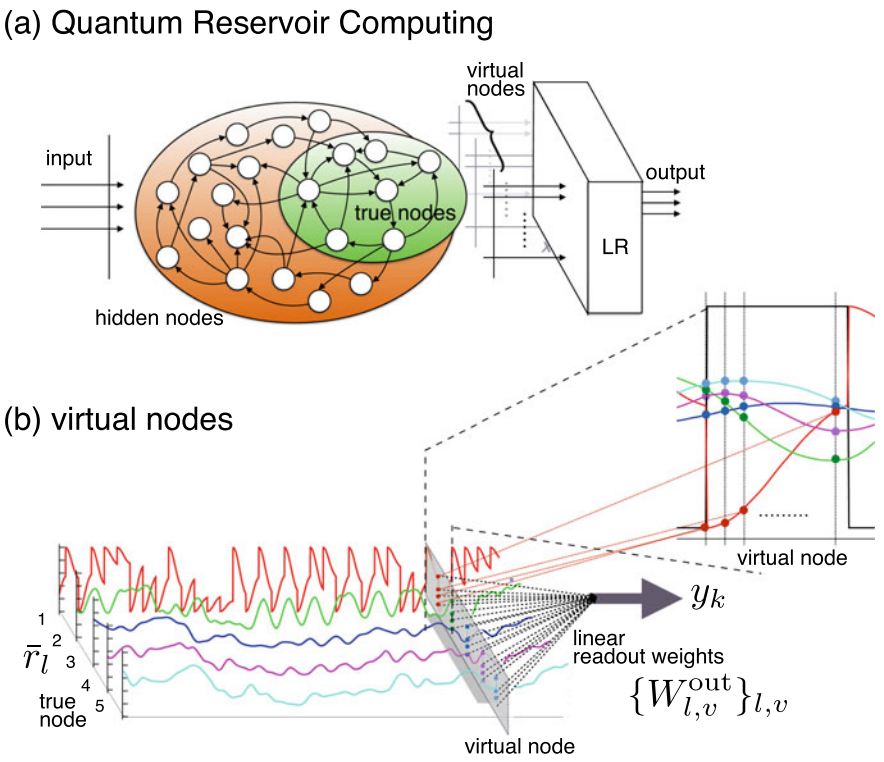


Fig. 3 **a** Quantum reservoir computing. **b** Virtual nodes and temporal multiplexing

Then, a unit time step is written as an input-depending linear transformation:

$$\boldsymbol{r}((k+1)\tau) = U_\tau S_{x_k} \boldsymbol{r}(k\tau). \tag{84}$$

where $\boldsymbol{r}(k\tau)$ indicates the hidden nodes at time $k\tau$.

Since the number of the hidden nodes are exponentially large, it is not feasible to observe all nodes from experiments. Instead, a set of observed nodes $\{\bar{r}_l\}_{l=1}^{M}$, which we call *true nodes*, is defined by a $M \times 4^N$ matrix R,

$$\bar{r}_l(k\tau) = \sum_i R_{li} r_i(k\tau). \tag{85}$$

The number of true nodes M has to be a polynomial in the number of qubits N. That is, from exponentially many hidden nodes, a polynomial number of true nodes are obtained to define the output from QR (see Fig. 3a):

$$y_k = \sum_l W^{\text{out}}_l \bar{r}_l(k\tau), \tag{86}$$

where W_{out} is the readout weights, which is obtained by using the training data. For simplicity, we take the single-qubit Pauli Z operator on each qubit as the true nodes, i.e.,

$$\bar{r}_l = \text{Tr}[Z_l \rho], \tag{87}$$

so that if there is no dynamics these nodes simply provide a linear output $(2x_k - 1)$ with respect to the input x_k.

Moreover, in order to improve the performance we also perform the temporal multiplexing. The temporal multiplexing has been found to be useful to extract complex dynamics on the exponentially large hidden nodes through the restricted number of the true nodes (Fujii and Nakajima 2017). In temporal multiplexing, not only the true nodes just after the time evolution U_τ, also at each of the subdivided V time intervals during the unitary evolution U_τ to construct V virtual nodes, as shown in Fig. 3b. After each input by S_{x_k}, the signals from the hidden nodes (via the true nodes) are measured for each subdivided intervals after the time evolution by $U_{v\tau/V}$ $(v = 1, 2, \ldots V)$, i.e.,

$$\boldsymbol{r}(k\tau + (v/V)\tau) \equiv U_{(v/V)\tau} S_{x_k} \boldsymbol{r}(k\tau). \tag{88}$$

In total, now we have $N \times V$ nodes, and the output is defined as their linear combination:

$$y_k = \sum_{l=1}^{N} \sum_{v=1}^{V} W_{j,v}^{\text{out}} \bar{r}_l(k\tau + (v/V)\tau). \tag{89}$$

By using the teacher data $\{\bar{y}_k\}_k^L$, the linear readout weights $W_{j,v}^{\text{out}}$ can be determined by using the pseudo inverse. In Fujii and Nakajima (2017), the performance of QRC has been investigated extensively for both binary and continuous inputs. The result shows that even if the number of the qubits are small like 5–7 qubits the performance as powerful as the echo state network of the 100–500 nodes have been reported both in short term memory and parity check capacities. Note that, although we do not go into detail in this chapter, the technique called spatial multiplexing (Nakajima et al. 2019), which exploits multiple quantum reservoirs with common input sequence injected, is also introduced to harness quantum dynamics as a computational resource. Recently, QRC has been further investigated in Kutvonen et al. (2020), Ghosh et al. (2019a), Chen and Nurdin (2019). Specifically, in Ghosh et al. (2019a), the authors use quantum reservoir computing to detect many-body entanglement by estimating nonlinear functions of density operators like entropy.

4.4 Emulating Chaotic Attractors Using Quantum Dynamics

To see a performance of QRC, here we demonstrate an emulation of chaotic attractors. Suppose $\{x_k\}_k^L$ is a discretized time sequence being subject to a complex nonlinear equation, which might has a chaotic behavior. In this task, the target, which the network is to output, is defined to be

$$\bar{y}_k = x_{k+1} = f(\{x_j\}_{j=1}^k). \tag{90}$$

That is, the system learns the input of the next step. Once the system successfully learns $\bar{y}_k$, by feeding the output into the input of the next step of the system, the system evolves autonomously.

Here, we employ the following target time series from chaotic attractors: (i) Lorenz attractor,

$$\frac{dx}{dt} = a(y - x), \tag{91}$$

$$\frac{dy}{dt} = x(b - z) - y, \tag{92}$$

$$\frac{dz}{dt} = xy - cz, \tag{93}$$

with $(a, b, c) = (10, 28, 8/3)$, (ii) the chaotic attractor of Mackey–Glass equation,

$$\frac{d}{dt}x(t) = \beta \frac{x(t-\tau)}{1 + x(t-\tau)^n} - \gamma x(t) \tag{94}$$

with $(\beta, \gamma, n) = (0.2, 0.1, 10)$ and $\tau = 17$, (iii) Rössler attoractor,

$$\frac{dx}{dt} = -y - z, \tag{95}$$

$$\frac{dy}{dt} = x + ay, \tag{96}$$

$$\frac{dz}{dt} = b + z(x - c), \tag{97}$$

with $(0.2, 0.2, 5.7)$, and (iv) Hénon map,

$$x_{t+1} = 1 - 1.4x_t + 0.3x_{t-1}. \tag{98}$$

Regarding (i)-(iii), the time series is obtained by using the fourth-order Runge–Kutta method with step size 0.02, and only $x(t)$ is employed as a target. For the time evolution of quantum reservoir, we employ a fully connected transverse-field Ising model

$$H = \sum_{ij} J_{ij} X_i X_j + h Z_i, \tag{99}$$

where the coupling strengths are randomly chosen such that J_{ij} is distributed randomly from $[-0.5, 0.5]$ and $h = 1.0$. The time interval and the number of the virtual nodes are chosen to be $\tau = 4.0$ and $v = 10$ so as to obtain the best performance. The first 10^4 steps are used for training. After the linear readout weights are determined, several 10^3 steps are predicted by autonomously evolving the quantum reservoir. The results are shown in Fig. 4 for each of (a) Lorenz attractor, (b) the chaotic attractor of Mackey–Glass system, (c) Rössler attractor, and (d) Hénon map. All these results show that training is done well and the prediction is successful for several

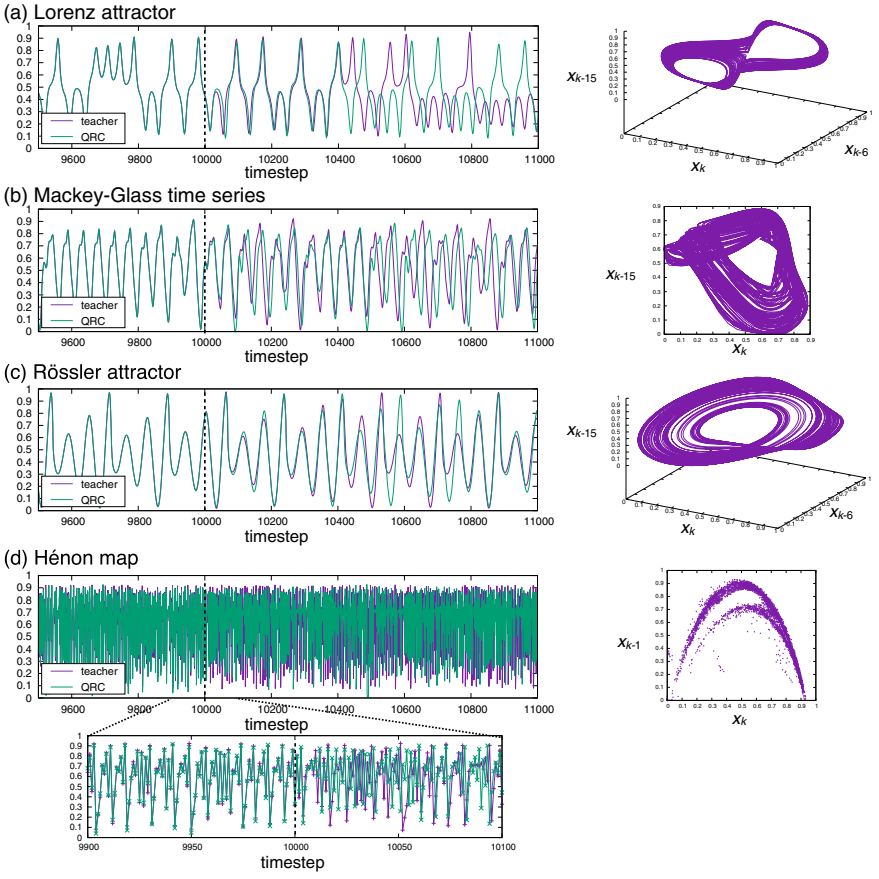


Fig. 4 Demonstrations of chaotic attractor emulations. **a** Lorenz attractor. **b** Mackey–Glass system. **c** Rössler attractor. **d** Hénon map. The dotted line shows the time step when the system is switched from teacher forced state to autonomous state. In the right side, delayed phase diagrams of learned dynamics are shown

hundreds steps. Moreover, the output from the quantum reservoir also successfully reconstruct the structures of these chaotic attractors as you can see from the delayed phase diagram.

5 Conclusion and Discussion

Here, we reviewed quantum reservoir computing and related approaches, quantum extreme learning machine and quantum circuit learning. The idea of quantum reservoir computing comes from the spirit of reservoir computing, i.e., outsourcing information processing to natural physical systems. This idea is best suited to quantum machine learning on near-term quantum devices in noisy intermediate quantum (NISQ) era. Since reservoir computing uses complex physical systems as a feature space to construct a model by the simple linear regression, this approach would be a good way to understand the power of a quantum enhanced feature space.

Acknowledgements KF is supported by KAKENHI No.16H02211, JST PRESTO JPMJPR1668, JST ERATO JPMJER1601, and JST CREST JPMJCR1673. KN is supported by JST PRESTO Grant Number JPMJPR15E7, Japan, by JSPS KAKENHI Grant Numbers JP18H05472, JP16KT0019, and JP15K16076. KN would like to acknowledge Dr. Quoc Hoan Tran for his helpful comments. This work is supported by MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) Grant No. JPMXS0118067394.

References

L. Appeltant, M.C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C.R. Mirasso, I. Fischer, Information processing using a single dynamical node as complex system. Nat. Commun. **2**, 468 (2011)

R. Barends et al., Superconducting quantum circuits at the surface code threshold for fault tolerance. Nature **508**, 500 (2014)

M. Benedetti et al., A generative modeling approach for benchmarking and training shallow quantum circuits. NPJ Quantum Inf. **5**, 45 (2019a)

M. Benedetti et al., Adversarial quantum circuit learning for pure state approximation. New J. Phys. **21** (2019b)

V. Bergholm et al., PennyLane: automatic differentiation of hybrid quantum-classical computations (2018), arXiv:1811.04968

I. Bloch, J. Dalibard, S. Nascimbène, Quantum simulations with ultracold quantum gases. Nat. Phys. **8**, 267 (2012)

S. Boixo, T.F. Rønnow, S.V. Isakov, Z. Wang, D. Wecker, D.A. Lidar, J.M. Martinis, M. Troyer, Evidence for quantum annealing with more than one hundred qubits. Nat. Phys. **10**, 218 (2014)

S. Boixo et al., Characterizing quantum supremacy in near-term devices. Nat. Phys. **14**, 595 (2018)

D. Brunner, M.C. Soriano, C.R. Mirasso, I. Fischer, Parallel photonic information processing at gigabyte per second data rates using transient states. Nat. Commun. **4**, 1364 (2013)

K. Caluwaerts, J. Despraz, A. Işçen, A.P. Sabelhaus, J. Bruce, B. Schrauwen, V. SunSpiral, Design and control of compliant tensegrity robots through simulations and hardware validation. J. R. Soc. Interface **11**, 20140520 (2014)

J. Chen, H.I. Nurdin, Learning nonlinear input-output maps with dissipative quantum systems. Quantum Inf. Process. **18**, 198 (2019)
H. Chen et al., Universal discriminative quantum neural networks. Quantum Mach. Intell. **3**, 1 (2021)
Z.-Y. Chen et al., VQNet: library for a quantum-classical hybrid neural network (2019), arXiv:1901.09133
J.I. Cirac, P. Zoller, Goals and opportunities in quantum simulation. Nat. Phys. **8**, 264 (2012)
J. Dambre, D. Verstraeten, B. Schrauwen, S. Massar, Information processing capacity of dynamical systems. Sci. Rep. **2**, 514 (2012)
Y. Du et al., Implementable quantum classifier for nonlinear data (2018), arXiv:1809.06056
Y. Du et al., The expressive power of parameterized quantum circuits. Phys. Rev. Res. **2** (2020)
E. Farhi, H. Neven, Classification with quantum neural networks on near term processors (2018), arXiv:1802.06002
E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, D. Preda, A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. Science **292**, 472 (2001)
C. Fernando, S. Sojakka, *Pattern Recognition in a Bucket*. Lecture Notes in Computer Science, vol. 2801 (Springer, 2003), p. 588
R.P. Feynman, Simulating physics with computers. Int. J. Theor. Phys. **21**, 467 (1982)
K. Fujii, *Quantum Computation with Topological Codes-From Qubit to Topological Fault-Tolerance* Springer Briefs in Mathematical Physics. (Springer, Berlin, 2015)
K. Fujii, S. Tamate, Computational quantum-classical boundary of complex and noisy quantum systems. Sci. Rep. **6**, 25598 (2016)
K. Fujii, K. Nakajima, Harnessing disordered-ensemble quantum dynamics for machine learning. Phys. Rev. Appl. **8** (2017)
K. Fujii, H. Kobayashi, T. Morimae, H. Nishimura, S. Tamate, S. Tani, Power of Quantum Computation with Few Clean Qubits, in *Proceedings of 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)* (2016), pp. 13:1–13:14
I.M. Georgescu, S. Ashhab, F. Nori, Quantum simulation. Rev. Mod. Phys. **86**, 153 (2014)
S. Ghosh et al., Quantum reservoir processing. NPJ Quantum Inf. **5**, 35 (2019a)
S. Ghosh, T. Paterek, T.C.H. Liew, Quantum neuromorphic platform for quantum state preparation. Phys. Rev. Lett. **123** (2019b)
S. Ghosh et al., *Reconstructing quantum states with quantum reservoir networks*. IEEE Trans. Neural Netw. Learn. Syst. 1–8 (2020)
I. Glasser, N. Pancotti, J.I. Cirac, From probabilistic graphical models to generalized tensor networks for supervised learning (2018), arXiv:1806.05964
T. Goto, Q.H. Tran, K. Nakajima, Universal approximation property of quantum feature maps (2020), arXiv: 2009.00298
A. Harrow, N. John, Low-depth gradient measurements can improve convergence in variational hybrid quantum-classical algorithms. Phys. Rev. Lett. **126**, 140502 (2021)
H. Hauser, A.J. Ijspeert, R.M. Füchslin, R. Pfeifer, W. Maass, Towards a theoretical foundation for morphological computation with compliant bodies. Biol. Cybern. **105**, 355 (2011)
V. Havlicek et al., Supervised learning with quantum enhanced feature spaces. Nature **567**, 209 (2019)
G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications. Neurocomputing **70**, 489 (2006)
W. Huggins et al., Towards quantum machine learning with tensor networks. Quantum Sci. Technol. **4** (2019)
H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**, 78 (2004)
T. Kadowaki, H. Nishimori, Quantum annealing in the transverse Ising model. Phys. Rev. E **58**, 5355 (1998)
J. Kelly et al., State preservation by repetitive error detection in a superconducting quantum circuit. Nature **519**, 66 (2015)

N. Killoran et al., Continuous-variable quantum neural networks. Phys. Rev. Res. **1** (2019)
Kusumoto et al., Experimental quantum kernel trick with nuclear spins in a solid. npj Quantum Inf. **7**, 94 (2021)
A. Kutvonen, K. Fujii, T. Sagawa, Optimizing a quantum reservoir computer for time series prediction. Sci. Rep. **10**, 14687 (2020)
L. Larger, M.C. Soriano, D. Brunner, L. Appeltant, J.M. Gutierrez, L. Pesquera, C.R. Mirasso, I. Fischer, Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing. Opt. Express **20**, 3241 (2012)
Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature **521**, 436 (2015)
J.-G. Liu, L. Wang, Phys. Rev. A **98** (2018)
W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. **14**, 2531 (2002)
J.R. McClean et al., Barren plateaus in quantum neural network training landscapes. Nat. Commun. **9**, 4812 (2018)
K. Mitarai, K. Fujii, Methodology for replacing indirect measurements with direct measurements. Phys. Rev. Res. **1** (2019)
K. Mitarai et al., Quantum circuit learning. Phys. Rev. A **98** (2018)
T. Morimae, K. Fujii, J.F. Fitzsimons, Hardness of classically simulating the one-clean-qubit model. Phys. Rev. Lett. **112** (2014)
K. Nakajima, H. Hauser, R. Kang, E. Guglielmino, D.G. Caldwell, R. Pfeifer, Computing with a muscular-hydrostat system, in *Proceedings of 2013 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1496 (2013a)
K. Nakajima, H. Hauser, R. Kang, E. Guglielmino, D.G. Caldwell, R. Pfeifer, A soft body as a reservoir: case studies in a dynamic model of octopus-inspired soft robotic arm Front. Comput. Neurosci. **7**, 1 (2013b)
K. Nakajima, T. Li, H. Hauser, R. Pfeifer, Exploiting short-term memory in soft body dynamics as a computational resource. J. R. Soc. Interface **11**, 20140437 (2014)
K. Nakajima, H. Hauser, T. Li, R. Pfeifer, Information processing via physical soft body. Sci. Rep. **5**, 10487 (2015)
K. Nakajima et al., Boosting computational power through spatial multiplexing in quantum reservoir computing. Phys. Rev. Appl. **11** (2019)
M. Negoro et al., Machine learning with controllable quantum dynamics of a nuclear spin ensemble in a solid (2018), arXiv:1806.10910
M.A. Nielsen, I.L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2010)
Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, S. Massar, Optoelectronic reservoir computing. Sci. Rep. **2**, 287 (2012)
J. Preskill, Quantum computing in the NISQ era and beyond. Quantum **2**, 79 (2018)
M. Rabinovich, R. Huerta, G. Laurent, Transient dynamics for neural processing. Science **321**, 48 (2008)
J. Romero, A. Aspuru-Guzik, Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions (2019), arXiv:1901.00848
T.F. Rønnow, Z. Wang, J. Job, S. Boixo, S.V. Isakov, D. Wecker, J.M. Martinis, D.A. Lidar, M. Troyer, Defining and detecting quantum speedup. Science **345**, 420 (2014)
M. Schuld et al., Evaluating analytic gradients on quantum hardware. Phys. Rev. A **99** (2019)
M. Schuld et al., Circuit-centric quantum classifiers. Phys. Rev. A **101** (2020)
P.W. Shor, Algorithms for quantum computation: Discrete logarithms and factoring, in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, vol. 124 (1994)
H. Situ et al., Quantum generative adversarial network for generating discrete data. Inf. Sci. **538**, 193 (2020)
G.R. Steinbrecher et al., Quantum optical neural networks. NPJ Quantum Inf. **5**, 60 (2019)
A.Z. Stieg, A.V. Avizienis, H.O. Sillin, C. Martin-Olmos, M. Aono, J.K. Gimzewski, Emergent criticality in complex Turing B-type atomic switch networks. Adv. Mater. **24**, 286 (2012)

Q.H. Tran, K. Nakajima, Higher-order quantum reservoir computing (2020), arXiv:2006.08999
K. Vandoorne, P. Mechet, T.V. Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, P. Bienstman, Experimental demonstration of reservoir computing on a silicon photonics chip. Nat. Commun. **5**, 3541 (2014)
D. Verstraeten, B. Schrauwen, M. D'Haene, D. Stroobandt, An experimental unification of reservoir computing methods. Neural Netw. **20**, 391 (2007)
J.G. Vidal, D.O. Theis, Calculus on parameterized quantum circuits (2018), arXiv:1812.06323
C.M. Wilson et al., Quantum kitchen sinks: an algorithm for machine learning on near-term quantum computers (2018), arXiv:1806.08321
D. Woods, T.J. Naughton, Photonic neural networks. Nat. Phys. **8**, 257 (2012)
J. Zeng et al., Learning and inference on generative adversarial quantum circuits. Phys. Rev. A **99** (2019)
D. Zhu et al., Training of quantum circuits on a hybrid quantum computer. Sci. Adv. **5**, 9918 (2019)

Toward NMR Quantum Reservoir Computing

Makoto Negoro, Kosuke Mitarai, Kohei Nakajima, and Keisuke Fujii

Abstract Reservoir computing is a framework used to exploit natural nonlinear dynamics with many degrees of freedom, which is called a reservoir, for a machine learning task. Here we introduce the NMR implementation of quantum reservoir computing and quantum extreme learning machine using the nuclear quantum reservoir. The implementation utilizes globally controlled dynamics of nuclear spin qubits in solid state and it has been demonstrated.

The physical implementation of quantum information processing has been extensively studied with various quantum systems (Nielsen and Chuang 2000; Ladd et al. 2010). Any physical realization of the quantum computer must satisfy the following criteria: (1) a scalable physical system with well-characterized qubits, (2) the ability to initialize the state of the qubits, (3) a *universal* set of quantum gates, (4) long relevant decoherence times, much longer than the gate-operation time, and (5) a qubit-specific measurement capability (DiVincenzo 2000). To this day, there have been many proposals of implementations which satisfy these criteria, such as superconducting qubits, semiconductor quantum dots, trapped atoms, photonics, and nuclear magnetic resonance (NMR). One of the most promising candidates is the superconducting qubit system. In 2019, Google has announced a quantum computer employed

M. Negoro (✉) · K. Mitarai · K. Fujii
Center for Quantum Information and Quantum Biology, Osaka University, Toyonaka, Osaka 560-8531, Japan
e-mail: negoro@qc.ee.es.osaka-u.ac.jp

K. Fujii
e-mail: fujii@qc.ee.es.osaka-u.ac.jp

M. Negoro · K. Mitarai · K. Nakajima · K. Fujii
JST PRESTO, Kawaguchi, Saitama 332-0012, Japan

K. Mitarai · K. Fujii
Graduate School of Engineering Science, Osaka University, Toyonaka, Osaka 560-8531, Japan

K. Nakajima
Graduate School of Information Science and Technology, The University of Tokyo, Bunkyo-Ku, 113-8656 Tokyo, Japan

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_19

with 53 superconducting qubits (Arute et al. 2019) with which they have claimed to achieve *quantum computational supremacy* (Preskill 2012). Arrays of atoms trapped in the vacuum have also gained attention as good candidates for large-scale quantum computing and simulation (Mazurenko et al. 2017; Bernien et al. 2017; Zhang et al. 2017). However, it is not yet clear that these implementations will ultimately be successful.

NMR quantum information processing (QIP) has been proposed in 1996 by Chuang et al. (Gershenfeld and Chuang 1997) and Cory et al. (1997). In this approach, we use nuclear spins in a molecule as qubits. In a static magnetic field, spin-1/2 particles have two energy levels, each corresponding to the states parallel ($|\uparrow\rangle$) and anti-parallel ($|\downarrow\rangle$) to the magnetic field. These two quantum states can be used as $|0\rangle$ and $|1\rangle$ of a qubit. An organic molecule, which can have many nuclei with spin-1/2, forms a many-qubit system that can be used for QIP purposes. A usual NMR sample has a macroscopic ensemble of identical copies of such a many-qubit system. The ensemble-nature of NMR allows the direct measurement of an observable. Qubits are controlled by applying oscillating magnetic fields at radiofrequency. Gate operations between two qubits can be implemented by utilizing the naturally existing interaction between the spins. As NMR spectroscopy has been applied for analysis of materials for over 60 years, vast knowledge to control nuclear spins has been accumulated (Slichter 1990). Early day proof-of-principle QIP experiment is performed with NMR, owing to the nice controllability of the many-qubit system. It is hoped that future progress of macromolecular and supramolecular technology will make it possible to array Avogadro number of spins in two or three dimensions, let alone far more than 10^9 spins, which is required for fault-tolerant quantum computation to outperform classical computation (Jones et al. 2012).

The NMR quantum computer has succeeded in implementing Shor's algorithm using 7 qubits (Vandersypen et al. 2001), quantum simulation using 7 qubits (Negrevergne et al. 2005), and quantum machine learning (Biamonte et al. 2017) using 4 qubits (Li et al. 2015). However, the *pseudo-initialization technique* (Cory et al. 1997) used in these implementations is not scalable, which blocks the exponential speedup of the quantum algorithms. The state with partially initialized spins in these implementations is inevitably separable, that is, they do not have entanglement (Braunstein et al. 1998). In order to efficiently solve BQP (bounded error quantum polynomial time) type problems like factoring, a scalable initialization technique of nuclear spin qubits is required. Nuclear spins are only slightly polarized even in the strong magnetic fields conventionally used in NMR spectroscopy at room temperature because the magnetic energy of nuclear spin is much smaller than the thermal energy. One of the scalable initialization schemes is to use dynamic nuclear polarization (DNP) which is operated at very low temperature in solids (A. Abragam, M. Goldman, Nuclear Magnetism: Order and Disorder (Clarendon Press 1982). Solid state NMR quantum information processing experiments have also been studied. Solid state nuclear spin qubits have a long coherence time (Ladd et al. 2005) and have already been controlled with high fidelity although not initialized (Ryan et al. 2008; Alvalez et al. 2015). The initialization technique is not still compatible with high-fidelity control to this day, which motivates us to seek a way to benefit

from the solid state NMR quantum computer with partially polarized qubits without pseudo-initialization.

A class of problems that can be solved scalably by quantum computers with partially polarized states is called DQC1 (deterministic quantum computation with 1 qubit) (Knill and Laflamme 1998). Recently, it has been proven theoretically that DQC1 is unsimulatable with classical computers under stable complexity conjectures (Morimae et al. 2014). Owing to the theoretical progress, now we can provide a roadmap to scale up the NMR quantum computer by first showing the quantum computational supremacy with partially polarized spins, DQC1 type, then with initialized spins. There are various oligomers and polymers with more than 50 spins. The simulation of ensemble dynamics of more than 50 qubits is computationally hard for classical computers whether the state is initialized or partially polarized. Quantum simulation experiments (Alvalez et al. 2015) have demonstrated that the controllable dynamics of partially polarized nuclear spins in molecular solids can be highly complex as more than 1000 spins are quantum mechanically correlated.

We experimentally demonstrated how to exploit such complex quantum dynamics of a nuclear spin ensemble for machine learning (Negoro et al. 2018). To this end, a quantum reservoir framework (QRC) is employed (Fujii and Nakajima 2017). Reservoir computing provides a framework for exploiting nonlinear dynamics with many degrees of freedom, called a reservoir, for machine learning (Maass et al. 2002; Jaeger and Haas 2004; Varsraeten et al. 2007). It was first proposed as an echo-state network or liquid-state machine (Maass et al. 2002; Jaeger and Haas 2004), where a conventional neural network is used as a reservoir, but its internal dynamics are randomly prefixed and only the linear readout weights are optimized to learn a nonlinear task. This black-box property allows for the use of actual physical systems that employ photonics (Vandoorne et al. 2013), spintronics (Torrejon et al. 2017), or soft robotics (Nakajima et al. 2015), as well as qubits (Fujii and Nakajima 2017). Originally proposed implementation of QRC requires the initialized spins and also the on-demand initialization in the protocol, which are not yet possible in actual experiments. To avoid this difficulty, we proposed a slightly modified algorithm together with the experimental realization (Negoro et al. 2018). Here we introduce the experimental proposal.

Figure 1a shows the physical implementation of a quantum reservoir consisting of nuclear spins with a partially polarized state in a molecular solid. In this implementation, all B spins are controlled with the same oscillating magnetic field, which results in a *global control* scheme (Benjamin and Bose 2003), where each of B spins is controlled in an equivalent way. At the first sight, this scheme might seem not sufficient for the universal quantum computation, but nonetheless, it has been shown that universal quantum circuits can be implemented with global control (Benjamin and Bose 2003; Lloyd 1993). Topological error correction can also be implemented with global control and on-demand initialization of qubits (Fujii et al. 2014). As a proof-of-principle experimental demonstration, we have performed non-temporal tasks using the natural quantum dynamics of the quantum reservoir, which we also refer to as quantum extreme learning machine (QELM), a quantum counterpart of extreme learning machine (see also another chapter written by Fujii and Nakajima

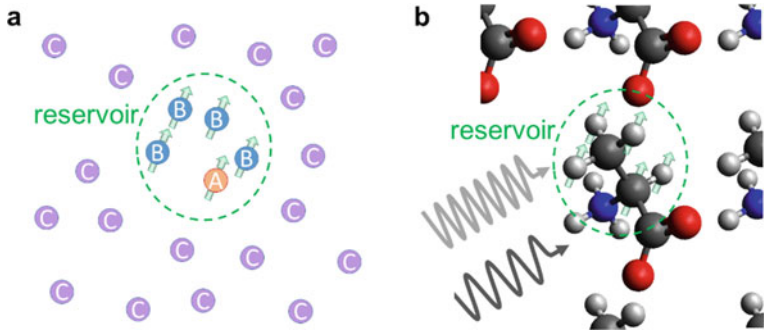


Fig. 1 NMR quantum reservoir (**a**) and its implementation with *l*-alanine (**b**)

in this book) (Butcher et al. 2013). Related quantum algorithms have been recently proposed (C.M. Wilson et al. 2018; Halvicek et al. 2019). In the experimental demonstration of QELM, we used isotopically labeled *l*-alanine (Negoro et al. 2018). This molecule has four ^{1}H spins and one ^{13}C spin, both of which have spin-1/2. We diluted it into a single crystal of $^{2}H_7$-*l*-alanine which has 7 ^{2}H spins as shown in Fig. 1b. ^{1}H, ^{13}C, and ^{2}H spins correspond to B, A, and C spins, respectively, and ^{1}H and ^{13}C spins are used as reservoir.

The quantum circuits of QELM and QRC are shown in Fig. 2. The quantum gate U, which is the source of nonlinearity of the output from the reservoir, is implemented as a time evolution that naturally arises from the interaction among the spins. Throughout the execution of the quantum circuit, the C spins do not interact with A and B spins. Accordingly, the ensemble reservoir is well isolated. In order to control the reservoir dynamics for input, we employed the selective rotation of A (Fig. 2a) or the global rotation of B spins (Fig. 2b). Input data can be fed into the reservoir using the selective rotation of A (Fig. 2a) or the global rotation of B spins (Fig. 2b). Our experimental demonstration uses the circuit of Fig. 2b. The initial state of QELM is a partially polarized state and that of QRC is initialized. In addition, the B spin is initialized by interacting with initialized bath spins, as implemented in Ryan et al. (2008), before every input (Fig. 2c).

QELM is a framework to perform non-temporal nonlinear tasks such as classification and pattern recognition with finite inputs. In the demonstration of QELM (Negoro et al. 2018), we consider the learning problem of a nonlinear function $y = f(s)$ using K training data sets including an input stream with the length of L, $\{s_{l,k}\}$ ($s_{l,k} \in [0,1]$), and corresponding teacher t_k. The input is processed by M cycles of the time evolution U, each of which is followed by the ensemble measurement of A spins. We repeat L series of the input process and cycles of evolution for each of K instances of the input $\{s_{l,k}\}$. Thus, we get a total of LM signals $x_{l,m}^{(k)}$ for a given input instance $\{s_{l,k}\}$. The output from the machine is defined as a weighted sum of the LM signals $x_{l,m}^{(k)}$:

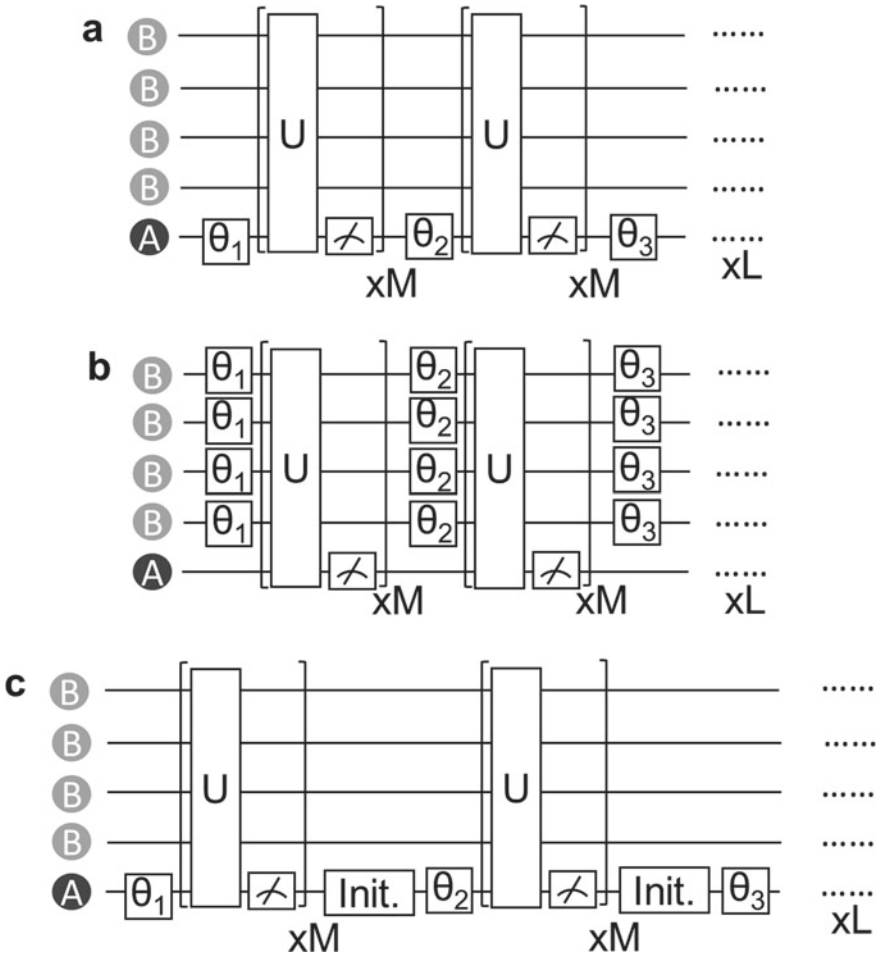


Fig. 2 Quantum circuits of QELM with selective rotation (**a**) and global rotation (**b**) and QRC (**c**)

$$y_k = \sum_{m=1}^{M} \sum_{l=1}^{L} W_{l,m} x_{l,m}^{(k)}, \tag{1}$$

where W is an LM-dimensional weight vector. After taking signals from K sets of training data, W is determined so as to minimize the mean squared error, $\Sigma_k (t_k - y_k)^2$.

We evaluated the computational capabilities of this QELM implementation with two benchmark tasks under a binary sequential input. One is the input recognition task, which measures how well each input is reconstructed from the quantum reservoir dynamics. Another is the parity check task, which tests how well the machine can perform a nonlinear transformation of the input. We found our implementation can perform both of the two. We further demonstrate to train the machine to compute

Table 1 Performance of learning functions

	MSE	# of error
2bit XOR	1.97×10^{-2}	0
3bit XOR	2.83×10^{-2}	0
4bit XOR	1.99×10^{-1}	3
NAND	2.26×10^{-2}	0
1bit Adder 0th order	2.26×10^{-2}	0
1bit Adder 1st order	2.01×10^{-2}	0
2bit Adder 0th order	4.36×10^{-2}	1
2bit Adder 1st order	2.02×10^{-1}	4
2bit Adder 2nd order	1.44×10^{-2}	0
Multiplication	2.32×10^{-3}	–
Division	5.22×10^{-4}	–
Nonlinear I	3.09×10^{-4}	–
Nonlinear II	7.64×10^{-3}	–

simple Boolean functions of input binary strings, which are nonlinear functions when the binary information is embedded into a continuous variable. The performance of learning functions measured by the mean squared error and the number of erroneous outputs are shown in Table 1. These experimental results are detailed and discussed in Negoro et al. (2018).

If we wish to perform QRC, we would have to initialize both qubits, namely in the above implementation, the C spins. For example, as mentioned earlier, nuclear spin qubits can be initialized by DNP which is performed at very low temperature of ~1 K (de Boer and Niinikoski 1974). For such a scheme, technological development for high-fidelity control at very low temperature is important (Cho et al. 2007). Alternatively, we can also develop and advance the initialization technique at room temperature (Tateishi et al. 2014), where nuclear spins can be controlled very accurately. In fact, we already have achieved the room-temperature initialization of up to 34% using photoexcitation of electrons (Tateishi et al. 2014). Initialization technique that is compatible with high-fidelity control is within reach. In Ryan et al. (2008), the error per gate is approximately 1% in solid state at room temperature. The other important direction is to boost computational power by increasing the number of the reservoir spins, as numerically demonstrated in Fujii and Nakajima (2017) and Nakajima et al. (2019).

Our implementation of the quantum reservoir and demonstration of nonlinear information processing therein pave the way for exploiting quantum computational supremacy in NMR ensemble systems for information processing with reachable technologies.

Acknowledgements M.N. is supported by JST PRESTO Grant Number JPMJPR15E7. K.M. is supported by JSPS KAKENHI No. 19J10978. We are supported by MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) Grant Number JPMXS0118067394. We acknowledge Masahiro Kitagawa.

References

A. Abragam, M. Goldman, Nuclear Magnetism: Order and Disorder (Clarendon Press, 1982)
G.A. Alvalez, D. Suter, R. Kaiser, Science **349**, 846 (2015)
F. Arute et al., Nature **574**, 505 (2019).
S.C. Benjamin, S. Bose, Phys. Rev. Lett. **90**, 247901 (2003)
H. Bernien et al., Nature **551**, 579–584 (2017)
J. Biamonte et al., Nature **549**, 195–202 (2017)
S.L. Braunstein et al., Phys. Rev. Lett. **83**, 1054–1057 (1998)
J.B. Butcher, D. Verstraeten, B. Schrauwen, C.R. Day, P.W. Haycock, Neural Netw. **38**, 76 (2013)
H. Cho, J. Baugh, C.A. Ryan, D.G. Cory, C. Ramanathan, J. Magn. Reson. **187**, 242 (2007)
D.G. Cory, A.F. Fahmy, T.F. Havel, Proc. Nat. Acad. Sci. U. S. A. **94**, 1634 (1997)
W. de Boer, T.O. Niinikoski, Nucl. Instrum. Methods **114**, 495 (1974)
D.P. DiVincenzo, Fortsch. Phys. **48**, 771–783 (2000)
K. Fujii, K. Nakajima, Phys. Rev. Appl. **8**, 024030 (2017)
K. Fujii, M. Negoro, N. Imoto, M. Kitagawa, Phys. Rev. X **4**, 041039 (2014)
N.A. Gershenfeld, I.L. Chuang, Science **275**, 350 (1997)
V. Halvicek et al., Nature **567**, 209–212 (2019)
H. Jaeger, H. Haas, Science **304**, 78 (2004)
N.C. Jones, R. Van Meter, A.G. Fowler, P.L. McMahon, J. Kim, T.D. Ladd, Y. Yamamoto, Phys. Rev. X **2**, 031007 (2012)
E. Knill, R. Laflamme, Phys. Rev. Lett. **81**, 5672 (1998)
T.D. Ladd, D. Maryenko, Y. Yamamoto, E. Abe, K.M. Itoh, Phys. Rev. B **71**, 014401 (2005)
T.D. Ladd et al., Nature **464**, 45–53 (2010)
Z. Li, X. Liu, N. Xu, J. Du, Phys. Rev. Lett. **114**, 140504 (2015)
S. Lloyd, Science **261**, 1569–1571 (1993)
W. Maass, T. Natschlager, H. Markram, Neural Comput. **14**, 2531 (2002)
A. Mazurenko et al., Nature **545**, 462–466 (2017)
T. Morimae, K. Fujii, J.F. Fitzsimons, Phys. Rev. Lett. **112**, 130502 (2014)
K. Nakajima et al., Sci. Rep. **5**, 10487 (2015)
K. Nakajima, K. Fujii, M. Negoro, K. Mitarai, M. Kitagawa, Phys. Rev. Appl. **11**, 034021 (2019)
M. Negoro, K. Mitarai, K. Fujii, K. Nakajima, M. Kitagawa (2018), https://arxiv.org/abs/1806.10910
C. Negrevergne et al., Phys. Rev. A **71**, 032344 (2005)
M.A. Nielsen, I.L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000)
J. Preskill (2012), https://arxiv.org/abs/1203.5813
C.A. Ryan, O. Moussa, J. Baugh, R. Laflamme, Phys. Rev. Lett. **100**, 140501 (2008)
C.P. Slichter, *Principles of Magnetic Resonance*, Third Enlarged and Updated edn. (Springer, Berlin, 1990)
K. Tateishi et al., Proc. Natl. Acad. Sci. U. S. A. **111**, 7527 (2014)

J. Torrejon et al., Nature **547**, 428 (2017)
L.M.K. Vandersypen et al., Nature **414**, 883–887 (2001)
K. Vandoorne et al., Nat. Commun. **4**, 1364 (2013)
D. Varsraeten, B. Schrauwen, M.D. Haene, D. Stroobandt, Neural Netw. **20**, 391 (2007). 5
C.M. Wilson et al. (2018), https://arxiv.org/abs/1806.08321
J. Zhang et al., Nature **551**, 601–604 (2017)