SCIENTIFIC REPORTS
natureresearch

OPEN

Echo State Networks with Self-Normalizing Activations on the Hyper-Sphere

Pietro Verzelli[1], Cesare Alippi[1,2] & Lorenzo Livi [3,4]

Among the various architectures of Recurrent Neural Networks, Echo State Networks (ESNs) emerged due to their simplified and inexpensive training procedure. These networks are known to be sensitive to the setting of hyper-parameters, which critically affect their behavior. Results show that their performance is usually maximized in a narrow region of hyper-parameter space called edge of criticality. Finding such a region requires searching in hyper-parameter space in a sensible way: hyper-parameter configurations marginally outside such a region might yield networks exhibiting fully developed chaos, hence producing unreliable computations. The performance gain due to optimizing hyper-parameters can be studied by considering the memory–nonlinearity trade-off, i.e., the fact that increasing the nonlinear behavior of the network degrades its ability to remember past inputs, and vice-versa. In this paper, we propose a model of ESNs that eliminates critical dependence on hyper-parameters, resulting in networks that provably cannot enter a chaotic regime and, at the same time, denotes nonlinear behavior in phase space characterized by a large memory of past inputs, comparable to the one of linear networks. Our contribution is supported by experiments corroborating our theoretical findings, showing that the proposed model displays dynamics that are rich-enough to approximate many common nonlinear systems used for benchmarking.

Although the use of *Recurrent Neural Networks (RNNs)* in machine learning is boosting, also as effective building blocks for deep learning architectures, a comprehensive understanding of their working principles is still missing[1,2]. Of particular relevance are Echo State Networks (ESNs), introduced by Jaeger[3] and independently by Maass *et al.*[4] under the name of LSM, which emerge from RNNs due to their training simplicity. The basic idea behind ESNs is to create a randomly connected recurrent network, called reservoir, and feed it with a signal so that the network will encode the underlying dynamics in its internal states. The desired – task dependent – output is then generated by a readout layer (usually linear) trained to match the states with the desired outputs. Despite the simplified training protocol, ESNs are universal function approximators[5] and have shown to be effective in many relevant tasks[6–12].

These networks are known to be sensitive to the setting of hyper-parameters like the SR, the input scaling and the sparseness degree[3], which critically affect their behavior and, hence, the performance at task. Fine tuning of hyper-parameters requires cross-validation or ad-hoc criteria for selecting the best-performing configuration. Experimental evidence and some results from the theory show that ESNs performance is usually maximized in correspondence of a very narrow region in hyper-parameter space called Edge of Criticality (EoC)[13–20]. However, we comment that beyond such a region ESNs behave chaotically, resulting in useless and unreliable computations. At the same time, it is everything but trivial configuring the hyper-parameters to lie on the EoC still granting a non-chaotic behavior. A very important property for ESNs is the Echo State Property (ESP), which basically asserts that their behavior should depend on the signal driving the network only, regardless of its initial conditions[21]. Despite being at the foundation of theoretical results[5], the ESP in its original formulation raises some issues, mainly because it does not account for multi-stability and is not tightly linked with properties of the specific input signal driving the network[21–23].

[1]Faculty of Informatics, Università della Svizzera Italiana, Lugano, 69000, Switzerland. [2]Department of Electronics, Information and bioengineering, Politecnico di Milano, Milan, 20133, Italy. [3]Departments of Computer Science and Mathematics, University of Manitoba, Winnipeg, MB, R3T 2N2, Canada. [4]Department of Computer Science, College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, EX4 4QF, United Kingdom. Correspondence and requests for materials should be addressed to P.V. (email: pietro.verzelli@usi.ch)
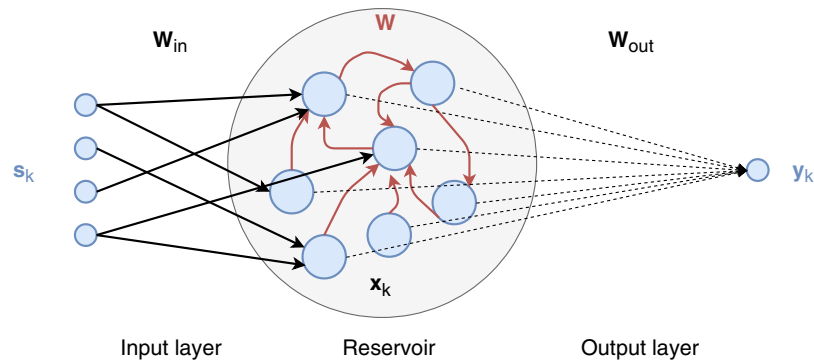
Figure 1. Schematic representation of an ESN with a 4-dimensional input, a mono-dimensional output and $N = 7$ neurons in the reservoir. Fixed connections are drawn using continuous lines while dashed lines represent connections that are learned.

In this context, the analysis of the memory capacity (as measured by the ability of the network to reconstruct or remember past inputs) of input-driven systems plays a fundamental role in the study of ESNs[24–27]. In particular, it is known that ESNs are characterized by a memory–nonlinearity trade-off[28–30], in the sense that introducing nonlinear dynamics in the network degrades memory capacity. Moreover, it has been recently shown that optimizing memory capacity does not necessarily lead to networks with higher prediction performance[31].

In this paper, we propose an ESN model that eliminates critical dependence on hyper-parameters, resulting in models that cannot enter a chaotic regime. In addition to this major outcome, such networks denote nonlinear behavior in phase space characterized by a large memory of past inputs: the proposed model generates dynamics that are rich-enough to approximate nonlinear systems typically used as benchmarks. Our contribution is based on a nonlinear activation function that normalizes neuron activations on a hyper-sphere. We show that the spectral radius of the reservoir, which is the most important hyper-parameter for controlling the ESN behavior, plays a marginal role in influencing the stability of the proposed model, although it has an impact on the capability of the network to memorize past inputs. Our theoretical analysis demonstrates that this property derives from the impossibility for the system to display a chaotic behavior: in fact, the maximum Lyapunov exponent is always null. An interpretation of this very important outcome is that the network always operates on the EoC, regardless of the setting chosen for its hyper-parameters. We tested the memory of our model on a series of benchmark time-series, showing that its performance for memorization tasks is comparable with that exposed by linear networks. We also explored the memory–nonlinearity trade–off following[30], showing that our model outperforms networks with linear and hyperbolical activations when both memory and nonlinearity are required by the task at hand.

Results

Echo state networks. An ESN without output feedback connections is defined as:

$$a_k = Wx_{k-1} + W_{\text{in}}s_k \tag{1a}$$

$$x_k = \phi(a_k) \tag{1b}$$

$$y_k = W_{\text{out}}x_k \tag{1c}$$

where $x_k \in \mathbb{R}^N$ is the system state at time-step k, N is the number of hidden neurons composing the network reservoir, and $W \in \mathbb{R}^{N \times N}$ is the connection matrix of the reservoir. The signal value at time k, $s_k \in \mathbb{R}^{N_{\text{in}}}$, is processed by the input-to-reservoir matrix $W_{\text{in}} \in \mathbb{R}^{N \times N_{\text{in}}}$. The activation function ϕ usually takes the form of the hyperbolic tangent function, for which the network is a universal function approximator[5]. Also linear networks (i.e., when ϕ is the identity) are commonly studied, both for the proven impact on applications and the very interesting results that can be derived in closed-form[24,25,27,31,32]. Other activation functions have been proposed in the neural networks literature, including those that normalize the activation values on a closed hyper-surface[33] and those based on non-parametric estimation via composition of kernel functions[34].

The output $y \in \mathbb{R}^{N_{\text{out}}}$ is generated by the matrix W_{out}, whose weights are learned: generally by ridge regression or lasso[3,35] but also with online training mechanisms[36]. In ESNs, the training phase does not affect the dynamics of the system, which are *de facto* task-independent. It follows that once a suitable reservoir rich in dynamics is generated, the same reservoir may serve to perform different tasks. A schematic representation of and ESN in depicted in Fig. 1.

Self-normalizing activation function. Here, we propose a new model for ESNs characterized by the use of a particular self-normalizing activation function that provides important features to the resulting network. Notably, the proposed activation function allows the network to exhibit nonlinear behaviors and, at the same time, provides memory properties similar to those observed for linear networks. This superior memory capacity is linked to the fact that the network never displays a chaotic behavior: we will show that the maximum Lyapunov
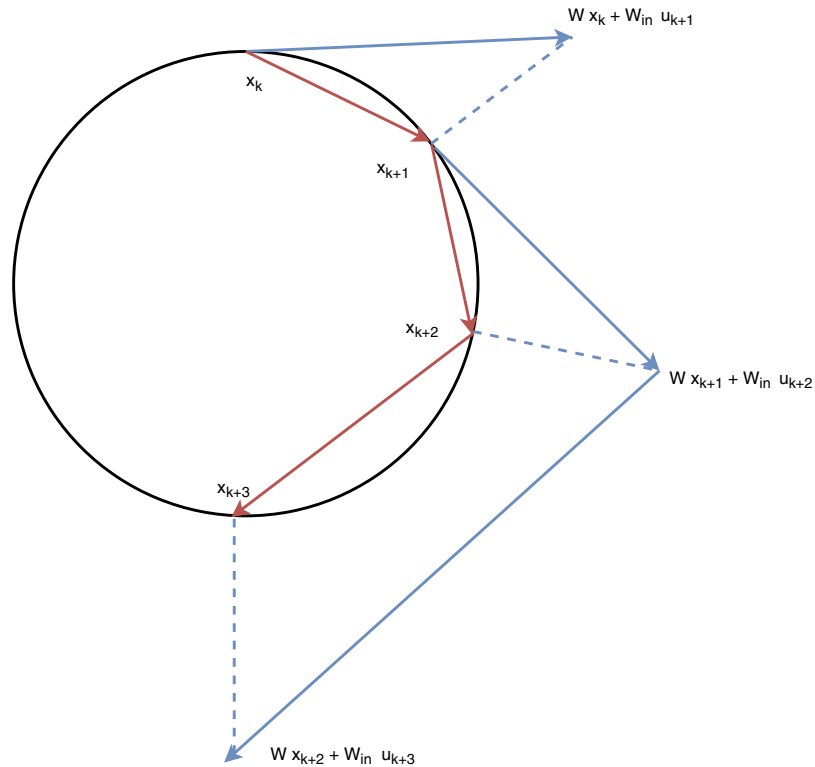
Figure 2. Example of the behavior of the proposed model in a 2-dimensional scenario. The blue lines represent the linear update step of Eq. (2a), while the dashed lines denote the projection of Eq. (2b). The red lines represent the actual steps performed by the system. Note that condition (3) accounts for the fact that the linear step must never bring the system state inside the hyper-sphere.

exponent is always zero, implying that the network operates on the EoC. The proposed activation function guarantees that the SR of the reservoir matrix (whose value is used as a control parameter) can vary in a wide range without affecting the network stability.

The proposed self-normalizing activation function is

$$\boldsymbol{a}_k = W\boldsymbol{x}_{k-1} + W_{\text{in}}\boldsymbol{s}_k \tag{2a}$$

$$\boldsymbol{x}_k = r\frac{\boldsymbol{a}_k}{\|\boldsymbol{a}_k\|} \tag{2b}$$

and leaves the readout (1c) unaltered. The normalization in Eq. 2b projects the network pre-activation a_k onto an $(N-1)$-dimensional hyper-sphere $\mathbb{S}_r^{N-1} := \{\boldsymbol{p} \in \mathbb{R}^N, \|\boldsymbol{p}\| = r\}$ of radius r. Figure 2 illustrates the normalization operator applied to the state. Note that the operation (2b) is not element-wise like most of activation functions as its effect is global, meaning that a neuron's activation value depends on all other values.

Universal function approximation property. The fact that recurrent neural networks are universal function approximators has been proven in previous works[37,38] and some results on the universality of reservoir-based computation are given in[4,39]. Recently, Grigoryeva and Ortega[5] proved that ESNs share the same property. Here, we show that the universal function approximation property also holds for the proposed ESNs model (2). To this end, we define a *squashing function* as a map $f: \mathbb{R} \to [-1, 1]$ that is non decreasing and saturating, i.e., $\lim_{x\to\pm\infty} f(x) = \pm 1$. We note that $\phi_i(\boldsymbol{x}) := x_i/\|\boldsymbol{x}\|$ can be intended as a squashing function for each ith component. In the same work[5], the authors show that an ESN in the form (1) is a universal function approximator provided it satisfies the *contractivity condition*: $\|\phi(x_1) - \phi(x_2)\| \leq \|x_1 - x_2\|$ for each x_1 and x_2, when ϕ is a squashing function. Jaeger and Haas[3] showed that this condition is sufficient to grant the ESP, implying that ESNs are universal approximators.

We prove in the Methods section that (2) grants the contractivity condition, provided that:

$$\sigma_{\min}(W) \geq 1 + \frac{\|W_{\text{in}}\| \|s_{\max}\|}{r} \tag{3}$$

where $\sigma_{\min}(W)$ is the smallest singular value of matrix W and $\|s_{\max}\|$ denotes the largest norm associated to an input. Equation 3 results in a sufficient yet not necessary condition that may be understood as requiring that the input will never be strong enough to contrast the expansive dynamics of the system, leading the network state

inside the hyper-sphere of radius r. In fact, unless the signal is explicitly designed for violating such a condition, it will very likely not bring the system inside the hyper-sphere as long as the norm of W is large enough compared to the signal variance.

Network state dynamics: the autonomous case. We now discuss the network state dynamics in the autonomous case, i.e., in the absence of input. This allows us to prove why the network cannot be chaotic.

From now on, we assume $r = 1$ as this does not affect the dynamics, provided that condition (3) is satisfied. From (2), the system state dynamics in the autonomous case reads:

$$\phi(\boldsymbol{x}) := \frac{W\boldsymbol{x}}{\|W\boldsymbol{x}\|} \tag{4}$$

At time-step n the system state is given by

$$\boldsymbol{x}_n = \phi(\boldsymbol{x}_{n-1}) = \frac{W\boldsymbol{x}_{n-1}}{\|W\boldsymbol{x}_{n-1}\|} = \frac{W}{\left\|WW\frac{\boldsymbol{x}_{n-2}}{\|W\boldsymbol{x}_{n-2}\|}\right\|} \frac{W\boldsymbol{x}_{n-2}}{\|W\boldsymbol{x}_{n-2}\|} = \frac{W^2\boldsymbol{x}_{n-2}}{\|W^2\boldsymbol{x}_{n-2}\|} \tag{5}$$

By iterating this procedure, one obtains:

$$\boldsymbol{x}_n := \phi^n(\boldsymbol{x}_0) = \frac{W^n\boldsymbol{x}_0}{\|W^n\boldsymbol{x}_0\|} \tag{6}$$

where $\boldsymbol{x}_0$ is the initial state. This implies that, for the autonomous case, a system evolving for n time-steps coincides with updating the state by performing n matrix multiplications and projecting the outcome only at the end.

It is worth to comment that this holds also if matrix W changes over time. In fact, let $W_n := W(n)$ be W at time time n. Then, the evolution of the dynamical system reads:

$$\boldsymbol{x}_n := \phi^n(x_0) = \frac{W_n W_{n-1} \dots W_2 W_1 \boldsymbol{x}_0}{\|W_n W_{n-1} \dots W_2 W_1 \boldsymbol{x}_0\|} \tag{7}$$

Furthermore, note that a system described by matrix W and a system characterized by $W' = aW$ coincide. In turn, this implies that the SR of the matrix does not alter the dynamics in the autonomous case.

Edge of criticality. When tuning the hyper-parameters of ESNs, one usually tries to bring the system close to the EoC, since it is in that region that their performance is optimal[14]. This can be explained by the fact that, when operating in that regime, the system introduces rich dynamics without denoting chaotic behavior.

Here, we show that the proposed recurrent model (2) cannot enter a chaotic regime. Notably, we prove that, when the number of neurons in the network is large, the maximum (local) Lyapunov exponent cannot be positive, hence neglecting the possibility to introduce a chaotic behavior. To this end, we determine the Jacobian matrix of (2b) and then show that, since its spectral radius tends to 1, the maximum LLE must be null. The Jacobian matrix of (2b) reads:

$$J_{ij} = \frac{\partial}{\partial x_i} \phi_j(\boldsymbol{x}) = \sum_l^N \frac{\partial \phi_j}{\partial a_l} \frac{\partial a_l}{\partial x_i} = \frac{W_{ij}}{\|\boldsymbol{a}\|}\left(1 - \frac{a_i a_j}{\|\boldsymbol{a}\|^2}\right) \tag{8}$$

where the time index k is omitted to ease the notation. We observe that, asymptotically for large networks ($n \to \infty$), we have that $a_i/\|\boldsymbol{a}\| \to 0$ for each i, meaning that the Jacobian matrix reduces to $J(\boldsymbol{x}) = W/\|W\boldsymbol{x}\|$. As we are considering the case with $r = 1$, we know that $\|\boldsymbol{x}\| = 1$.

This allows us to approximate the norm of W with its SR $\rho = \rho(W)$,

$$J \approx \frac{W}{\|W\boldsymbol{x}\|} \approx \frac{W}{\rho}. \tag{9}$$

Under this approximation (9), the largest eigenvalue of J must be 1 as the SR ρ is the largest absolute value among the eigenvalues of W. We thus characterize the global behavior of (2b) by considering the maximum LLE[14], which is defined as:

$$\Lambda := \lim_{n \to \infty} \frac{1}{n} \log\left(\prod_k^n \rho_k\right) \tag{10}$$

where ρ_k is the spectral radius of the Jacobian at time-step k. Equation 10 implies that $\Lambda = 0$ as $n \to \infty$, hence proving our claim.

In order to demonstrate that $\Lambda = 0$ holds also for networks with a finite number N of neurons in the recurrent layer, we numerically compute the maximum LLE by considering the Jacobian in (8). The results are displayed in Fig. 3. Figure 3 panel (a) shows the average value of the maximum LLE with the related standard deviation obtained for different values of SR. Results show that the LLE is not significantly different from zero. In Fig. 3 panel (b), instead, we show the Lyapunov spectrum of a network with $N = 100$ neurons in the recurrent layer,
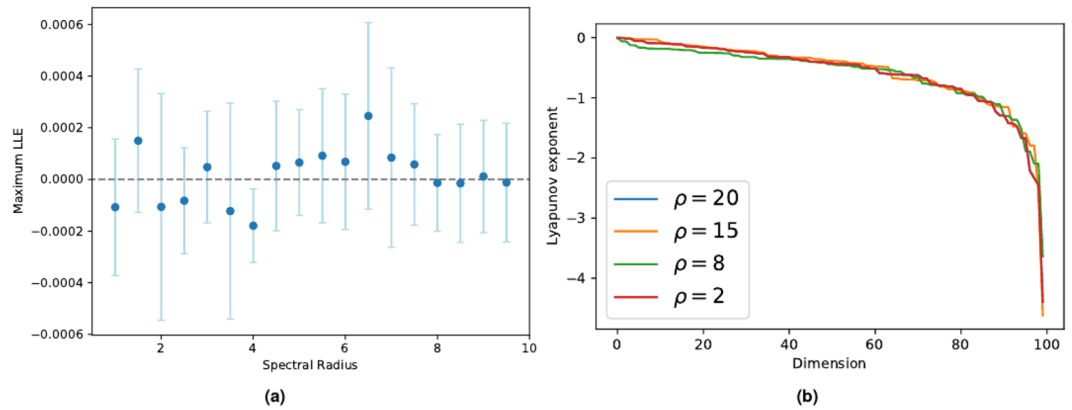
Figure 3. Panel (a) shows LLEs for different value of the SR. Each point in the plot represents the mean of 10 different realizations, using a network with $N = 500$ neurons. In panel (b) examples of the Lyapunov spectrum for different 100-dimensional networks are plotted. The Lyapunov exponents are ordered by decreasing magnitude. Note that the largest exponent is always zero.

obtained for different SR values. Again, our results show that the maximum LLE of (2b) is zero for finite-size networks, regardless of the values chosen for the SR.

Network state dynamics: input-driven case. Let us define the "effective input" contributing to the neuron activation as $\boldsymbol{u} := W_{\text{in}}\boldsymbol{s}$. Accordingly, Eq. 2a takes on the following form:

$$\boldsymbol{a}_k = W\boldsymbol{x}_{k-1} + \boldsymbol{u}_k, \tag{11}$$

where $\boldsymbol{u}_k$ operates as a time-dependent bias vector. Let us define the normalization factor as:

$$N_k = \| W\boldsymbol{x}_{k-1} + \boldsymbol{u}_k \|, \tag{12}$$

Then, as shown in the Methods section, the state at time-step n can be written as:

$$\boldsymbol{x}_n = M^{(n,0)}\boldsymbol{x}_0 + \sum_{k=1}^{n} M^{(n,k)}\boldsymbol{u}_k \tag{13}$$

where

$$M^{(n,k)} := \frac{W^{n-k}}{\prod_{l=k}^{n} N_l}. \tag{14}$$

By looking at (13), it is possible to note that each $\boldsymbol{u}_k$ is multiplied by a time-dependent matrix, i.e., the network's final state $\boldsymbol{x}_n$ is obtained as a linear combination of all previous inputs.

Memory of past inputs. Here, we analyze the ability of the proposed model (2) to retain memory of past inputs in the state. In order to perform such analysis, we define a measure of network's memory that quantifies the impact of past inputs on current state $\boldsymbol{x}_n$. Our proposal shares similarities with a memory measure called Fisher memory, first proposed by Ganguli *et al.*[27] and then further developed in[24,32]. However, our measure can be easily applied also to non-linear networks like the one Eq. 2 proposed here, justifying the analysis discussed in the following.

Considering one time-step in the past, we have:

$$\boldsymbol{x}_n = \frac{W}{N_n}\boldsymbol{x}_{n-1} + \frac{1}{N_n}\boldsymbol{u}_n \tag{15}$$

All past history of the signal is processed in $\boldsymbol{x}_{n-1}$. Note that (15) keeps the same form for every n. Going backward in time one more step, we obtain:

$$\boldsymbol{x}_n = \frac{W}{N_n}\left(\frac{W}{N_{n-1}}\boldsymbol{x}_{n-2} + \frac{1}{N_{n-1}}\boldsymbol{u}_{n-1} \right) + \frac{1}{N_n}\boldsymbol{u}_n \tag{16}$$

As $\|\boldsymbol{x}_n\| = \|\boldsymbol{x}_{n-1}\| = \|\boldsymbol{x}_{n-2}\| = 1$, we see that there is a sort of recursive structure in this procedure, where each incoming input is summed to the current state and then their sum is normalized. This is a key feature of the proposed architecture. In fact, it guarantees that each input will have an impact on the state, since the norm of the activations will not be too big or too small, because of the normalization. We now express this idea in a more formal way.

The norm of (14) can be written as

$$\|M^{(n,k)}\| = \frac{1}{\prod_{l=k}^{n} N_l} \|W^{n-k}\| \overset{n-k \gg 1}{\approx} \frac{1}{\prod_{l=k}^{n} N_l} \rho^{n-k} \tag{17}$$

where the approximation holds thanks to the Gelfand's formula[a]. If the input is null, we expect each N_l to be of the order of ρ as $\|x_l\| = 1$. So we write:

$$N_l = \|Wx_{l-1} + u_l\| = \rho + \delta_l \tag{18}$$

where δ_l denotes the impact of the l-th input on the l-th normalization factor. Its presence is due to the fact that without any input N_l would be approximately equal to ρ, while the input modifies the state and so the normalization value will be modified accordingly. The value δ_l is introduced to account for this fact: $\delta_l = N_l - \rho$.

If we assume that each input produces a similar effect on the state (i.e. $\delta_l = \delta$ for every l), we finally obtain:

$$\|M^{(n,k)}\| \approx \frac{\rho^{n-k}}{(\rho + \delta)^{n-k}} \tag{19}$$

We note that such assumption is reasonable for inputs that are symmetrically distributed around a mean value with relatively small variance, e.g. for Gaussian or uniformly distributed inputs (as demonstrated by our results). However, our argument might not hold for all types of inputs and a more detailed analysis is left as future research.

We define the memory of the network $\mathcal{M}_\alpha(k|n)$ as the influence of input at time k on the network state at time n. More formally, having defined $\alpha := \rho/\delta$ (since (19) only depends on this ratio), we use (19) to define the memory as:

$$\mathcal{M}_\alpha(k|n) := \left(\frac{\alpha}{\alpha+1}\right)^{n-k} = \left(1 - \frac{1}{\alpha+1}\right)^{n-k} \tag{20}$$

Equation 20 goes to 0 (i.e., no impact of the input on the states) as $\alpha \to 0$ and tends to 1 for $\alpha \to \infty$, implying that for an infinitely large SR the network perfectly remembers its past inputs, regardless of their occurrence in the past. Note that (20) does not depend on k and n individually, but only on their difference (elapsed time): the larger the difference, the lower the value of $\mathcal{M}_\alpha(k|n)$, meaning that far away inputs have less impact on the current state.

Memory loss. By using (20), we define the memory loss between state x_n and x_m of the signal at time-step k (with $m > n > k$ and $m = n + \tau$) as follows:

$$\begin{aligned}
\Delta\mathcal{M}(k|m, n) &\coloneqq \mathcal{M}_\alpha(k|m) - \mathcal{M}_\alpha(k|n) \\
&= \left(\frac{\alpha}{\alpha+1}\right)^{m-k} - \left(\frac{\alpha}{\alpha+1}\right)^{n-k} \\
&= \frac{(\alpha+1)^k}{\alpha^k} \frac{\alpha^n}{(\alpha+1)^n} \left(\frac{\alpha^\tau}{(\alpha+1)^\tau} - 1\right) \\
&= \left(1 - \frac{1}{\alpha+1}\right)^n \left(1 + \frac{1}{\alpha}\right)^k \left(\left(1 - \frac{1}{\alpha+1}\right)^\tau - 1\right) \le 0
\end{aligned} \tag{21}$$

For very large values of α, we have $\Delta\mathcal{M}(k|m, n) \to 0$, meaning that in our model (2b) larger spectral radii eliminate memory losses of past inputs. Now, we want to assess if inputs in the far past have higher/lower impact on the state more than recent inputs. To this end, we selected $n > k > l$ and defined $k = n - a$ and $l = n - b$, leading to $b > a > 0$ and $b = a + \delta$. Define:

$$\begin{aligned}
\Delta\mathcal{M}(k, l|n) &= \mathcal{M}(n - a|n) - \mathcal{M}(n - a - \delta|n) \\
&= \frac{\alpha^a}{(\alpha+1)^a} - \frac{\alpha^{a+\delta}}{(\alpha+1)^{a+\delta}} \\
&= \frac{\alpha^a}{(\alpha+1)^a} \left(1 - \frac{\alpha^\delta}{(\alpha+1)^\delta}\right) \\
&= \left(1 - \frac{1}{\alpha+1}\right)^a \left(1 - \left(1 - \frac{1}{\alpha+1}\right)^\delta\right) \ge 0
\end{aligned} \tag{22}$$

We have that $\lim_{\delta \to \infty} \Delta\mathcal{M}(n - a, n - a - \delta|n) = \left(1 - \frac{1}{\alpha+1}\right)^a$, showing how the impact of an input that is infinitely far in the past is not null compared with one that is only $a < \infty$ steps behind the current state. This implies that the proposed network is able to preserve in the network state (partial) information from all inputs observed so far.

---

[a]The Gelfand's formula[40] states that for any matrix norm $\rho(A) = \lim_{k \to \infty} \|A^k\|^{\frac{1}{k}}$.

Linear networks.   In order to assess the memory of linear models, we perform the same analysis for linear RNNs (i.e., $\boldsymbol{x}_n = \boldsymbol{a}_n$) by using the definitions given in the previous section. In this case, an expression for the memory can be obtained straightforwardly and reads:

$$\mathcal{M}_L(k|n) := \rho^{n-k} \tag{23}$$

It is worth noting that there is no dependency on δ and, therefore, on the input. Just like before, we have the memory loss of the signal at time-step k between state $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$, as:

$$\begin{aligned}
\Delta\mathcal{M}_L(k|m, n) &= \mathcal{M}_L(k|m) - \mathcal{M}_L(k|n) \\
&= \mathcal{M}_L(k|n + \tau) - \mathcal{M}_L(k|n) \\
&= (\rho^\tau - 1)\rho^{n-k} \leq 0
\end{aligned} \tag{24}$$

where we set $m > n > k$ and $m = n + \tau$. As before, we also discuss the case of two different inputs observed before time-step n. By selecting time instances $n > k > l$, $k = n - a$ and $l = n - b$, we have $b > a > 0$ allowing us to write $b = a + \delta$. As such:

$$\Delta\mathcal{M}_L(k, l|n) = \mathcal{M}_L(k|n) - \mathcal{M}_L(l|n) = (1 - \rho^\delta)\rho^a \geq 0. \tag{25}$$

We see that in both (24) and (25) the memory loss tends to zero as the spectral radius tends to one, which is the critical point for linear systems. So, according to our analysis, linear networks should maximize their ability to remember past inputs when their SR in close to one and, moreover, their memory should be the same disregarding the particular signal they are dealing with. We will see in the next section that both these claims are confirmed by our simulations.

Hyperbolic tangent.   We now extend the analysis to standard ESNs, i.e., those using a tanh activation function. Define $\phi :=$ tanh (applied element-wise). Then, for generic scalars a and b, when $|b| \ll 1$ the following approximation holds:

$$\phi(\boldsymbol{a} + \boldsymbol{b}) = \frac{\phi(\boldsymbol{a}) + \phi(\boldsymbol{b})}{1 + \phi(\boldsymbol{a})\phi(\boldsymbol{b})} \approx \phi(\boldsymbol{a}) + \phi(\boldsymbol{b}) \tag{26}$$

When $\boldsymbol{a}$ is the state and $\boldsymbol{b}$ is the input, our approximation can be understood as a small-input approximation[41]. Then, the state-update reads:

$$\begin{aligned}
\boldsymbol{x}_n &= \phi(W\boldsymbol{x}_{n-1} + \boldsymbol{u}_n) \\
&\approx \phi(W\boldsymbol{x}_{n-1}) + \phi(\boldsymbol{u}_n) \\
&= \phi(W\phi(W\boldsymbol{x}_{n-2} + \boldsymbol{u}_{n-1})) + \phi(\boldsymbol{u}_n) \\
&\approx \phi(W\phi(W\boldsymbol{x}_{n-2})) + \phi(W\phi(\boldsymbol{u}_{n-1})) + \phi(\boldsymbol{u}_n)
\end{aligned} \tag{27}$$

Thus, applying the same argument used in the previous cases, it is possible to write:

$$\begin{aligned}
\|\boldsymbol{x}_n\| &= S_\rho^{(0)}(\phi(\|\boldsymbol{u}_n\|)) + S_\rho^{(1)}(\phi(\|\boldsymbol{u}_{n-1}\|)) + \cdots + S_\rho^{(n)}(\phi(\|\boldsymbol{u}_0\|)) \\
&= \sum_{k=0}^{n} S_\rho^{(n-k)}(\phi(\|\boldsymbol{u}_k\|))
\end{aligned} \tag{28}$$

where we defined:

$$S_\rho^{(n)}(\xi) := \underbrace{\phi(\rho \cdot \ldots \phi(\rho \cdot \phi(\xi)))}_{n \text{ times}} \tag{29}$$

We see that, differently from the previous cases, the final state $\boldsymbol{x}_n$ is a sum of nonlinear functions of the signal $\boldsymbol{u}_k$. Each signal element $\boldsymbol{u}_k$ is encoded in the network state by first multiplying it by ρ and then passing the outcome through the nonlinearity $\phi(\cdot)$. This implies that, for networks equipped with hyperbolic tangent function, larger spectral radii favour the forgetting of inputs (as we are in the non-linear region of $\phi(\cdot)$). On the other hand, for small spectral radii the network operates in the linear regime of the hyperbolic tangent and the network behaves like in the linear case.

A plot depicting the decay of the memory of input at time-step k for all three cases described above is shown in Fig. 4. The trends demonstrate that, in all cases, the decay is consistent with our theoretical predictions.

Performance on memory tasks.   We now perform experiments to assess the ability of the proposed model to reproduce inputs seen in the past, meaning that the desired output at time-step k is $\boldsymbol{y}_k = \boldsymbol{u}_{k-\tau}$, where τ ranges from 0 to 100. We compare our model with a linear network and a network with hyperbolic tangent (tanh) activation functions. We use fixed, but reasonable hyper-parameters for all networks, since in this case we are only interested in analyzing the network behavior on different tasks. In particular, we selected hyper-parameters that worked well in all cases taken into account; in preliminary experiments, we noted that different values did not result in substantial (qualitative) changes of the results. The number of neuron N is fixed to 1000 for all models. For linear and nonlinear networks, the input scaling (a constant scaling factor of the input signal) is fixed to 1 and
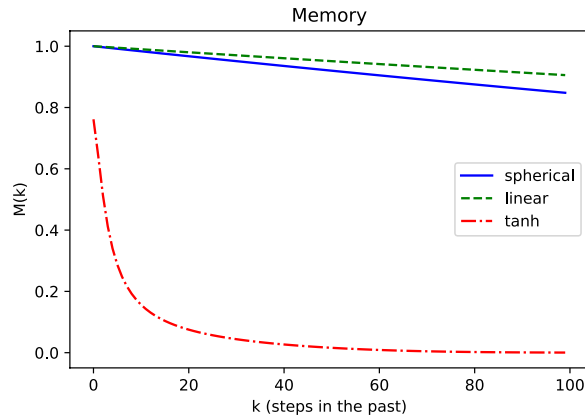
Figure 4. Decay of the memory in the cases associated with expressions (20), (23) and (28). In all cases, the decay is consistent with the predictions. The memory for the hyperbolic tangent is estimated assuming a signal of magnitude 1.

the SR equals $\rho = 0.95$. For the proposed model (2), the input scaling is chosen to be 0.01, while the SR is $\rho = 15$. For the sake of simplicity, in what follows we refer to ESNs resulting from the use of (2) as "spherical reservoir".

To evaluate the performance of the network, we use the accuracy metric defined as $\gamma = \max\{1 - \text{NRMSE}, 0\}$, where the NRMSE is defined as:

$$\text{NRMSE} := \sqrt{\frac{\langle \|\mathbf{y}_k - \hat{\mathbf{y}}_k\|^2 \rangle}{\langle \|\mathbf{y}_k - \langle \mathbf{y}_k \rangle\|^2 \rangle}} \,. \tag{30}$$

Here, $\hat{\mathbf{y}}_k$ denotes the computed output and $\langle \cdot \rangle$ represents the time average over all time-steps taken into account. In the following, we report the accuracy γ while varying τ in the past for various benchmark tasks. Each result in the plot is computed using the average over 20 runs with different network initializations. The networks are trained on a data set of length $L_{\text{train}} = 5000$ and the associated performance is evaluated using a test set of length $L_{\text{test}} = 2000$. The shaded area represents the standard deviation. All the considered signals were normalized to have unit variance.

White noise. In this task, the network is fed with white noise uniformly distributed in the $[-1, 1]$ interval. Results are shown in Fig. 5, panel (a). We note that networks using the spherical reservoir have a performance comparable with linear networks, while tanh networks do not correctly reconstruct the signal when k exceeds 20. It is worth highlighting the similarity of the plots shown here with our theoretical predictions about the memory (Eqs (20), (23) and (28)) shown in Fig. 4.

Multiple superimposed oscillators. The network is fed with Multiple Superimposed Oscillators MSO with 3 incommensurable frequencies:

$$u(k) = \sin(0.2k) + \sin(0.311k) + \sin(0.42k) \tag{31}$$

Results are shown in Fig. 5, panel (b). This task is difficult because of the multiple time scales characterizing the inputs[3]. We note the performance of the linear and the spherical reservoirs are again similar and both outperform the network using the hyperbolic tangent. The accuracy peak when $k \approx 60$ is due to the fact that the autocorrelation of the signal reaches its maximum value at that time-step.

Lorenz series. The Lorenz system is a popular mathematical model, originally developed to describe atmospheric convection.

$$\begin{cases} \dot{x} = \sigma(y - x), \\ \dot{y} = x(\rho - z) - y, \\ \dot{z} = xy - \beta z. \end{cases} \tag{32}$$

It is well-known that this system is chaotic when $\sigma = 10$, $\beta = \frac{8}{3}$ and $\rho = 28$. We simulated the system with these parameters and then fed the network with the x coordinate only. Results are shown in Fig. 5, panel (c). Also in this case, while the accuracy for spherical and linear networks does not seem to be affected by k, the performance of networks using the tanh activation dramatically decreases when k is large. This stresses the fact that non-linear networks are significantly penalized when they are requested to memorize past inputs.

Mackley-Glass system. The Mackley-Glass system is given by the following delayed differential equation:
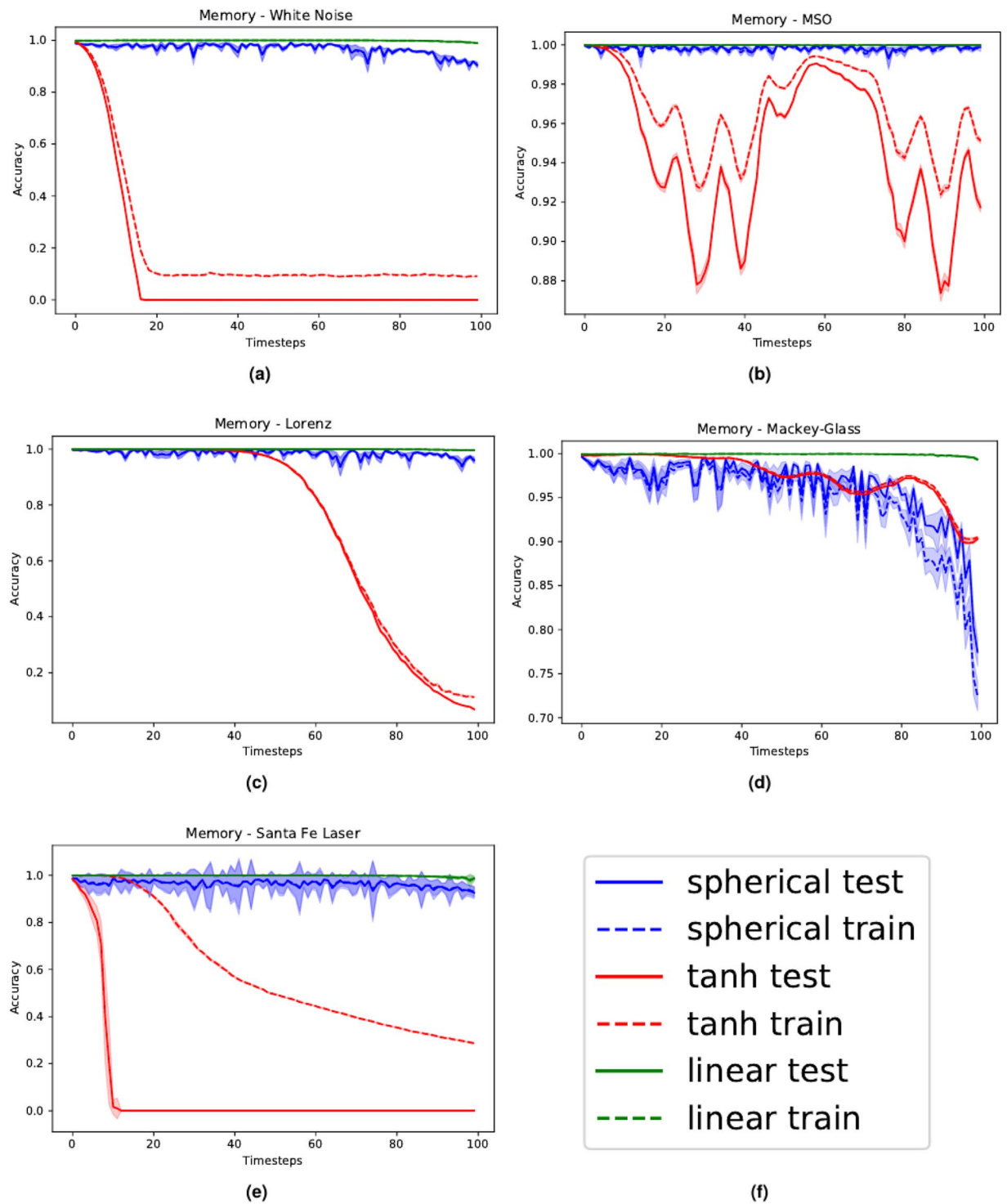
Figure 5. Results of the experiments on memory for different benchmarks. Panel (a) displays the white noise memorization task, (**b**) the MSO, (**c**) the x-coordinate of the Lorenz system, (**d**) the Mackey-Glass series and (**e**) the Santa Fe laser dataset. As described in the legend (**f**), different line types account for results obtained on training and test data. The shaded areas represent the standard deviations, computed using 20 different realization for each point.

$$\dot{x}(t) = -\beta x(t) + \frac{\alpha x(t - \lambda)}{1 + x(t - \lambda)} \qquad (33)$$

In our experiments, we simulated the system using standard parameters, that is, $\lambda = 17, \alpha = 0.2$ and $\beta = 0.1$. Results are shown in Fig. 5, panel (d). Note that in this case the performance of the network with spherical
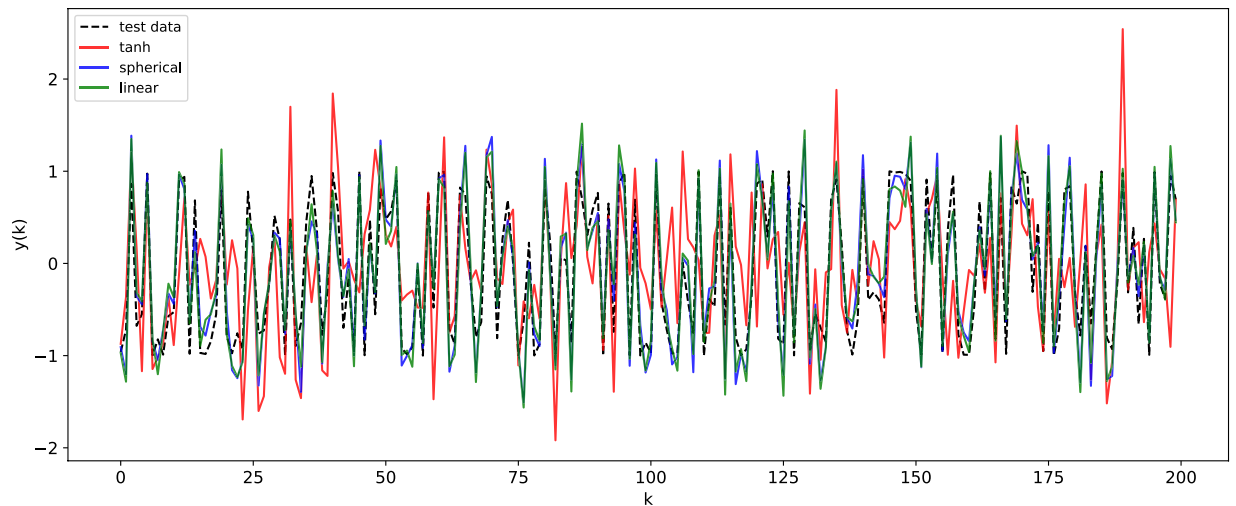
Figure 6. Comparison of the network prediction the memory–nonlinearity task for $\nu = 2.5$ and $\tau = 10$. The hyper-parameters of the networks are the same used to generate Fig. 5. Here the accuracy values are $\gamma_{\text{tanh}} = 0.12, \gamma_{\text{spherical}} = 0.63$ and $\gamma_{\text{linear}} = 0.61$.
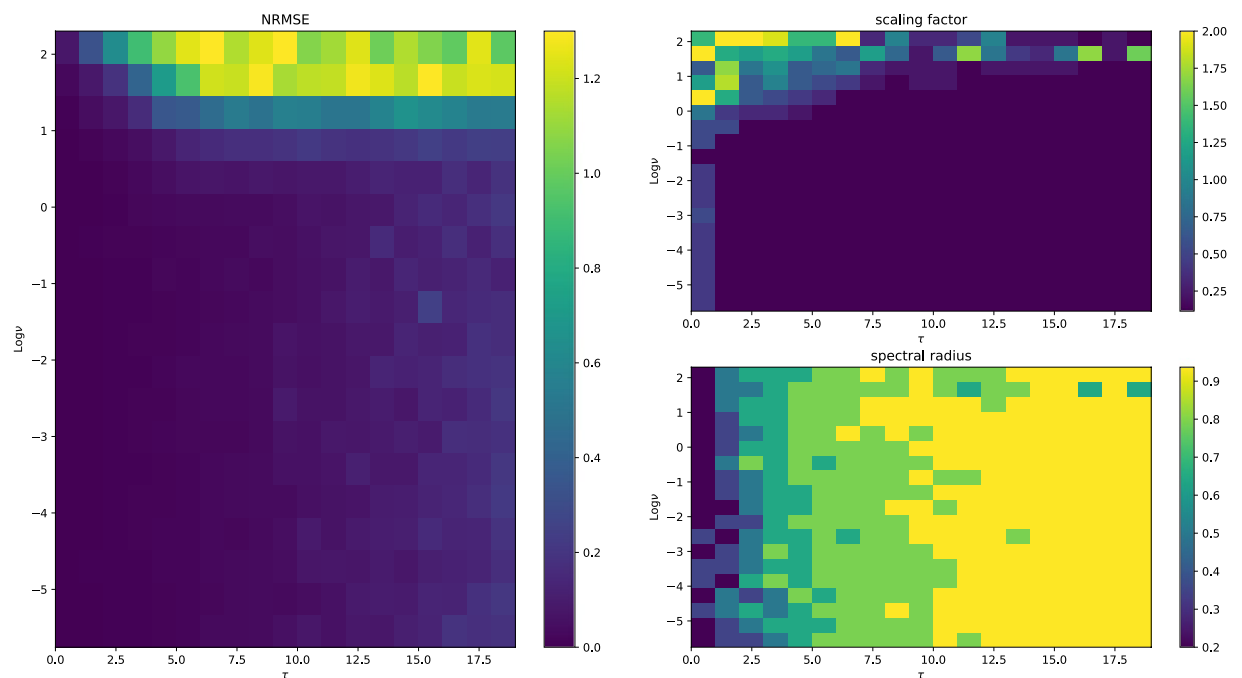

Figure 7. Results for the hyperbolic tangent activation function. The network performs as expected: the error grows with the memory required to solve the task. The choice of the spectral radius displays a pattern, where larger SRs are preferred when more memory is required. The scaling factor tends to be small for almost every configuration.

reservoir is comparable with the one obtained using the hyperbolic tangent and both of them are outperformed by the linear networks.

Santa Fe laser dataset. The Santa Fe laser dataset is a chaotic time series obtained from laser experiments. The results are shown in Fig. 5, panel (e). Also in this case the hyperbolic tangent networks do not manage to remember the signal, while the other systems show the usual behavior.

Memory/Non-Linearity Trade-Off. Here, we evaluate the capability of the network to deal with tasks characterized by tunable memory and non-linearity features[30]. The network is fed with a signal where the u_ks are univariate random variables drawn from a uniform distribution in $[-1, 1]$. The network is then trained to recon-
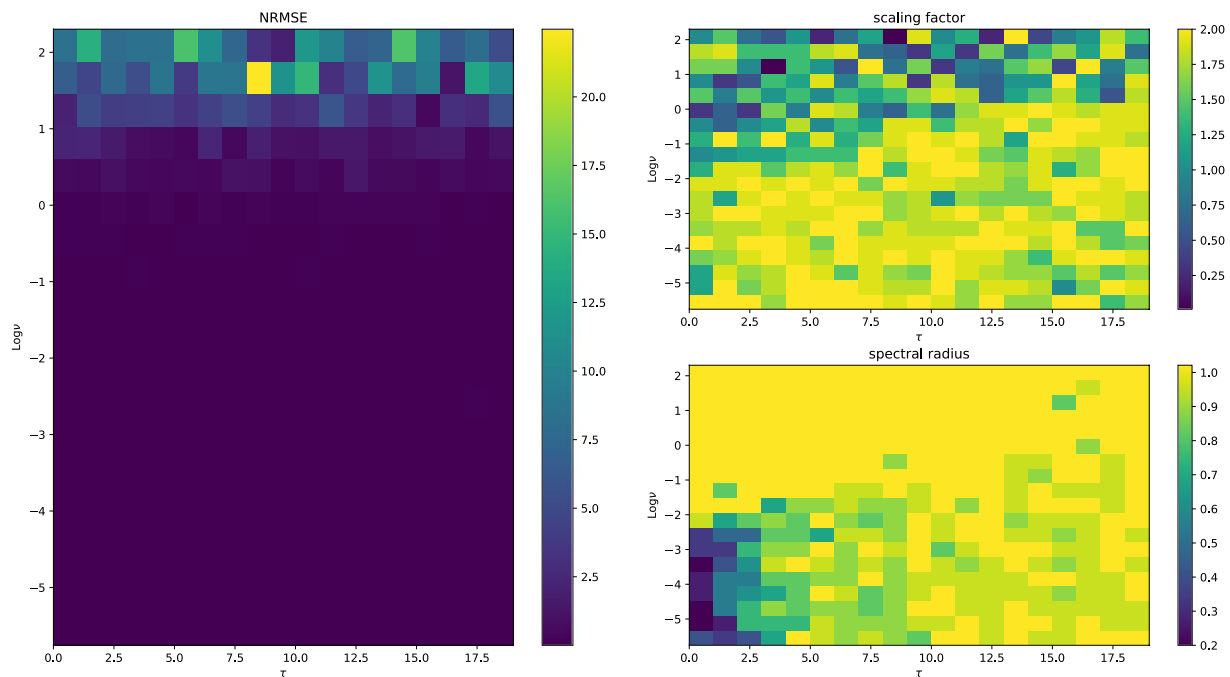
Figure 8. Performance of linear networks. We note that τ seems to have no significant effect on the performance. In fact, we note very large errors when the nonlinearity of the task is high. The choice of the scaling factor and of the spectral radius shows a really weak tendency to certain values, indicating that the performance is only weakly influenced by the hyper-parameters.
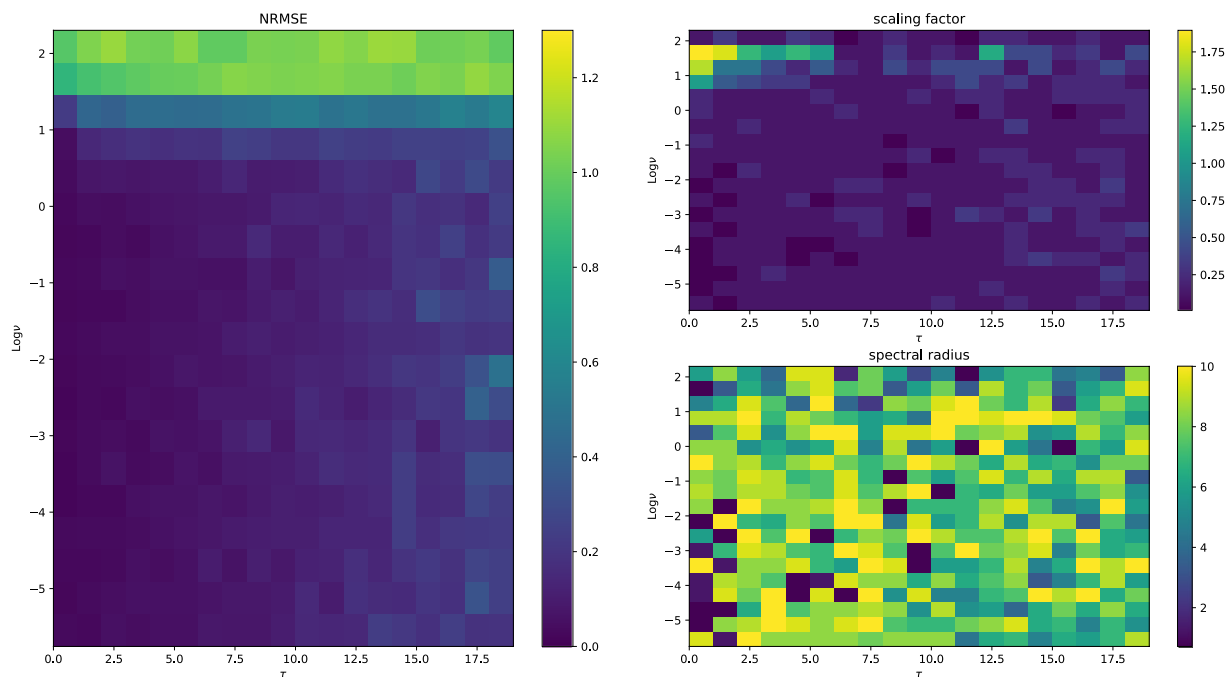

Figure 9. Performance of the proposed model. We see that the network performs reasonably well for all the tasks displaying only a weak dependency on the memory. Moreover the spectral radius does not to play any role in the network performance. The choice of the scaling factor denotes similar patterns with the hyperbolic tangent case.

struct the desired output of $y_k = \sin(\nu \cdot u_{k-\tau})$. We see that τ accounts for the memory required to correctly solve the task, while ν controls the amount non-linearity involved in the computation. For each configuration of τ and ν chosen in suitable ranges, we run a grid search on the range of admissible values of SR and input scaling.

Notably, we considered 20 equally-spaced values of the SR and for the input scaling. For networks using hyperbolic tangent, the SR varies in [0.2, 3]; [0.2, 1.5] for linear networks, and [0.2, 10] for networks with spherical reservoir. The input scaling always ranges in [0.01, 2][b]. We then choose the hyper-parameters configuration that minimizes (30) on a training set of length $L_{\text{train}} = 500$ and then assess the error on a test set with $L_{\text{test}} = 200$.

In Figs 7, 8 and 9, we show the NRMSE for the task described above for different values of ν and τ, and the ranges of the input scaling factor and the spectral radius which performed best for the hyperbolic tangent, linear and spherical activation function, respectively. The results of our simulations agree with those reported in[30] and, most importantly, confirm our theoretical prediction: the proposed model possess memory of past inputs that is comparable with the one of linear networks but, at the same time, it is also able to perform nonlinear computations. This explains why the proposed model denotes the best performance when the task requires both memory and nonlinearity, i.e., when both τ and ν are large. Predictions obtained for a specific instance of this task requiring both features are given in Fig. 6, showing how the proposed model outperforms the competitors. In order to explore the memory-nonlinearity relationship we followed the experimental design proposed in[30]: our goal is to study the memory property of the proposed model and not to develop a model specifically designed to maximize the compromise between memory and nonlinearity. The reader interested in models that aim at explicitly tackling the memory–nonlinearity problem may refer to recently-developed hybrid systems[30,42] which try to deal with the memory–nonlinearity trade–off by combining, in different ways, linear and non-linear units so that the network can exploit a combination of them according to the problem at hand. These approaches introduce new hyper-parameters which, basically, allow to control the memory–nonlinearity trade–off. Future works will investigate such a trade–off.

Discussion

In this work, we studied the properties of ESNs equipped with an activation function that projects at each time-step the network state on a hyper-sphere. The proposed activation function is global, contrarily to most activation functions used in the literature, meaning that the activation value of a neuron depends on the activations of all other neurons in the recurrent layer. Here, we first proved that the proposed model is a universal approximator just like regular ESNs and gave sufficient conditions to support this claim. Our theoretical analysis shows that the behavior of the resulting network is never chaotic, regardless of the setting of the main hyper-parameters affecting its behavior in phase space. This claim was supported both analytically and experimentally by showing that the maximun Lyapunov exponent is always zero, implying that the proposed model always operates on the EoCs. This leads to networks which are (globally) stable for any hyper-parameter configurations, regardless of the particular input signal driving the network, hence solving stability problems that might affect ESNs.

The proposed activation function allows the model to display a memory capacity comparable with the one of linear ESNs, but its intrinsic nonlinearity makes it capable to deal with tasks for which rich dynamics are required. To support this claim, we developed a novel theoretical framework to analyze the memory of the network. By taking inspiration from the Fisher memory proposed by Ganguli et al.[27], we focused on quantifying how much past inputs are encoded in future network states. The developed theoretical framework allowed us to account for the nonlinearity of the proposed model and predicted a memory capacity comparable with the one of linear ESNs. This theoretical prediction was validated experimentally by studying the behavior of the memory loss using white noise as inputs as well as well-known benchmarks used in the literature. Finally, we experimentally verified that the proposed model offers an optimal balance of nonlinearity and memory, showing that it outperforms both linear and ESNs with tanh activation functions in those tasks where both features are required at the same time.

Methods

Contractivity condition. The universal approximation property, as exhaustively discussed in[5], can be proved for an ESN of the form (1b) provided that it has the ESP. To prove the ESP, it is sufficient to show that the network has the contractivity condition, i.e., that for each $\boldsymbol{x}$ and $\boldsymbol{y}$ in the domain of the activation function ϕ, it must be true that:

$$\|\phi(\boldsymbol{x}) - \phi(\boldsymbol{y})\| \leq \|\boldsymbol{x} - \boldsymbol{y}\| \tag{34}$$

Let us introduce the the notation $\hat{\boldsymbol{x}} := \boldsymbol{x}/\|\boldsymbol{x}\|$ and $\hat{\boldsymbol{y}} := \boldsymbol{y}/\|\boldsymbol{y}\|$. Taking the square of the norm, one gets:

$$\|\phi(\boldsymbol{x}) - \phi(\boldsymbol{y})\|^2 = r^2(2 - 2(\hat{\boldsymbol{x}} \cdot \hat{\boldsymbol{y}})) \leq r^2\left(\left(\frac{\|\boldsymbol{x}\|}{\|\boldsymbol{y}\|} + \frac{\|\boldsymbol{y}\|}{\|\boldsymbol{x}\|}\right) - 2\hat{\boldsymbol{x}} \cdot \hat{\boldsymbol{y}}\right) \tag{35}$$

where $\boldsymbol{a} \cdot \boldsymbol{b}$ is the scalar product between two vectors and the inequality $\frac{a}{b} + \frac{b}{a} \geq 2$ follows from the fact that $(a - b)^2 = a^2 + b^2 - 2ab > 0$ for all $a, b > 0$. Now, we assume $\|\boldsymbol{x}\|, \|\boldsymbol{y}\| > r$ and show that:

$$
\begin{aligned}
r^2 \cdot \left(\frac{\|\boldsymbol{x}\|}{\|\boldsymbol{y}\|} + \frac{\|\boldsymbol{y}\|}{\|\boldsymbol{x}\|} - 2\hat{\boldsymbol{x}} \cdot \hat{\boldsymbol{y}}\right) &\leq \|\boldsymbol{x}\|\|\boldsymbol{y}\|\left(\frac{\|\boldsymbol{x}\|}{\|\boldsymbol{y}\|} + \frac{\|\boldsymbol{y}\|}{\|\boldsymbol{x}\|} - 2\hat{\boldsymbol{x}} \cdot \hat{\boldsymbol{y}}\right) \\
&= \|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - \underbrace{2(\|\boldsymbol{x}\|)(\|\boldsymbol{y}\|)\hat{\boldsymbol{x}} \cdot \hat{\boldsymbol{y}}}_{2\boldsymbol{x} \cdot \boldsymbol{y}} \\
&= \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \tag{36}
\end{aligned}
$$

[b]This choice of exploring large values of SR for the hyperspherical case is motivated by the fact that condition Eq. 3 must be satisfied in order for the network to work properly: choosing a large SR will also lead to larger σmin(W), as discussed in the 'Universal function approximation property' section.

proving the contractivity condition (34).

We see that the only condition needed is $\|x\|, \|y\| > r$, which means that the linear part of the update (2a) must map states outside the hyper-sphere of radius r. Finally, by applying properties of norms, we observe that:

$$\|W \cdot x_k + W_{in} \cdot s_k\| \geq \|W \cdot x_k\| - \|W_{in} \cdot s_k\| \geq \sigma_{min}(W)r - \|W_{in}\| \|s_{max}\| \tag{37}$$

and asking this to be larger than r results in the condition (3).

Input-driven dynamics. Here we show how to derive (13). Consider a dynamical system evolving according to (2): if we explicitly expand the first steps from the initial state x_0 we obtain:

$$x_1 = \frac{Wx_0 + u_1}{\|Wx_0 + u_1\|} = \frac{Wx_0 + u_1}{N_1} = \frac{W}{N_1}x_0 + \frac{1}{N_1}u_1 \tag{38}$$

$$x_2 = \frac{Wx_1 + u_1}{N_2} = \frac{W^2}{N_2 N_1}x_0 + \frac{W}{N_2 N_1}u_1 + \frac{1}{N_2}u_2 \tag{39}$$

$$x_3 = \frac{Wx_2 + u_3}{N_3} = \frac{W^3}{N_3 N_2 N_1}x_0 + \frac{W^2}{N_3 N_2 N_1}u_1 + \frac{W}{N_3 N_2}u_2 + \frac{1}{N_3}u_3 \tag{40}$$

So that the general case can be written as:

$$\begin{aligned} x_n &= \frac{Wx_{n-1} + u_n}{N_n} \\ &= \frac{W^n}{N_n N_{n-1}\cdots N_1}x_0 + \frac{W^{n-1}}{N_n N_{n-1}\cdots N_1}u_1 \\ &\quad + \frac{W^{n-2}}{N_n N_{n-1}\cdots N_2}u_2 + \cdots + \frac{1}{N_n}u_n \end{aligned} \tag{41}$$

and in a more compact form as follows:

$$x_n = M^{(n,0)}x_0 + \sum_{k=1}^{n} M^{(n,k)}u_k \tag{42}$$

where

$$M^{(n,k)} := \frac{W^{n-k}}{\prod_{l=k}^{n} N_l} \tag{43}$$

and $N_0 = 1$.

References

1. Sussillo, D. & Barak, O. Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation* **25**, 626–649, https://doi.org/10.1162/NECO_a_00409 (2013).
2. Ceni, A., Ashwin, P. & Livi, L. Interpreting recurrent neural networks behaviour via excitable network attractors. *Cognitive Computation*, https://doi.org/10.1007/s12559-019-09634-2 (2019).
3. Jaeger, H. & Haas, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science* **304**, 78–80, https://doi.org/10.1126/science.1091277 (2004).
4. Maass, W., Natschläger, T. & Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation* **14**, 2531–2560, https://doi.org/10.1162/089976602760407955 (2002).
5. Grigoryeva, L. & Ortega, J.-P. Echo state networks are universal. *Neural Networks* **108**, 495–508, https://doi.org/10.1016/j.neunet.2018.08.025 (2018).
6. Pathak, J., Lu, Z., Hunt, B. R., Girvan, M. & Ott, E. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **27**, 121102, https://doi.org/10.1063/1.5010300 (2017).
7. Pathak, J., Hunt, B., Girvan, M., Lu, Z. & Ott, E. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical Review Letters* **120**, 024102, https://doi.org/10.1103/PhysRevLett.120.024102 (2018).
8. Pathak, J. *et al.* Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **28**, 041101, https://doi.org/10.1063/1.5028373 (2018).
9. Bianchi, F. M., Scardapane, S., Uncini, A., Rizzi, A. & Sadeghian, A. Prediction of telephone calls load using echo state network with exogenous variables. *Neural Networks* **71**, 204–213, https://doi.org/10.1016/j.neunet.2015.08.010 (2015).
10. Bianchi, F. M., Scardapane, S., Løkse, S. & Jenssen, R. Reservoir computing approaches for representation and classification of multivariate time series. *arXiv preprint arXiv:1803.07870* (2018).
11. Palumbo, F., Gallicchio, C., Pucci, R. & Micheli, A. Human activity recognition using multisensor data fusion based on reservoir computing. *Journal of Ambient Intelligence and Smart Environments* **8**, 87–107 (2016).
12. Gallicchio, C., Micheli, A. & Pedrelli, L. Comparison between deepesns and gated rnns on multivariate time-series prediction. *arXiv preprint arXiv:1812.11527* (2018).
13. Sompolinsky, H., Crisanti, A. & Sommers, H.-J. Chaos in random neural networks. *Physical Review Letters* **61**, 259, https://doi.org/10.1103/PhysRevLett.61.259 (1988).
14. Livi, L., Bianchi, F. M. & Alippi, C. Determination of the edge of criticality in echo state networks through Fisher information maximization. *IEEE Transactions on Neural Networks and Learning Systems* **29**, 706–717, https://doi.org/10.1109/TNNLS.2016.2644268 (2018).

15. Verzelli, P., Livi, L. & Alippi, C. A characterization of the edge of criticality in binary echo state networks. In *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, 1–6 (IEEE, 2018).
16. Legenstein, R. & Maass, W. Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks* **20**, 323–334, https://doi.org/10.1016/j.neunet.2007.04.017 (2007).
17. Bertschinger, N. & Natschläger, T. Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation* **16**, 1413–1436, https://doi.org/10.1162/089976604323057443 (2004).
18. Rajan, K., Abbott, L. F. & Sompolinsky, H. Stimulus-dependent suppression of chaos in recurrent neural networks. *Physical Review E* **82**, 011903, https://doi.org/10.1103/PhysRevE.82.011903 (2010).
19. Rivkind, A. & Barak, O. Local dynamics in trained recurrent neural networks. *Physical Review Letters* **118**, 258101, https://doi.org/10.1103/PhysRevLett.118.258101 (2017).
20. Gallicchio, C. Chasing the echo state property. *arXiv preprint arXiv:1811.10892* (2018).
21. Yildiz, I. B., Jaeger, H. & Kiebel, S. J. Re-visiting the echo state property. *Neural Networks* **35**, 1–9, https://doi.org/10.1016/j.neunet.2012.07.005 (2012).
22. Manjunath, G. & Jaeger, H. Echo state property linked to an input: Exploring a fundamental characteristic of recurrent neural networks. *Neural Computation* **25**, 671–696, https://doi.org/10.1162/NECO_a_00411 (2013).
23. Wainrib, G. & Galtier, M. N. A local echo state property through the largest Lyapunov exponent. *Neural Networks* **76**, 39–45, https://doi.org/10.1016/j.neunet.2015.12.013 (2016).
24. Tiňo, P. & Rodan, A. Short term memory in input-driven linear dynamical systems. *Neurocomputing* **112**, 58–63, https://doi.org/10.1016/j.neucom.2012.12.041 (2013).
25. Goudarzi, A. *et al.* Memory and information processing in recurrent neural networks. *arXiv preprint arXiv:1604.06929* (2016).
26. Jaeger, H. *Short term memory in echo state networks*, vol. 5 (GMD-Forschungszentrum Informationstechnik, 2002).
27. Ganguli, S., Huh, D. & Sompolinsky, H. Memory traces in dynamical systems. *Proceedings of the National Academy of Sciences* **105**, 18970–18975, https://doi.org/10.1073/pnas.0804451105 (2008).
28. Dambre, J., Verstraeten, D., Schrauwen, B. & Massar, S. Information processing capacity of dynamical systems. *Scientific Reports* **2**, https://doi.org/10.1038/srep00514 (2012).
29. Verstraeten, D., Dambre, J., Dutoit, X. & Schrauwen, B. Memory versus non-linearity in reservoirs. In *IEEE International Joint Conference on Neural Networks*, 1–8 (IEEE, Barcelona, Spain, 2010).
30. Inubushi, M. & Yoshimura, K. Reservoir computing beyond memory-nonlinearity trade-off. *Scientific Reports* **7**, 10199, https://doi.org/10.1038/s41598-017-10257-6 (2017).
31. Marzen, S. Difference between memory and prediction in linear recurrent networks. *Physical Review E* **96**, 032308, https://doi.org/10.1103/PhysRevE.96.032308 (2017).
32. Tiňo, P. Asymptotic fisher memory of randomized linear symmetric echo state networks. *Neurocomputing* **298**, 4–8 (2018).
33. Andrecut, M. Reservoir computing on the hypersphere. *International Journal of Modern Physics C* **28**, 1750095, https://doi.org/10.1142/S0129183117500954 (2017).
34. Scardapane, S., Van Vaerenbergh, S., Totaro, S. & Uncini, A. Kafnets: Kernel-based non-parametric activation functions for neural networks. *Neural Networks* **110**, 19–32, https://doi.org/10.1016/j.neunet.2018.11.002 (2019).
35. Lukoševičius, M. & Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Computer Science Review* **3**, 127–149, https://doi.org/10.1016/j.cosrev.2009.03.005 (2009).
36. Sussillo, D. & Abbott, L. F. Generating coherent patterns of activity from chaotic neural networks. *Neuron* **63**, 544–557, https://doi.org/10.1016/j.neuron.2009.07.018 (2009).
37. Siegelmann, H. T. & Sontag, E. D. On the computational power of neural nets. *Journal of computer and system sciences* **50**, 132–150, https://doi.org/10.1006/jcss.1995.1013 (1995).
38. Hammer, B. On the approximation capability of recurrent neural networks. *Neurocomputing* **31**, 107–123, https://doi.org/10.1016/S0925-2312(99)00174-5 (2000).
39. Hammer, B. & Tiňo, P. Recurrent neural networks with small weights implement definite memory machines. *Neural Computation* **15**, 1897–1929 (2003).
40. Lax, P. D. *Functional analysis*. Pure and Applied Mathematics: A Wiley-Interscience Series of Texts, Monographs and Tracts (Wiley, 2002).
41. Verstraeten, D. & Schrauwen, B. On the quantification of dynamics in reservoir computing. In *Artificial Neural Networks–ICANN 2009*, 985–994, https://doi.org/10.1007/978-3-642-04274-4_101 (Springer Berlin Heidelberg, 2009).
42. Di Gregorio, E., Gallicchio, C. & Micheli, A. Combining memory and non-linearity in echo state networks. In *International Conference on Artificial Neural Networks*, 556–566 (Springer, 2018).

Acknowledgements

Author Contributions

L.L. outlined the research ideas. P.V. conceived the methods and performed the experiments. C.A. contributed to the technical discussion. All authors developed the theoretical framework, took part to the paper writing and approved the final manuscript.

Additional Information

Competing Interests: The authors declare no competing interests.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.