Diversifying Non-dissipative Reservoir Computing Dynamics

Claudio Gallicchio(✉)

Department of Computer Science, University of Pisa,
Largo Bruno Pontecorvo, 3, Pisa, Italy
gallicch@di.unipi.it

Abstract. The Euler State Network (EuSNs) model is a recently proposed Reservoir Computing methodology that provides stable and non-dissipative untrained dynamics by discretizing an appropriately constrained ODE. In this paper, we propose alternative formulations of the reservoirs for EuSNs, aiming at improving the diversity of the resulting dynamics. Our empirical analysis points out the effectiveness of the proposed approaches on a large pool of time-series classification tasks.

Keywords: Euler State Networks · Reservoir Computing · Echo State Networks

1 Introduction

Reservoir Computing (RC) [17,19,22] is a popular technique for efficiently training Recurrent Neural Networks (RNNs) by utilizing the stable neural dynamics of a fixed recurrent reservoir layer and a trainable readout for output computation. This approach has been successful in various applications, in particular for implementing distributed learning functionalities in embedded systems [2,3,6] and as a reference paradigm for neuromorphic hardware implementations of recurrent neural models [20,21].

The effective operation of RC neural networks depends largely on the stability of its dynamics, which can be achieved through a global asymptotic stability property known as the Echo State Property in the widely used Echo State Network (ESN) model [14,15]. This property ensures that the dynamics of the reservoir remain stable, while at the same time it limits its memory and state-space structure, thus preventing the transmission of input information across multiple time steps.

Recently, a new approach to overcome the limitations of fading memory in standard ESNs has been proposed, which involves discretizing an Ordinary Differential Equation (ODE) while ensuring stability and non-dissipative constraints. This approach computes the reservoir dynamics as the forward Euler

This work is partially supported by the EC H2020 programme under project TEACHING (grant n. 871385), and by EMERGE, a project funded by EU Horizon research and innovation programme (grant n. 101070918).

L. Iliadis et al. (Eds.): ICANN 2023, LNCS 14261, pp. 172–183, 2023.
https://doi.org/10.1007/978-3-031-44198-1_15

solution of an ODE, hence the resulting model is called the *Euler State Network* (EuSN) [7,9]. As their dynamics are neither unstable nor lossy, EuSNs are capable of preserving input information over time, making them better suited than ESNs for tasks involving long-term memorization. The EuSN approach has already been shown to exceed the accuracy of ESNs and achieve comparable performance levels to fully trainable state-of-the-art RNN models on time-series classification tasks, while still maintaining the efficiency advantage of RC [9]. At the same time, the study of the architectural organization of the EuSN reservoir system is still largely unexplored.

In this paper, we deepen the analysis of EuSN architectures and propose ways to improve the diversification of reservoir dynamics. Our first proposal is to introduce a variability factor by using different integration rates in different reservoir neurons. The second variability factor is to consider different diffusion coefficients, which result in different strengths for the self-feedback connections in the reservoir neurons. We analyze the effects of these factors, individually and in synergy, on the resulting dynamical characterization of the reservoir system and in a wide range of experiments on time-series classification benchmarks.

The rest of this paper is organized as follows. In Sect. 2 we summarize the fundamental aspects of the RC methodology and of the popular ESN model, while in Sect. 3 we introduce the crucial concepts behind non-dissipative RC dynamics and the EuSN model. Then, in Sect. 4, we illustrate the proposed approach to enhance the diversification of reservoir dynamics in EuSNs. Our empirical analysis on several time-series classification benchmarks is given in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Reservoir Computing

Reservoir Computing (RC) [17,22] refers to a category of efficiently trainable recurrent neural models in which the internal connections pointing to the hidden recurrent layer, the *reservoir*, are left untrained after randomization subject to asymptotic stability constraints. The neural architecture is then completed by an output layer, the *readout*, which is the only trained component of the model. Within such a class, we introduce the popular Echo State Network (ESN) [14,15] model, which employs the tanh non-linearity and operates in discrete time-steps.

To set our notation, let us consider a reservoir that comprises N_h neurons, and that is stimulated by a driving (external) N_x-dimensional input signal. Accordingly, we denote the reservoir state and the input at time step t respectively as $\mathbf{h}(t) \in \mathbb{R}^{N_h}$, and $\mathbf{x}(t) \in \mathbb{R}^{N_x}$. We refer to the general case of leaky integrator ESNs [16], and describe the dynamical operation of the reservoir by the following iterated map:

$$\mathbf{h}(t) = (1-\alpha)\,\mathbf{h}(t-1) + \alpha\,\tanh(\mathbf{W}_h\,\mathbf{h}(t) + \mathbf{W}_x\,\mathbf{x}(t) + \mathbf{b}), \qquad (1)$$

where $\mathbf{W}_h \in \mathbb{R}^{N_h \times N_h}$ is the reservoir recurrent weight matrix, $\mathbf{W}_x \in \mathbb{R}^{N_h \times N_x}$ is the input weight matrix, $\mathbf{b} \in \mathbb{R}^{N_h}$ is the bias vector, and $\tanh(\cdot)$ denotes the element-wise applied hyperbolic tangent non-linearity. Moreover, $\alpha \in (0,1]$

represents the leaking rate hyper-parameter, influencing the relative speed of reservoir dynamics with respect to the dynamics of the input. Before being driven by the external input signal $\mathbf{x}(t)$, the reservoir state is typically initialized in the origin, i.e., $\mathbf{h}(0) = \mathbf{0}$.

After their initialization, the weight values of $\mathbf{W}_h$, $\mathbf{W}_x$, and $\mathbf{b}$ are kept fixed in accordance with the Echo State Property (ESP) [23], which ensures global asymptotic stability of the reservoir dynamical system. In practice, the recurrent weights in $\mathbf{W}_h$ are typically randomly drawn from a uniform distribution over $(-1, 1)$ and then adjusted to limit the resulting spectral radius[1] $\rho(\mathbf{W}_h)$ to values smaller than 1. The value of $\rho(\mathbf{W}_h)$ has a direct influence on the dynamical properties of the resulting reservoir, and in particular on the extent of its fading memory. As such, it is a crucial hyper-parameter of the ESN model. As the spectral radius re-scaling of (a potentially large) matrix $\mathbf{W}_h$ can represent a computational bottleneck, in this paper we resort to an efficient initialization scheme introduced in [10], which leverages results in random matrix theory to provide a fast initialization of the recurrent weights in $\mathbf{W}_h$. The input weight matrix $\mathbf{W}_x$ and bias vector $\mathbf{b}$ are also randomly initialized, and then re-scaled to control their magnitude. A widely used approach consists in drawing their values from uniform distributions over $(-\omega_x, \omega_x)$ and $(-\omega_b, \omega_b)$, respectively, where ω_x and represents the input scaling hyper-parameter, and ω_b is the bias scaling hyper-parameter.

The ESN architecture also includes a trainable dense readout layer which, in the case of time-series classification tasks, is fed by the last reservoir state corresponding to each input time series. As the reservoir parameters are kept fixed, the readout is often trained in closed form [17], e.g., by pseudo-inversion or ridge regression.

Finally, it is worth remarking that the ESN model relies on the ESP stability property to regulate the reservoir dynamics. This property ensures that when the network is fed with a long input time-series, the initial state conditions eventually fade away, and the state encoding produced by the reservoir becomes stable. However, this characterization is linked to the fading memory and suffix-based Markovian organization of the reservoir state space (see, e.g., [8,11,13]). These properties make it difficult to transfer information across multiple time-steps, limiting the effectiveness of ESNs for tasks that require long-term memory retention of the input information.

3 Non-dissipative Reservoir Computing

To overcome the limitations of a fading memory reservoir system, an alternative approach based on discretizing ODEs subject to stability and non-dissipativity conditions has recently been proposed [7,9]. The resulting RC model is derived from the continuous-time dynamics expressed by the following ODE:

$$\mathbf{h}'(t) = \tanh(\mathbf{W}_h \mathbf{h}(t) + \mathbf{W}_x \mathbf{x}(t) + \mathbf{b}), \tag{2}$$

[1] The spectral radius of a matrix $\mathbf{A}$ is defined as the maximum length of an eigenvalue of $\mathbf{A}$.

requiring that the corresponding Jacobian has eigenvalues with ≈ 0 real parts. In addition to stability, such a critical condition implies non-dissipative system dynamics, which can be leveraged to effectively propagate the input information over multiple time-steps [5,12]. Crucially, the requested condition on the eigenvalus of the Jacobian of Eq. 2, can be easily met *architecturally* by the use of an antisymmetric recurrent weight matrix, i.e. requiring that $\mathbf{W}_h = -\mathbf{W}_h^T$. In such a case, indeed, the eigenvalues of both $\mathbf{W}_h$ and the Jacobian are on the imaginary axis (see, e.g., [5,9] for further details). Interestingly, this property does not need to be learned from data, rather it can be enforced in the neural processing system by design. In other words, provided that the antisymmetric condition holds, the recurrent weight matrix $\mathbf{W}_h$ can be initialized with random weights and then left untrained, as in standard RC approaches. Finally, the resulting constrained ODE system is discretized by Euler forward method, yielding the following state transition equation ruling the behavior of a discrete-time recurrent neural layer:

$$\mathbf{h}(t) = \mathbf{h}(t-1) \; + \varepsilon \tanh\Big((\mathbf{W}_h - \gamma \mathbf{I})\mathbf{h}(t-1) + \mathbf{W}_x \mathbf{x}(t) + \mathbf{b}\Big), \qquad (3)$$

where $\mathbf{W}_h = -\mathbf{W}_h^T$ is the antisymmetric recurrent weight matrix, while ε and γ are two (typically small) positive hyper-parameters that represents respectively the *step size* of integration, and the *diffusion* coefficient used to stabilize the discretization [12]. As in standard ESNs, the weight values in $\mathbf{W}_h$, $\mathbf{W}_x$ and $\mathbf{b}$ are left untrained after initialization, and the resulting RC model is named *Euler State Network* (EuSN). In particular, the weight values in $\mathbf{W}_h$ in Eq. 3 can be obtained starting from a random matrix $\mathbf{W}$ whose elements are drawn from a uniform distribution in $(-\omega_r, \omega_r)$, with ω_r representing a recurrent scaling hyper-parameter, and then setting $\mathbf{W}_h = \mathbf{W} - \mathbf{W}^T$, which grants the antisymmetric property. The weight values in $\mathbf{W}_x$ and $\mathbf{b}$ are initialized as described in Sect. 2 for ESNs. Moreover, as in standard ESNs, the state is initialized in the origin, i.e., $\mathbf{h}(0) = \mathbf{0}$, and the neural network architecture is completed by a readout layer that is the only trained component of the model. It has already been shown in the literature that the EuSN model is extremely efficient at propagating input information across many time steps, providing an exceptional trade-off between complexity and accuracy in time-series classification tasks. Overall, EuSNs make it possible to retain the efficiency typical of untrained RC networks while achieving - and even exceeding - the accuracy of fully trained recurrent models (see [7,9] for an extended comparison in this regard). In this paper, starting from the basic EuSN model, we show how its dynamics can be enriched by simple architectural modifications that affect the variety of its dynamic behavior.

4 Diversifying Dynamics in Euler State Networks Reservoirs

We start our analysis by noting that the reservoir system of an EuSN, as described in Eq. 3 has an effective spectral radius intrinsically close to unity.

In fact, using standard arguments in RC area, we can observe how the Jacobian of the system in Eq. 3, analyzed around the origin and for null input, takes the following form:

$$\mathbf{J} = (1 - \varepsilon\,\gamma)\,\mathbf{I} + \varepsilon\,\mathbf{W}_h, \tag{4}$$

whose eigenvalues have a fixed real part, given by $1 - \varepsilon\gamma$, and imaginary part given by a small perturbation of one of the eigenvalues of $\mathbf{W}_h$. Using $\lambda_k(\cdot)$ to denote the k-th eigenvalue of its matrix argument, we have that:

$$\lambda_k(\mathbf{J}) = 1 - \varepsilon\gamma + i\;\varepsilon\beta_k, \tag{5}$$

where $\beta_k = Im(\lambda_k(\mathbf{W}_h))$. All eigenvalues are thus concentrated (vertically in the Gaussian plane) in a neighborhood of $1 - \varepsilon\gamma$. Given that both ε and γ take small positive values, we can notice that all the eigenvalues in Eq. 5 are close to 1 by design, and the eigenvalues of $\mathbf{W}_h$ have only a minor perturbation impact. This is illustrated in Fig. 1 (top, left). As analyzed in [9], this characterization can be interpreted as an architectural bias of the EuSN model towards critical dynamics. Notice that this bias is fundamentally different from the suffix-based Markovian nature of reservoir dynamics typical of the conventional ESN [8].

Despite the application success of the EuSN model already in its original form (as evidenced by the results in [7,9]), the dynamic characterization of the model seems to be improvable. In particular, while one of the keys to the success of RC is that it can cover a wide range of dynamic behaviour by randomizing the reservoir parameters, in the case of EuSNs randomization does not seem to be fully exploited. This can be seen firstly from the squeezing of the Jacobian eigenvalues on a line, and secondly from the observation that the reservoir state transition function in Eq. 3 contains a self-loop term modulated by the same γ value for all neurons. Accordingly, in the following, we introduce variants of the basic EuSN model in which different neurons can have different values of the step size parameter ε and the diffusion parameter γ.

Step Size Variability. We consider EuSN reservoir neurons with different values of the step size. The resulting state transition function is given by:

$$\mathbf{h}(t) = \mathbf{h}(t-1)\; + \boldsymbol{\varepsilon} \odot \tanh\Big((\mathbf{W}_h - \gamma\mathbf{I})\mathbf{h}(t-1) + \mathbf{W}_x\mathbf{x}(t) + \mathbf{b}\Big), \tag{6}$$

where $\boldsymbol{\varepsilon} \in \mathbb{R}^{N_h}$ is a vector containing the step size of integration of the different neurons, and $\odot$ denotes component-wise (Hadamard) multiplication. As an effect of this modification, the neurons in the EuSN reservoir exhibit dynamics with variable integration speed, potentially offering greater richness to the encoding produced by the system. Moreover, the resulting Jacobian is given by:

$$\mathbf{J} = diag(\mathbf{1} - \gamma\boldsymbol{\varepsilon}) + diag(\boldsymbol{\varepsilon})\mathbf{W}_h, \tag{7}$$

where $diag(\cdot)$ indicates a diagonal matrix with specified diagonal elements, and $\mathbf{1} \in \mathbb{R}^{N_h}$ is a vector of ones. The resulting eigenvalues are no longer characterized by the same real part, and present a more varied configuration, as illustrated in

Fig. 1 (top, right). In the following, we use EuSN-ε to refer to an EuSN network whose reservoir is ruled by Eq. 6.

Diffusion Variability. We consider EuSN reservoir neurons with different values of the diffusion coefficient. In this case, the state transition function is given by:

$$\mathbf{h}(t) = \mathbf{h}(t-1) \ + \varepsilon \tanh\Big((\mathbf{W}_h - diag(\boldsymbol{\gamma}))\mathbf{h}(t-1) + \mathbf{W}_x\mathbf{x}(t) + \mathbf{b}\Big), \quad (8)$$

where $\boldsymbol{\gamma} \in \mathbb{R}^{N_h}$ is a vector containing the diffusion term of the different neurons. Differently from the previous case of EuSN-ε, all the reservoir neurons operate at the same integration speed, but the reservoir topology is enriched by different strengths of the self-loops. The resulting Jacobian is given by:

$$\mathbf{J} = diag(\mathbf{1} - \varepsilon\boldsymbol{\gamma}) + \varepsilon\,\mathbf{W}_h, \quad (9)$$

whose eigenvalues variability is illustrated in Fig. 1 (bottom, left). In the following, EuSN-γ is used to refer to an EuSN network whose reservoir is described by Eq. 8.

Full Variability. We finally introduce an EuSN in which each reservoir neuron presents its own step size of integration and diffusion coefficient. This configuration includes both variability factors introduced by EuSN-γ and EuSN-γ, and is denoted by EuSN-ε, γ. In this case, the reservoir state transition function reads as follows:

$$\mathbf{h}(t) = \mathbf{h}(t-1) \ + \boldsymbol{\varepsilon} \odot \tanh\Big((\mathbf{W}_h - diag(\boldsymbol{\gamma}))\mathbf{h}(t-1) + \mathbf{W}_x\mathbf{x}(t) + \mathbf{b}\Big), \quad (10)$$

and the resulting Jacobian is given by:

$$\mathbf{J} = diag(\mathbf{1} - \boldsymbol{\varepsilon} \odot \boldsymbol{\gamma}) + diag(\boldsymbol{\varepsilon})\,\mathbf{W}_h, \quad (11)$$

The reservoir exhibits both dynamics with multiple scales of integration speed and diverse self-loops. Moreover, while preserving the architectural bias toward eigenvalues of the Jacobian near 1, these show wider variability, as illustrated in Fig. 1 (bottom, right).

5 Experiments

We have experimentally evaluated the performance of the proposed EuSN variants (introduced in Sect. 4), in comparison to the base EuSN setup (described in Sect. 3) and the conventional ESN model (described in Sect. 2).

Datasets. The performed analysis involved experiments on a large pool of diverse time-series classification benchmarks. The first 20 datasets have been taken from the UEA & UCR time-series classification repository [4], namely:

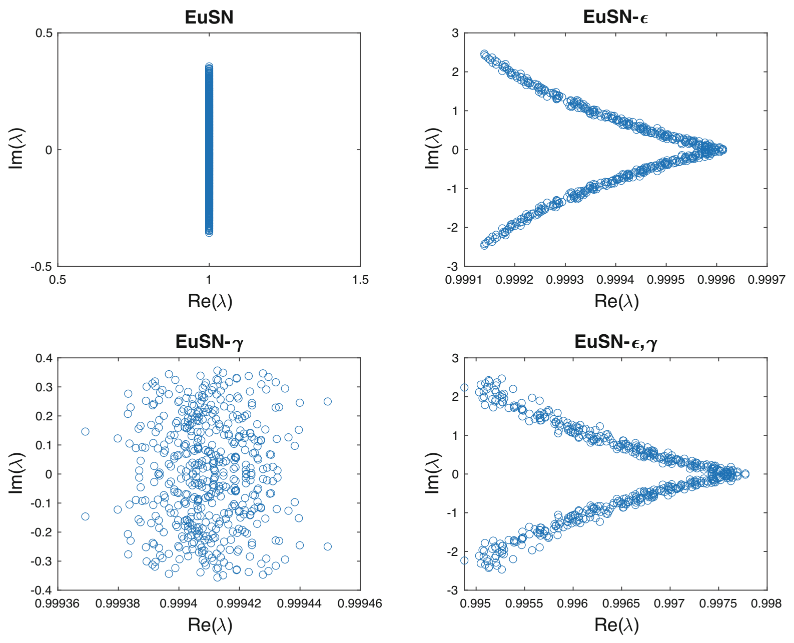

Fig. 1. Eigenvalues of the Jacobian for a 500-dimensional reservoir in EuSN (top left), EuSN-ε (top right), EuSN-γ (bottom left), and EuSN-ε, γ (bottom right). The plots correspond to a system with $\omega_r = 1$, $\varepsilon = 0.01$, $\gamma = 0.01$. Variable values of the step size were randomly sampled from a uniform distribution on $[\varepsilon, \varepsilon + 0.1]$. Variable values of the diffusion were randomly sampled from a uniform distribution on $[\gamma, \gamma + 0.1]$.

Adiac, Blink, CharacterTrajectories, Computers, Cricket, ECG5000, Epilepsy, FordA, FordB, HandOutlines, HandMovementDirection, Handwriting, Hearthbeat, KeplerLightCurves, Libras, Lightning2, Mallat, MotionSenseHAR, ShapesAll, Trace, UWaveGestureLibraryAll, Wafer, and Yoga. We have also run experiments on the *IMDB* movie review sentiment classification dataset [18], and on the *Reuters* newswire classification dataset from UCI [1], which were used in the publicly online available forms[2]. For these two tasks, we applied a preprocessing step in order to represent each sentence by a time series of 32-dimensional word embeddings[3]. For all datasets, we used the original splits into training and test,

[2] IMDB: https://keras.io/api/datasets/imdb/.
Reuters: https://keras.io/api/datasets/reuters/.

[3] For each dataset individually, every sentence was represented by a sequence of words from the 10k most frequent ones in the corresponding database, with a truncation to a maximum length of 200. To obtain the word embeddings, we trained an MLP network with a preliminary embedding layer of 32 units, followed by a hidden layer of 128 units with ReLU activation, and finally by a dense output layer. The MLP architecture was trained on the training set using the RMSProp optimizer for 100 epochs and early stopping with patience = 10 (on a validation set containing the

Table 1. Information on the time-series classification benchmarks used in our experiments, including the number of sequences in the training set (# Seq Tr) and in the test set (# Seq Ts), the maximum length of a sequence in the dataset (Length), the number of input features (Feat.), and the number of output classes (Classes).

Name	# Seq Tr	# Seq Ts	Length	Feat.	Classes
Adiac	390	391	176	1	37
Blink	500	450	510	4	2
CharacterTrajectories	1422	1436	182	3	20
Computers	250	250	720	1	2
Cricket	108	72	1197	6	12
ECG5000	500	4500	140	1	5
FordA	3601	1320	500	1	2
FordB	3636	810	500	1	2
HandOutlines	1000	370	2709	1	2
HandMovementDirection	160	74	400	10	4
Handwriting	150	850	152	3	26
Heartbeat	204	205	405	61	2
IMDB	25000	25000	200	32	2
KeplerLightCurves	920	399	4767	1	7
Libras	180	180	45	2	15
Lightning2	60	61	637	1	2
Mallat	55	2345	1024	1	8
MotionSenseHAR	966	265	1000	12	6
Reuters	8982	2246	200	32	46
ShapesAll	600	600	512	1	60
Trace	100	100	275	1	4
UWaveGestureLibraryAll	896	3582	945	1	8
Wafer	1000	6164	152	1	2
Yoga	300	3000	426	1	2

applying a further 67% - 33% stratified splitting of the original training data into training and validation sets. Relevant information on the used datasets is reported in Table 1.

Experimental Settings. In our experiments, we considered EuSN with a number of recurrent units N_h ranging between 10 and 500. We explored values of ω_r, ω_x and ω_b in $\{10^{-3}, 10^{-2}, \ldots, 10\}$, ε and γ in $\{10^{-5}, 10^{-4}, \ldots, 1\}$. For EuSN set-

33% of the original training data). After this, the output of the embedding layer for each sentence in the dataset was used as input feature in our experiments with the RC models.

Table 2. Results on the time-series classification benchmarks. For every task, it is reported the accuracy on the test set achieved by ESN, EuSN, EuSN with variable step size (EuSN-ε), EuSN with variable diffusion (EuSN-γ), and EuSN with both variable step size and diffusion (EuSN-ε, γ). Results are averaged (and std are given) over 10 random guesses. Best results for each task are highlighted in bold.

Task	ESN	EuSN	EuSN-ε	EuSN-γ	EuSN-ε, γ
Adiac	$0.307_{\pm 0.07}$	$0.690_{\pm 0.01}$	$\mathbf{0.691}_{\pm 0.01}$	$0.634_{\pm 0.01}$	$0.649_{\pm 0.01}$
Blink	$0.620_{\pm 0.02}$	$0.943_{\pm 0.01}$	$0.934_{\pm 0.01}$	$0.946_{\pm 0.01}$	$\mathbf{0.969}_{\pm 0.01}$
CharacterTrajectories	$0.964_{\pm 0.00}$	$\mathbf{0.993}_{\pm 0.00}$	$0.989_{\pm 0.00}$	$0.989_{\pm 0.00}$	$0.985_{\pm 0.00}$
Computers	$0.652_{\pm 0.00}$	$0.638_{\pm 0.02}$	$0.707_{\pm 0.01}$	$\mathbf{0.716}_{\pm 0.01}$	$0.607_{\pm 0.02}$
Cricket	$0.976_{\pm 0.01}$	$0.933_{\pm 0.02}$	$0.993_{\pm 0.01}$	$\mathbf{1.000}_{\pm 0.00}$	$\mathbf{1.000}_{\pm 0.00}$
ECG5000	$0.921_{\pm 0.00}$	$\mathbf{0.938}_{\pm 0.00}$	$0.932_{\pm 0.00}$	$0.937_{\pm 0.00}$	$\mathbf{0.938}_{\pm 0.00}$
FordA	$0.591_{\pm 0.02}$	$0.691_{\pm 0.01}$	$0.656_{\pm 0.01}$	$0.677_{\pm 0.01}$	$\mathbf{0.700}_{\pm 0.02}$
FordB	$0.519_{\pm 0.00}$	$\mathbf{0.652}_{\pm 0.01}$	$0.639_{\pm 0.01}$	$0.555_{\pm 0.02}$	$0.645_{\pm 0.01}$
HandOutlines	$0.690_{\pm 0.02}$	$0.912_{\pm 0.01}$	$0.908_{\pm 0.00}$	$\mathbf{0.919}_{\pm 0.00}$	$0.911_{\pm 0.00}$
HandMovementDirection	$0.551_{\pm 0.03}$	$0.585_{\pm 0.03}$	$\mathbf{0.664}_{\pm 0.01}$	$0.612_{\pm 0.02}$	$0.641_{\pm 0.04}$
Handwriting	$0.297_{\pm 0.01}$	$0.312_{\pm 0.01}$	$\mathbf{0.447}_{\pm 0.01}$	$0.390_{\pm 0.01}$	$0.365_{\pm 0.01}$
Heartbeat	$0.660_{\pm 0.01}$	$0.719_{\pm 0.01}$	$0.738_{\pm 0.01}$	$0.738_{\pm 0.02}$	$\mathbf{0.762}_{\pm 0.01}$
IMDB	$0.874_{\pm 0.00}$	$0.876_{\pm 0.00}$	$\mathbf{0.877}_{\pm 0.00}$	$0.873_{\pm 0.00}$	$0.876_{\pm 0.00}$
KeplerLightCurves	$0.321_{\pm 0.07}$	$0.452_{\pm 0.07}$	$\mathbf{0.489}_{\pm 0.04}$	$0.452_{\pm 0.02}$	$0.459_{\pm 0.05}$
Libras	$0.669_{\pm 0.05}$	$\mathbf{0.845}_{\pm 0.01}$	$0.835_{\pm 0.01}$	$0.765_{\pm 0.01}$	$0.781_{\pm 0.01}$
Lightning2	$0.607_{\pm 0.00}$	$0.623_{\pm 0.00}$	$0.720_{\pm 0.04}$	$0.605_{\pm 0.02}$	$\mathbf{0.772}_{\pm 0.03}$
Mallat	$0.649_{\pm 0.01}$	$0.842_{\pm 0.04}$	$0.883_{\pm 0.01}$	$0.905_{\pm 0.01}$	$\mathbf{0.913}_{\pm 0.01}$
MotionSenseHAR	$0.870_{\pm 0.02}$	$0.864_{\pm 0.01}$	$0.883_{\pm 0.03}$	$0.863_{\pm 0.02}$	$\mathbf{0.956}_{\pm 0.01}$
Reuters	$0.739_{\pm 0.00}$	$0.777_{\pm 0.00}$	$0.779_{\pm 0.00}$	$0.776_{\pm 0.00}$	$\mathbf{0.780}_{\pm 0.00}$
ShapesAll	$0.592_{\pm 0.02}$	$0.806_{\pm 0.01}$	$\mathbf{0.822}_{\pm 0.01}$	$0.804_{\pm 0.01}$	$0.803_{\pm 0.01}$
Trace	$0.648_{\pm 0.07}$	$0.980_{\pm 0.00}$	$0.991_{\pm 0.01}$	$0.986_{\pm 0.01}$	$\mathbf{0.999}_{\pm 0.00}$
UWaveGestureLibraryAll	$0.833_{\pm 0.01}$	$0.952_{\pm 0.00}$	$0.962_{\pm 0.00}$	$\mathbf{0.963}_{\pm 0.00}$	$0.957_{\pm 0.00}$
Wafer	$0.984_{\pm 0.00}$	$0.989_{\pm 0.00}$	$\mathbf{0.994}_{\pm 0.00}$	$0.992_{\pm 0.00}$	$0.988_{\pm 0.00}$
Yoga	$0.702_{\pm 0.03}$	$0.755_{\pm 0.02}$	$0.834_{\pm 0.01}$	$\mathbf{0.846}_{\pm 0.01}$	$0.783_{\pm 0.01}$

tings with step size variability, we explored values of $\Delta\varepsilon$ in $\{10^{-5}, 10^{-4}, \ldots, 1\}$, and generated values of $\boldsymbol{\varepsilon}$ from a uniform distribution in $[\varepsilon, \varepsilon + \Delta\varepsilon]$. A similar setting was used for exploring the case with diffusion variability. For comparison, we ran experiments with standard ESNs, exploring values of $\rho(\mathbf{W}_h)$ in $\{0.3, 0.6, 0.9, 1.2\}$, α in $\{10^{-5}, 10^{-4}, \ldots, 1\}$, ω_x and ω_b as for the EuSN models. In all the cases, the readout was trained by ridge regression (with regularization coefficient equal to 1).

For each model individually, the values of the hyper-parameters were fine-tuned by model selection, by means of a random search with 1000 iterations. After the model selection process, for each model the selected configuration was

Table 3. Average ranking across all the time-series classification benchmarks.

Model	Avg Rank
EuSN-ε, γ	2.208
EuSN-ε	2.208
EuSN-γ	2.583
EuSN	2.750
ESN	4.458

instantiated 10 times (generating random reservoir guesses). These 10 instances were trained on the entire training set and then evaluated on the test set. Our code was written in Keras[4], and was run on a system with 2×20 Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20 GHz.

Results. The achieved results are given in Table 2, which reports the test accuracy of each tested model, averaged over the 10 repetitions. The results in the table show the practical effectiveness of the architectural variants proposed in this paper, which overall achieve the best result in the vast majority of the cases examined. In particular, the variant comprising the maximum variability explored in the paper, i.e., EuSN-ε, γ is the one that is found to be superior in most cases. Taken individually, the variability on the step size (EuSN-ε) is slightly less effective than the full variability, while the variability on the diffusion term (EuSN-γ) is the one that individually results in less effectiveness. It is interesting to note that although in some cases the difference in performance between the best proposed variance and the baseline EuSN model is minimal, in many cases (including Blink, Computers, Cricket, HandMovementDirection, Handwriting, Heartbeat, Lightning2, Mallat, MotionSenseHAR, and Yoga) the improvement achieved is definitely relevant. Furthermore, the results show clear confirmation of the accuracy advantage of the EuSN approach over traditional ESNs. In the few cases where ESNs exceed the accuracy of standard EuSNs (Computers, Cricket, MotionSenseHAR), the proposed EuSN variants achieve even higher accuracy.

Our analysis is further supported by the ranking values given in Table 3, which indicate that on average on the considered datasets, EuSN-ε, γ and EuSN-ε models perform the best, followed by EuSN-γ and standard EuSN, while ESN has the worst performance.

6 Conclusions

In this paper we have empirically explored the effects of (architecturally) introducing dynamical variability in the behavior of Euler State Networks (EuSNs), a recently introduced Reservoir Computing (RC) methodology featured by non-dissipative dynamics. Diversity has been enforced by using reservoir neurons with

[4] Source code available at https://github.com/gallicch/VariabilityEuSN.

variable step size of integration (EuSN-ε), and with different diffusion coefficient (EuSN-γ). Both the approaches impact on the organization of the diversification of the dynamical behavior of the model, as pointed out by analyzing the eigenvalues of the resulting Jacobian. Moreover, results on several time-series classification benchmarks showed the efficacy of the proposed variants, and of their synergy, as the EuSN model with both the introduced variability factors (EuSN-ε, γ) resulted in the highest accuracy in a larger number of cases. Notwithstanding the clear advantage of basic EuSNs over conventional Echo State Networks in the explored tasks, from a practical point of view, the results suggest the convenience in exploring EuSNs in conjunction with at least the EuSN-ε, γ variant.

Future work will focus on theoretical analysis of the effects of the dynamic variability factors introduced in this paper, and their application in pervasive artificial intelligence contexts.

References

1. Apté, C., Damerau, F., Weiss, S.M.: Automated learning of decision rules for text categorization. ACM Trans. Inf. Syst. (TOIS) **12**(3), 233–251 (1994)
2. Bacciu, D., et al.: Teaching-trustworthy autonomous cyber-physical applications through human-centred intelligence. In: 2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS), pp. 1–6. IEEE (2021)
3. Bacciu, D., Barsocchi, P., Chessa, S., Gallicchio, C., Micheli, A.: An experimental characterization of reservoir computing in ambient assisted living applications. Neural Comput. Appl. **24**(6), 1451–1464 (2014)
4. Bagnall, A., Vickers, J.L.W., Keogh, E.: The UEA & UCR time series classification repository. www.timeseriesclassification.com
5. Chang, B., Chen, M., Haber, E., Chi, E.H.: AntisymmetricRNN: a dynamical system view on recurrent neural networks. arXiv preprint arXiv:1902.09689 (2019)
6. Dragone, M., et al.: A cognitive robotic ecology approach to self-configuring and evolving AAL systems. Eng. Appl. Artif. Intell. **45**, 269–280 (2015)
7. Gallicchio, C.: Reservoir computing by discretizing ODEs. In: Proceedings of ESANN (2021)
8. Gallicchio, C., Micheli, A.: Architectural and Markovian factors of echo state networks. Neural Netw. **24**(5), 440–456 (2011)
9. Gallicchio, C.: Euler state networks. arXiv preprint arXiv:2203.09382 (2022)
10. Gallicchio, C., Micheli, A., Pedrelli, L.: Fast spectral radius initialization for recurrent neural networks. In: Oneto, L., Navarin, N., Sperduti, A., Anguita, D. (eds.) INNSBDDL 2019. PINNS, vol. 1, pp. 380–390. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-16841-4_39
11. Grigoryeva, L., Ortega, J.P.: Echo state networks are universal. Neural Netw. **108**, 495–508 (2018)
12. Haber, E., Ruthotto, L.: Stable architectures for deep neural networks. Inverse Prob. **34**(1), 014004 (2017)
13. Hammer, B., Tiňo, P.: Recurrent neural networks with small weights implement definite memory machines. Neural Comput. **15**(8), 1897–1929 (2003)
14. Jaeger, H.: 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)

15. Jaeger, H., Haas, H.: Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)
16. Jaeger, H., Lukoševičius, M., Popovici, D., Siewert, U.: Optimization and applications of echo state networks with leaky-integrator neurons. Neural Netw. **20**(3), 335–352 (2007)
17. Lukoševičius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Comput. Sci. Rev. **3**(3), 127–149 (2009)
18. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 142–150. Association for Computational Linguistics, Portland, Oregon, USA, June 2011. http://www.aclweb.org/anthology/P11-1015
19. Nakajima, K., Fischer, I. (eds.): Reservoir Computing. NCS, Springer, Singapore (2021). https://doi.org/10.1007/978-981-13-1687-6
20. Van der Sande, G., Brunner, D., Soriano, M.C.: Advances in photonic reservoir computing. Nanophotonics **6**(3), 561–576 (2017)
21. Tanaka, G., et al.: Recent advances in physical reservoir computing: a review. Neural Netw. **115**, 100–123 (2019)
22. Verstraeten, D., Schrauwen, B., d'Haene, M., Stroobandt, D.: An experimental unification of reservoir computing methods. Neural Netw. **20**(3), 391–403 (2007)
23. Yildiz, I., Jaeger, H., Kiebel, S.: Re-visiting the echo state property. Neural Netw. **35**, 1–9 (2012)