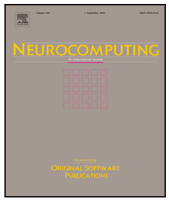
Neurocomputing 579 (2024) 127411

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Euler State Networks: Non-dissipative Reservoir Computing

Claudio Gallicchio

University of Pisa, Department of Computer Science, Largo Bruno Pontecorvo, 3, Pisa, 56127, Italy

ARTICLE INFO

Communicated by B. Shen

Keywords:
Reservoir computing
Echo state networks
Recurrent neural networks
Stable neural architectures

ABSTRACT

Inspired by the numerical solution of ordinary differential equations, in this paper, we propose a novel Reservoir Computing (RC) model, called the Euler State Network (EuSN). The presented approach makes use of forward Euler discretization and antisymmetric recurrent matrices to design reservoir dynamics that are both stable and non-dissipative by construction.

Our mathematical analysis shows that the resulting model is biased towards a unitary effective spectral radius and zero local Lyapunov exponents, intrinsically operating near the edge of stability. Experiments on long-term memory tasks show the clear superiority of the proposed approach over standard RC models in problems requiring effective propagation of input information over multiple time steps. Furthermore, results on time-series classification benchmarks indicate that EuSN can match (or even exceed) the accuracy of trainable Recurrent Neural Networks, while retaining the training efficiency of the RC family, resulting in up to ≈ 464-fold savings in computation time and $\approx$ 1750-fold savings in energy consumption. At the same time, our results on time-series modeling tasks show competitive results against standard RC when the architecture is complemented by direct input-readout connections.

1. Introduction

The interest in studying neural network architectures from a dynamical system perspective has recently been attracting increasing research attention [2–5]. The key insight is that the computation performed by some kinds of neural networks, e.g., residual networks, can be understood as the numerical solution to an ordinary differential equation (ODE) through discretization [6]. This intuitively simple observation brings about the possibility of imposing desirable properties in the behavior of the neural network by imposing specific conditions on the corresponding ODE. Stability plays a key role in this sense, being related to the propagation of both input signals, during inference, and gradients, during training.

In this paper, we focus on Recurrent Neural Network (RNN) architectures, and especially on Reservoir Computing (RC) [7,8]. RC establishes a particularly attractive approach for the design and training of RNNs, where the hidden recurrent *reservoir* layer is left untrained, leaving the entire training effort to the output layer alone. The stability of forward signal propagation through the reservoir is therefore of great importance. Since the parameters that determine the dynamics of the reservoir are untrained, some form of constraint is indeed necessary to avoid instability when the network is operated with a driving input. This aspect is related to the widely known fading memory property of RC networks, which makes it difficult to maintain the input signal information in the state dynamics across several time-steps. Despite this limitation, RC has become more and more popular because of the impressively advantageous trade-off between predictive performance and training efficiency. As a result, it is often the chosen approach for embedded applications distributed at the edge [9,10], as well as for RNN implementations in neuromorphic hardware [11–15]. However, the performance gap with fully trainable state-of-the-art RNNs still leaves space for improvements.

This paper extends our previous work in [1],[1] introducing and analyzing a new RC method that is inspired by the numerical solution of ODEs. Specifically, the model we propose in this paper uses a fundamental numerical technique for the integration of differential equations, i.e., the forward Euler method, to discretize the continuous

E-mail address: gallicch@di.unipi.it.

[1] Specifically, the journal version extends the proceedings version [1] in several aspects. First, we have extended the description of the EuSN model and its introduction from the dynamical systems perspective. Second, we have introduced a thorough mathematical characterization of the computational properties and architectural bias of the proposed non-dissipative approach to RC design, analyzing asymptotic stability properties, effective spectral radius, and local Lyapunov exponents of EuSNs. Moreover, the extended version includes a substantially deeper experimental analysis, including new experiments on long-term information retention tasks, a significantly larger set of time-series classification benchmarks, and new experiments on time-series modeling tasks. Furthermore, the experiments in this version cover a wider range of model hyper-parameterizations, a substantially larger pool of baseline models, and an extended evaluation of the accuracy-efficiency trade-off (which now includes a comparative estimation of energy consumption).

https://doi.org/10.1016/j.neucom.2024.127411
Received 24 March 2023; Received in revised form 1 December 2023; Accepted 16 February 2024
Available online 24 February 2024

dynamics of a suitably constrained reservoir ODE system. The discretization allows for the explicit computation of the network state at each time step, analogous to how the Euler method steps through the solution of an ODE. As the reservoir architecture is obtained by forward Euler discretization of an intrinsically stable and non-dissipative ODE, the resulting model is called the *Euler State Network* (EuSN). The initialization condition imposed on the reservoir inherently yields state dynamics that are biased to be both stable and non-dissipative, thereby mitigating the lossy transmission of input signals over time, sensibly reducing the performance gap with fully trainable RNNs in time-series classification tasks.

The contributions of this work can be summarized as follows:

- We introduce the EuSN model, a novel approach to designing RC neural network architectures based on ODE discretization. Overcoming the natural limitations of standard RC models, EuSN is specifically designed to fulfill both stability and non-dissipative properties.
- We derive a mathematical characterization of the proposed reservoir system, focusing on its asymptotic stability and local Lyapunov exponents.
- We provide an extensive experimental evaluation of EuSN. First, we shed light on its long-term memorization abilities in comparison to standard RC models. Then, we assess the accuracy-efficiency trade-off of EuSN in comparison to both RC and fully trainable neural models on several time-series classification benchmarks. Finally, we exercise the EuSN approach on tasks on time-series modeling.

The rest of this paper is organized as follows. In Section 2 we give an overview of the basic concepts of RC. We introduce the EuSN model in Section 3, and we elaborate its mathematical analysis in Section 4. The experimental assessment of the approach is provided in Section 5. Finally, Section 6 concludes the paper.

2. Reservoir computing and echo state networks

Reservoir Computing (RC) is a common denomination for a class of recurrent neural models that are based on untrained internal dynamics [7,8,13]. In general, the architecture comprises a fixed hidden recurrent layer called *reservoir*, and an output *readout* layer that is the only trainable component. Here, in particular, we focus on the Echo State Network (ESN) model [16,17], where the reservoir operates in discrete time-steps and typically makes use of tanh non-linearity. We consider a reservoir architecture with N reservoir neurons and X input units, using $\mathbf{h}(t)$ and $\mathbf{x}(t)$ to denote, respectively, the state of the reservoir and the external input at time-step t. Referring to the general ESN formulation with leaky integrator neurons [18], the reservoir state is evolved according to the following iterated map:

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

where $\alpha \in (0,1]$ is the leaking rate, $\mathbf{W}_h \in \mathbb{R}^{N\times N}$ is the recurrent reservoir weight matrix, $\mathbf{W}_x \in \mathbb{R}^{N\times X}$ is the input weight matrix, and $\mathbf{b} \in \mathbb{R}^N$ is a vector of bias weights. As an initial condition, the reservoir state is typically started at the origin, i.e., the null vector: $\mathbf{h}(0) = \mathbf{0}$.

Interestingly, all the weight values that parametrize the behavior of the reservoir, i.e., the values in $\mathbf{W}_h$, $\mathbf{W}_x$ and $\mathbf{b}$ can be left untrained provided that the state dynamics satisfy a global asymptotic stability property called the Echo State Property (ESP) [17,19]. Essentially, the ESP states that when driven by a long input time-series, reservoir dynamics tend to synchronize [20] irrespectively of initial conditions, whose influence on the state progressively fades away. Tailoring the reservoir initialization to the properties of the input signal and task still represents a challenging theoretical research question [21,22]. However, in RC practice, it is common to follow a simple initialization process linked to a necessary condition for the ESP [8,17,23]. The recurrent reservoir weight matrix $\mathbf{W}_h$ is initialized randomly, e.g., from a uniform distribution over $[-1,1]$, and then re-scaled to limit its spectral radius $\rho(\mathbf{W}_h)$ to a value smaller than 1. Notice that this would require computing the eigenvalues of potentially large matrices, and thereby might result in a computationally demanding process. A more convenient procedure, in this case, can be obtained by using the circular law from random matrix theory, for instance, generating random weights from a uniform distribution whose parameters depend on the desired value of the spectral radius, as illustrated in [24]. The weight values in $\mathbf{W}_x$ and $\mathbf{b}$ are randomly drawn from a uniform distribution over $[-\omega_x,\omega_x]$ and $[-\omega_b,\omega_b]$, respectively. The values of α, $\rho(\mathbf{W}_h)$, ω_x, and ω_b are treated as hyper-parameters, and are tuned by model selection.

The trainable readout can be implemented in several forms [8]. In this paper, when focusing on classification tasks, we use a dense linear output layer that is fed by the last state computed by the reservoir on an input time-series. When dealing with regression tasks (e.g., time-series modeling), we use as readout a dense linear output layer that is fed by the state computed by the reservoir at each time step. In the latter case, it is also common to consider a practical architectural variant in which direct (and trainable) input-to-readout connections are included.

ESP limitations – Regarding the ESP, it is worth noticing that while enabling a stable encoding robust to input perturbations, the required stability property gives a fading memory structure to the reservoir dynamics. As shown in [25], the reservoir state space tends to organize in a suffix-based fashion, in agreement with the architectural Markovian bias of contractive RNNs [26,27]. On the one hand, this means that ESNs are particularly well suited to solve tasks defined in compliance with this characterization, i.e., where the information in the suffix of the input time-series is dominant in determining the output [25]. On the other hand, reservoir dynamics are constrained to progressively forget previous stimuli, making it difficult to propagate input information efficiently through many time steps, as might be required for example in time-series classification tasks.

Ring Reservoirs – While randomization plays an important role in RC, several studies in literature tried to find reservoir organizations that perform better than just random ones. A fruitful line of research deals with constraining the reservoir topology [28] to get desirable algebraic properties of the recurrent weight matrix $\mathbf{W}_h$, especially based on an orthogonal structure [29,30]. A simple yet effective way to achieve this goal consists of shaping the reservoir topology such that the reservoir neurons are arranged to form a cycle, or a *ring*. This specific architectural variant of ESNs has been shown to have noticeable advantages, e.g., in terms of the richness of the developed dynamics, longer memory, and performance on non-linear tasks [31–33]. In this paper, we thereby consider ESN based on ring reservoirs (R-ESN) as a competitive alternative to the standard vanilla setup.

3. The Euler state network model

To introduce the proposed model, we start by considering the operation of a continuous-time recurrent neural system modeled by the following ODE:

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

where $\mathbf{h}(t) \in \mathbb{R}^N$ and $\mathbf{x}(t) \in \mathbb{R}^X$ respectively denote the state and the driving input, $\mathbf{W}_h \in \mathbb{R}^{N\times N}$ is the recurrent weight matrix, $\mathbf{W}_x \in \mathbb{R}^{N\times X}$ is the input weight matrix, and $\mathbf{b} \in \mathbb{R}^N$ is the bias.

We are interested in applying two types of constraints to the system in Eq. (2), namely *stability* and *non-dissipativity*. The former is required to develop a robust information processing system across time steps, avoiding the explosion of input perturbations that would result in poor generalization. The latter is important to avoid developing lossy

dynamics, which would determine catastrophic forgetting of past inputs during the state evolution.

In what follows, to develop our model, we make use of results from the stability theory of autonomous dynamical systems. As the system in Eq. (2) is non-autonomous, the mathematical analysis is then approximated in the sense that stability and non-dissipativity can be interpreted as referred to small perturbations in the state of the system. Notwithstanding this approximation, the mathematical analysis reported here is useful for deriving empirical initialization conditions in line with the RC literature [7], as well as with the analysis presented in [34] for fully trainable RNN models.

In the context of ODE stability, a fundamental role is played by the Jacobian of the system dynamics. In our case, the Jacobian of the system in Eq. (2) is given by the following equation:

$$\mathbf{J}_f(\mathbf{h}(t), \mathbf{x}(t)) = \frac{\partial f(\mathbf{h}(t), \mathbf{x}(t))}{\partial \mathbf{h}(t)} = \mathbf{D}(t)\,\mathbf{W}_h, \tag{3}$$

in which $\mathbf{D}(t)$ is a diagonal matrix whose non-zero entries are given by $1-\tilde{\mathbf{h}}_1^2(t)$, $1-\tilde{\mathbf{h}}_2^2(t)$, …, $1-\tilde{\mathbf{h}}_N^2(t)$, with $\tilde{\mathbf{h}}_i(t)$ denoting the ith component of the vector $\tanh\Big(\mathbf{W}_h\mathbf{h}(t)+\mathbf{W}_x\mathbf{x}(t)+\mathbf{b}\Big)$.

The interested reader can find a precise formulation of the stability of ODEs solutions in [35,36]. Here, we limit ourselves to recall that the solution to an ODE is stable if the real part of all eigenvalues of the corresponding Jacobian are ≤ 0.[2] Using $\lambda_k(\cdot)$ to indicate the kth eigenvalue of its matrix argument, the stability constraint in our case can be expressed as

$$\max_k Re(\lambda_k(\mathbf{J}_f(\mathbf{h}(t), \mathbf{x}(t)))) \leq 0. \tag{4}$$

Essentially, the real parts of the eigenvalues of the Jacobian indicate the degree of local "expansion" (or "contraction") of the state along specific dimensions of the space. As discussed in [6,34], having a 0 upper bound on these quantities is important to avoid unbounded amplification of signal transmission. On the other hand, if the real parts of these eigenvalues are $\ll 0$, then the resulting dynamics are lossy, in the sense that the state information is progressively shrunk exponentially rapidly over time.[3] We thereby seek a critical condition under which the eigenvalues of the Jacobian of our ODE are featured by ≈ 0 real parts, i.e.:

$$Re(\lambda_k(\mathbf{J}_f(\mathbf{h}(t), \mathbf{x}(t)))) \approx 0 \quad k = 1, \ldots, N. \tag{5}$$

A rather simple way to meet the condition in Eq. (5) is to require that the recurrent weight matrix $\mathbf{W}_h$ in Eq. (2) is antisymmetric [6], i.e., it satisfies the condition:

$$\mathbf{W}_h = -\mathbf{W}_h^T. \tag{6}$$

In fact, under the mild assumption that the diagonal matrix $\mathbf{D}(t)$ is invertible (i.e., in our case for neurons in a non-saturating regime), the eigenvalues of the Jacobian $\mathbf{J}_f(\mathbf{h}(t), \mathbf{x}(t)) = \mathbf{D}(t)\,\mathbf{W}_h$, are all purely imaginary.[4] Crucially, this property is *intrinsic* to the antisymmetric structure imposed to the recurrent weight matrix $\mathbf{W}_h$, irrespectively of the particular choice of its values. In other words, the targeted critical dynamical behavior can be seen as an *architectural bias* of the neural system described by Eq. (2) under the imposed antisymmetric constraint in Eq. (6). Taking an RC perspective, we can then leave all the weight values in Eq. (2), i.e., $\mathbf{W}_h$, $\mathbf{W}_x$ and $\mathbf{b}$, *untrained*, provided that Eq. (6) is satisfied.

Finally, to develop our discrete-time reservoir dynamics, we note that the ODE in Eq. (2) can be solved numerically by using the forward Euler method [39], resulting in the following discretization:

$$\begin{aligned}\mathbf{h}(t) &= F(\mathbf{h}(t-1), \mathbf{x}(t)) \\ &= \mathbf{h}(t-1) + \\ &\quad \varepsilon \tanh\Big((\mathbf{W}_h - \gamma \mathbf{I})\mathbf{h}(t-1) + \mathbf{W}_x\mathbf{x}(t) + \mathbf{b}\Big),\end{aligned} \tag{7}$$

where ε is the step size of integration, and γ is a diffusion coefficient used for stabilizing the discrete forward propagation [6]. Note that both ε and γ are typically small positive scalars, and are treated as hyper-parameters. We can now look at Eq. (7) as the state transition equation of a discrete-time *reservoir* layer with N recurrent neurons and X input units, where $\mathbf{W}_h$, $\mathbf{W}_x$ and $\mathbf{b}$ are untrained, and $\mathbf{W}_h$ satisfies the constraint of being antisymmetric in Eq. (6). Since the untrained reservoir dynamics evolve as the forward Euler solution to an ODE, we call the resulting model the *Euler State Network* (EuSN). As in standard RC/ESN, the reservoir state is initialized at the origin, i.e., $\mathbf{h}(0) = \mathbf{0}$. Moreover, the reservoir feeds a *readout* layer, which is the only trained component of the EuSN architecture. We implement the readout layer of a EuSN as a linear dense layer as in standard ESN practice (i.e., in accordance with Section 2).

To *initialize* the reservoir, we propose the following simple procedure, analogous to the common approaches in RC literature:

- To set the recurrent weights, we start by randomly initializing the values of a matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$ from a uniform distribution over $[-\omega_r, \omega_r]$, where ω_r is a positive *recurrent scaling* coefficient. We then set $\mathbf{W}_h = \mathbf{W} - \mathbf{W}^T$, which satisfies Eq. (6) as it is antisymmetric by construction.
- The values of the input weight matrix $\mathbf{W}_x$ are randomly initialized from a uniform distribution over $[-\omega_x, \omega_x]$, where ω_x is a positive *input scaling coefficient* as in ESNs.
- The values of the bias vector $\mathbf{b}$ are randomly initialized from a uniform distribution over $[-\omega_b, \omega_b]$, where ω_b is a positive *bias scaling coefficient* as in ESNs.

Notice that the stability properties of the reservoir are due to the antisymmetric structure of $\mathbf{W}_h$, without requiring to scale its spectral radius as in ESNs. The recurrent, input, and bias scaling coefficients ω_r, ω_x, and ω_b, are introduced simply to balance the contributions of the different terms in the state transition Eq. (7), and are treated as hyper-parameters. The values of these hyper-parameters, along with those of ε, γ, and the number of reservoir neurons N, should be tuned on a validation set for each task to be addressed (as in standard RC practice).

It is worth mentioning that, in the broader landscape of trainable RNN research, the introduced EuSN model brings similarities to the Antisymmetric RNN (A-RNN) [34], which is featured by state dynamics evolving as in Eq. (7), under a similar antisymmetric constraint on the recurrent weight matrix. Crucially, while in EuSN all the internal weights are left untrained after initialization, in A-RNN they are all trainable. In this view, the study of EuSN can be seen under the perspective of randomized neural networks [40], emphasizing the intrinsic capabilities of stable recurrent architectures based on forward Euler discretization and antisymmetric recurrent weight matrices even in the absence (or prior to) training of internal connections.

The following Section 4 delves into the mathematical analysis of the EuSN reservoir system, studying its asymptotic stability properties and its intrinsically critical dynamical regime.

[2] In the context of non-autonomous ODEs, this condition comes down to a weakly contractive constraint on the system's dynamics.

[3] The solution of the linearized system evolves state perturbations proportionally to the exponential of the Jacobian multiplied by time [36]. The real parts of the eigenvalues of the Jacobian then determine the rate of exponential amplification (if positive) or shrinking (if negative) of state perturbations. In particular, when the real parts of such eigenvalues are $\ll 0$, state perturbations quickly vanish and no longer have any influence on the dynamics.

[4] The eigenvalues of $\mathbf{W}_h$ are all purely imaginary due to its antisymmetric structure (see, e.g., [37,38]). Using $\mathbf{S}$ to denote the square root of $\mathbf{D}(t)$, i.e., $\mathbf{S} = \mathbf{D}(t)^{1/2}$, it can be seen that $\mathbf{D}(t)\,\mathbf{W}_h$ is similar to $\mathbf{S}\,\mathbf{W}_h\,\mathbf{S}$ (a change of basis matrix is given by $\mathbf{P} = \mathbf{D}(t)^{-1/2}$), and thereby has its same eigenvalues. As $\mathbf{S}\,\mathbf{W}_h\,\mathbf{S}$ is itself antisymmetric, it follows that the eigenvalues of $\mathbf{D}(t)\,\mathbf{W}_h$ are all purely imaginary. The interested reader can find a different proof, under the same conditions, in [34].

4. Mathematical analysis

We study the reservoir of a EuSN as a discrete-time dynamical system evolving according to the function F defined in Eq. (7), where we remind that $\mathbf{W}_h$ is antisymmetric, and both ε and γ are small positive reals. We use here the same mathematical approximations used in Section 3 to define the model. The local behavior of the reservoir system around a specific state $\mathbf{h}_0$ can be analyzed by linearization, i.e., by approximating its dynamics as follows:

$$\mathbf{h}(t) = \mathbf{J}_F(\mathbf{h}_0, \mathbf{x}(t))\,(\mathbf{h}(t-1) - \mathbf{h}_0) + F(\mathbf{h}_0, \mathbf{x}(t)). \tag{8}$$

Here $\mathbf{J}_F(\mathbf{h}_0, \mathbf{x}(t))$ is used to denote the Jacobian of the function F in Eq. (7) at time-step t and evaluated at $\mathbf{h}_0$, i.e.:

$$\mathbf{J}_F(\mathbf{h}_0, \mathbf{x}(t)) = \frac{\partial F(\mathbf{h}(t-1), \mathbf{x}(t))}{\partial \mathbf{h}(t-1)}\Big|_{\mathbf{h}=\mathbf{h}_0}. \tag{9}$$

In classical ESN literature, it is common to study the eigenspectrum of the Jacobian of the reservoir, and especially its spectral radius, to characterize the asymptotic stability behavior of the system. In particular, when introducing necessary conditions for the ESP [17], it is common to study the autonomous system (i.e., in the absence of driving input $\mathbf{x}(t) = \mathbf{0}$ for every t, and bias $\mathbf{b} = \mathbf{0}$), linearized around the origin (i.e., $\mathbf{h}_0 = \mathbf{0}$). For ESNs, this procedure then results in an algebraic constraint that is imposed on the recurrent weight matrix of the reservoir to control its effective spectral radius. Interestingly, by taking the same assumptions in the case of EuSN, we find that all the eigenvalues of the Jacobian are naturally confined to values close to 1. More precisely, the following Proposition 1 characterizes the values that can be taken by the eigenvalues of the Jacobian of the autonomous EuSN reservoir system linearized around the origin. In the following, we use the notation $\lambda_k(\cdot)$ to indicate the kth eigenvalue of its argument.

Proposition 1. *Let us consider a EuSN whose reservoir dynamics are ruled by the state transition function in Eq. (7), with step size ε, diffusion coefficient γ, and reservoir weight matrix $\mathbf{W}_h$. Assume autonomous (i.e., with null driving input and bias) and linearized dynamics around the origin. Then, the eigenvalues of the resulting Jacobian satisfy the following condition:*

$$\begin{aligned} &\text{for } k = 1, \dots, N: \\ &Re\big(\lambda_k(\mathbf{J}_F(\mathbf{0}, \mathbf{0}))\big) = 1 - \varepsilon\,\gamma, \\ &Im\big(\lambda_k(\mathbf{J}_F(\mathbf{0}, \mathbf{0}))\big) \in [-\varepsilon\,\rho(\mathbf{W}_h),\ \varepsilon\,\rho(\mathbf{W}_h)], \end{aligned} \tag{10}$$

Proof. Computing an expression for the Jacobian in Eq. (9) around the origin and in the absence of external input, we find that:

$$\mathbf{J}_F(\mathbf{0}, \mathbf{0}) = \mathbf{I} + \varepsilon(\mathbf{W}_h - \gamma\mathbf{I}) = (1 - \varepsilon\,\gamma)\mathbf{I} + \varepsilon\,\mathbf{W}_h. \tag{11}$$

We can notice that each eigenvalue of the sum of matrices in the right-hand side of Eq. (11) is given by an eigenvalue of $\mathbf{I}(1-\varepsilon\gamma)$ added to an eigenvalue of $\varepsilon\mathbf{W}_h$ (in fact, the two matrices commute). Thereby, we have:

$$\lambda_k(\mathbf{J}_F(\mathbf{0}, \mathbf{0})) = 1 - \varepsilon\,\gamma + \varepsilon\,\lambda_k(\mathbf{W}_h) \quad k = 1, \dots, N. \tag{12}$$

As matrix $\mathbf{W}_h$ is antisymmetric, its eigenvalues are purely imaginary. In particular, except for a 0 eigenvalue in correspondence to the odd-dimensional case, they come in the form $\lambda_k(\mathbf{W}_h) = i\,\beta_k$, where i is used to denote the imaginary unit, and $\beta_k = Im(\lambda_k(\mathbf{W}_h))$ can take the possible values $\pm b_1$, …, $\pm b_{\lfloor N/2 \rfloor}$. We can then rewrite Eq. (12) as follows:

$$\lambda_k(\mathbf{J}_F(\mathbf{0}, \mathbf{0})) = 1 - \varepsilon\,\gamma + i\,\varepsilon\,\beta_k \quad k = 1, \dots, N, \tag{13}$$

from which it is straightforward to conclude that $Re\big(\lambda_k(\mathbf{J}_F(\mathbf{0}, \mathbf{0}))\big) = 1 - \varepsilon\,\gamma$. Moreover, observing that $\beta_k \in [-\rho(\mathbf{W}_h), \rho(\mathbf{W}_h)]$, we can also derive that $Im\big(\lambda_k(\mathbf{J}_F(\mathbf{0}, \mathbf{0}))\big) \in [-\varepsilon\,\rho(\mathbf{W}_h), \varepsilon\,\rho(\mathbf{W}_h)]$. □

Proposition 1 enables us to locate the eigenvalues of $\mathbf{J}_F(\mathbf{0}, \mathbf{0})$ as a function of ε, γ and $\rho(\mathbf{W}_h)$. Recalling that, in practice, both ε and γ are small positive scalars, we can observe that $\lambda_k(\mathbf{J}_F(\mathbf{0}, \mathbf{0})) \approx 1$ for $k = 1, \dots, N$. This has interesting consequences in terms of the asymptotic behavior of the reservoir. Indeed, as all the eigenvalues of the Jacobian $\mathbf{J}_F(\mathbf{0}, \mathbf{0})$ are ≈ 1, it follows that the autonomous operation of the reservoir system around the origin tends to preserve all the components of the state while performing infinitesimal rotations due to the antisymmetric structure of the recurrent matrix. Any perturbation to the zero state (i.e., of the origin) is preserved over time without vanishing, which essentially means that EuSNs do not tend to show the ESP under the proposed initialization conditions.

The qualitative difference between the behaviors of a EuSN and those of an ESN is graphically illustrated in Fig. 1. In the figure, taking inspiration from the graphical analysis of the inner dynamics of deep architectures used in [6,34], we visualize examples of state trajectories developed by 2-dimensional autonomous reservoirs in different conditions: ESN with and without ESP, R-ESN with orthogonally shaped weight matrix, and EuSN. In the case of ESNs, when the ESP is valid (Fig. 1(a)), the origin is an attractor and all the trajectories asymptotically converge to it. When instead the ESN does not satisfy the ESP (Fig. 1(b)), the origin is a repeller. In the case of R-ESN (Fig. 1(c)), even if the recurrent weight matrix has an orthogonal-based structure, trajectories still converge to the origin (more slowly than in Fig. 1(a)) given the contractive properties of the tanh non-linearity. Interestingly, in the case of EuSN (Fig. 1(d)), the states rotate around the origin without being attracted nor repelled, and the distance between the orbits is preserved, reflecting the differences among the initial conditions. In other words,[5] differently from ESN variants, the effect of the external input on the reservoir dynamics in EuSN does not die out nor explode with time. This qualitative analysis is confirmed by the quantitative results reported in Fig. 2, which provides the average Hausdorff distance [41] between the reservoir trajectories from perturbed states,[6] in correspondence to the same conditions used in Fig. 1. In cases when ESP is valid, all trajectories converge after a transient that is rather short for the ESN, and longer for the case of R-ESN. When the ESP is not valid, the divergence of state trajectories of the same reservoir starting from different initial conditions is visible in the initial increasing trend in the average Hausdorff distance. Finally, in the case of EuSN, the distance between the reservoir state trajectories stays essentially constant over long time spans.

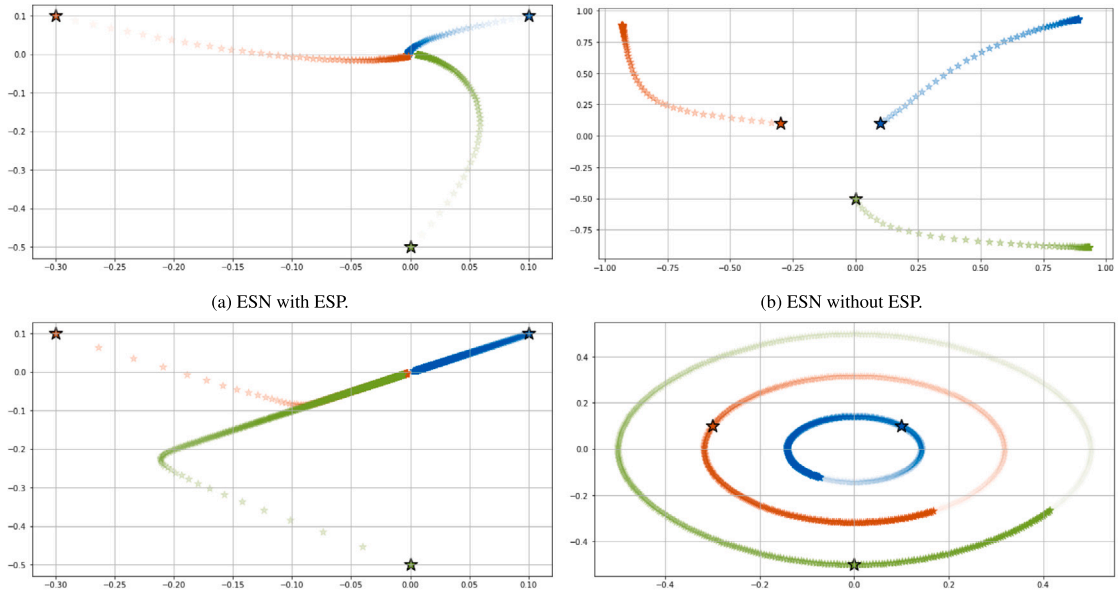
A closer inspection of Eq. (13) also reveals that the small deviations from 1 in the eigenvalues of the Jacobian in $\mathbf{J}_F(\mathbf{0}, \mathbf{0})$ are possibly due to the combined effects of the actual values of ε, γ, and of the imaginary part of the eigenvalues of $\mathbf{W}_h$. In this case, it is also possible to derive a simple and direct expression for the effective spectral radius of the autonomous linearized EuSN system, as given by the following Corollary 1.

Corollary 1. *Let us consider a EuSN whose reservoir dynamics are ruled by the state transition function in Eq. (7), with step size ε, diffusion coefficient γ, and reservoir weight matrix $\mathbf{W}_h$. Assume autonomous (i.e., with null driving input and bias) and linearized dynamics around the origin. The spectral radius of the resulting Jacobian is then given by the following expression:*

$$\rho(\mathbf{J}_F(\mathbf{0}, \mathbf{0})) = \sqrt{1 + \varepsilon^2\gamma^2 - 2\varepsilon\gamma + \varepsilon^2\rho(\mathbf{W}_h)^2}. \tag{14}$$

[5] The different initial conditions can be interpreted as the same reservoir state perturbed by different external input values.

[6] The Hausdorff distance between two non-empty sets A and B measures how far the points of A are from the points of B. In formulas, $H(A, B) = \max\{sup_{a\in A} d(a, B), sup_{b\in B} d(A, b)\}$. In the context of dynamical systems, this measure is particularly useful to measure how the system's behavior evolves under different initial conditions, providing a robust way of quantifying the possible differences among state trajectories.



(a) ESN with ESP. (b) ESN without ESP.

(c) R-ESN. (d) EuSN.

Fig. 1. Autonomous reservoir dynamics around the origin visualized for instances of: (a) ESN with ESP, (b) ESN without ESP, (c) R-ESN, and (d) EuSN. The same three initial conditions are used in all the cases, illustrated as full stars of different colors: $\mathbf{u}_0 = [0.1, 0.1]^T$ in blue, $\mathbf{v}_0 = [-0.3, 0.1]^T$ in red, and $\mathbf{z}_0 = [0, -0.5]^T$ in green. Trajectories in the 2-dimensional reservoir space are shown by (sampled) points of different shapes, where more transparent colors indicate earlier time steps. The recurrent weight matrix is: $\mathbf{W}_h = \left[[0.7, 0.1]^T, [-0.1, 0.7]^T\right]$ in (a); $\mathbf{W}_h = \left[[1.7, 0.1]^T, [-0.1, 1.7]^T\right]$ in (b); $\mathbf{W}_h = \left[[0, 0.95]^T, [0.95, 0]^T\right]$ in (c); $\mathbf{W}_h = \left[[0, 1.5]^T, [-1.5, 0]^T\right]$ in (d). In (a), (b), and (c) the leaking rate is $\alpha = 0.001$. In (d) the step size and the diffusion coefficient are $\epsilon, \gamma = 0.001$.

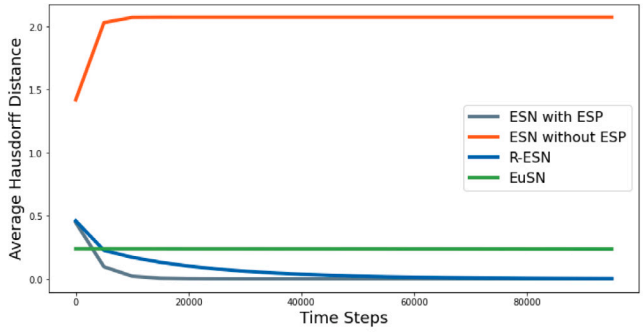


Fig. 2. Average Hausdorff distance between the states generated by the autonomous reservoir for instances of ESN with ESP, ESN without ESP, R-ESN, and EuSN. The initial conditions and the reservoir parameters are the same as Fig. 1. The Hausdorff distance is computed between pairs of consecutive 5000-long portions of the state trajectories generated from running the reservoir for 100000 time steps from the different initial conditions. Each line indicates the evolution of the average distance between the orbits starting from the different initial conditions.

Proof. From Proposition 1, we know that the eigenvalues of $\mathbf{J}_F(\mathbf{0}, \mathbf{0})$ with the largest modulus are $1 - \varepsilon\,\gamma \pm i\,\varepsilon\,\rho(\mathbf{W}_h)$. From this observation, the spectral radius of the Jacobian is easily computed, as follows:

$$\begin{aligned} \rho(\mathbf{J}_F(\mathbf{0}, \mathbf{0})) &= \sqrt{(1 + \varepsilon\,\gamma)^2 + (\varepsilon\,\rho(\mathbf{W}_h))^2} \\ &= \sqrt{1 + \varepsilon^2\gamma^2 - 2\varepsilon\gamma + \varepsilon^2\rho(\mathbf{W}_h)^2}. \quad \square \end{aligned} \tag{15}$$

Interestingly, applying the result of Corollary 1 in correspondence of small values of both ε and γ, we can easily observe that $\rho(\mathbf{J}_F(\mathbf{0}, \mathbf{0})) \approx 1$ by design. In particular, Corollary 1 indicates the marginal role of the spectral radius of the recurrent weight matrix $\mathbf{W}_h$ in determining the effective spectral radius of the EuSN, hence the asymptotic behavior of the system. The influence of $\rho(\mathbf{W}_h)$ on $\rho(\mathbf{J}_F(\mathbf{0}, \mathbf{0}))$ is indeed progressively more negligible for smaller values of the step size parameter ε. This aspect represents another crucial difference with respect to ESNs and gives support to the initialization strategy proposed in Section 3, which does not require scaling the spectral radius of $\mathbf{W}_h$, and rather suggests a simpler scaling by an ω_r hyper-parameter after the random initialization.

Fig. 3 provides some examples of values of the effective spectral radius of EuSN, computed considering reservoirs with $N = 100$ neurons, varying ε and γ between 10^{-10} and 10^{-1}, and for different values of ω_r varying from 0.001 to 1 (with resulting values of $\rho(\mathbf{W}_h)$ ranging from ≈ 0.015 to ≈ 15). For every configuration, the reported values are computed by averaging over 3 repetitions, following the reservoir initialization process described in Section 3. Results confirm that the effective spectral radius is restricted to values ≈ 1 in all the configurations, with a few exceptions, represented by the largest values of ε and ω_r.

We now extend our analysis to conditions that include non-zero input signals. To this end, we find it useful to consider the spectrum of (pseudo) local Lyapunov exponents (LLEs) [42,43] of the reservoir. An N-dimensional system has N LLEs, denoted as $LLE_1, \ldots, LLE_N$. These quantities provide a measure of the sensitivity of the system to small perturbations of the state trajectories, characterizing the dynamical behavior locally to the state evolved over time. In particular, the maximum local Lyapunov exponent (MLLE) is a useful indicator of the regime of the reservoir dynamics [7,44,45]: values > 0 indicate unstable dynamics, values < 0 denote stable dynamics, while the condition of $MLLE = 0$ corresponds to a *critical* condition of transition between stability and instability, often referred to as the *edge of stability* (or the edge of chaos). In the RC literature, it has been observed that reservoir systems operating at the edge of stability develop richer dynamics and often result in higher performance in applications [46–48]. For the case of EuSN, we can find interesting bounds for all the LLEs (hence also for the MLLE), as stated by the following Proposition 2.

Proposition 2. *Let us consider a EuSN whose reservoir dynamics are ruled by the state transition function in Eq. (7), with step size ϵ, diffusion coefficient γ, and reservoir weight matrix $\mathbf{W}_h$. The LLEs of the reservoir satisfy the following condition:*

$$\begin{aligned} &\text{for } k = 1, \ldots, N: \\ &LLE_k \in [\ln(1 - \varepsilon(\rho(\mathbf{W}_h) + \gamma)), \ln(1 + \varepsilon(\rho(\mathbf{W}_h) + \gamma))]. \end{aligned} \tag{16}$$

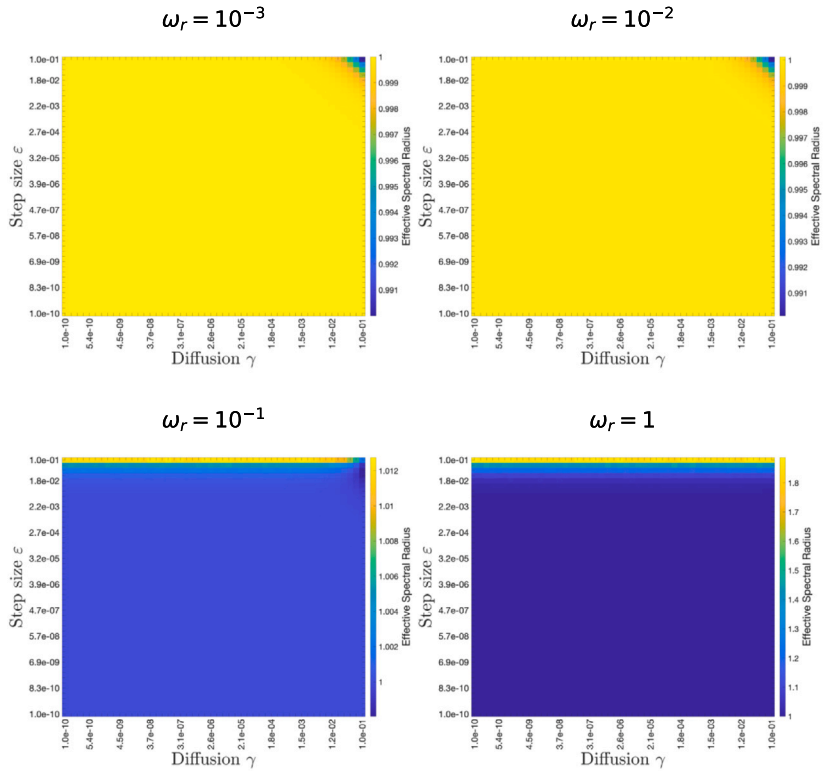


Fig. 3. Effective spectral radius of EuSN with $N = 100$ reservoir neurons, for varying values of the step size (ε, vertical axis in each plot), of the diffusion (γ, horizontal axis in each plot), and of scaling of the recurrent weight matrix $\mathbf{W}_h$ (ω_r, different plots). For each configuration, the results are averaged over 3 repetitions.

Proof. Let us suppose to drive the reservoir dynamics by an external time-series of length T, i.e., $\mathbf{x}(1), \dots \mathbf{x}(T)$. To evaluate the LLEs, we consider the dynamics along the resulting state trajectory and apply the following estimator [45], for each $k = 1, \dots, N$:

$$LLE_k = \frac{1}{T} \sum_{t=1}^{T} \ln |\lambda_k(\mathbf{J}_F(\mathbf{h}(t-1), \mathbf{x}(t)))|. \tag{17}$$

Computing the full Jacobian in Eq. (9), we get:

$$\mathbf{J}_F(\mathbf{h}(t-1), \mathbf{x}(t)) = \mathbf{I} + \varepsilon \mathbf{D}(t)\mathbf{W}_h - \varepsilon\gamma\mathbf{D}(t), \tag{18}$$

where $\mathbf{D}(t)$ is a diagonal matrix with diagonal values given by $1 - \tilde{\mathbf{h}}_1^2(t)$, $1 - \tilde{\mathbf{h}}_2^2(t)$, …, $1 - \tilde{\mathbf{h}}_N^2(t)$, with $\tilde{\mathbf{h}}_i(t)$ denoting the ith component of the vector $\tanh\Big((\mathbf{W}_h - \gamma\mathbf{I})\mathbf{h}(t-1) + \mathbf{W}_x\mathbf{x}(t) + \mathbf{b}\Big)$.

By virtue of the Bauer–Fike's theorem [49], we can observe that the eigenvalues of $\mathbf{J}_F(\mathbf{h}(t-1), \mathbf{x}(t))$ satisfy the following inequality:

$$\begin{aligned} &\text{for } k = 1, \dots, N: \\ &|\lambda_k(\mathbf{J}_F(\mathbf{h}(t-1), \mathbf{x}(t))) - 1| \quad \leq \|\varepsilon\mathbf{D}(t)\mathbf{W}_h - \varepsilon\gamma\mathbf{D}(t)\|_2 \\ &\qquad\qquad\qquad\qquad\quad = \varepsilon\|\mathbf{D}(t)\mathbf{W}_h - \gamma\mathbf{D}(t)\|_2 \\ &\qquad\qquad\qquad\qquad\quad \leq \varepsilon\|\mathbf{W}_h\|_2 + \varepsilon\gamma \\ &\qquad\qquad\qquad\qquad\quad = \varepsilon(\rho(\mathbf{W}_h) + \gamma), \end{aligned} \tag{19}$$

where we have used that $\mathbf{D}(t)$ is a diagonal matrix with non-zero values in $[0, 1]$, and $\|\mathbf{W}_h\|_2 = \rho(\mathbf{W}_h)$ for antisymmetric matrices. Eq. (19) indicates that every eigenvalue of the Jacobian is bounded within a ball of radius $\varepsilon(\rho(\mathbf{W}_h) + \gamma)$ around 1, and its modulus is thereby bounded as follows:

$$\begin{aligned} &\text{for } k = 1, \dots, N: \\ &1 - \varepsilon(\rho(\mathbf{W}_h) + \gamma) \leq |\lambda_k(\mathbf{J}_F(\mathbf{h}(t-1), \mathbf{x}(t)))| \leq 1 + \varepsilon(\rho(\mathbf{W}_h) + \gamma). \end{aligned} \tag{20}$$

Finally, using Eq. (17) and the result from Eq. (20), we can conclude that the LLEs of the reservoir satisfy the following condition:

$$\begin{aligned} &\text{for } k = 1, \dots, N: \\ &\ln\big(1 - \varepsilon(\rho(\mathbf{W}_h) + \gamma)\big) \leq LLE_k \leq \ln\big(1 + \varepsilon(\rho(\mathbf{W}_h) + \gamma)\big), \end{aligned} \tag{21}$$

which gives the thesis. □

The result of Proposition 2 allows us to bound the values of the LLEs of the reservoir based on ε, γ and $\rho(\mathbf{W}_h)$. Interestingly, as in practice both ε and γ are small positive scalars, the role of $\rho(\mathbf{W}_h)$ is negligible, and the result is that LLEs are all confined to a small neighborhood of 0 due to the peculiar architectural design of the reservoir. In particular, $MLLE \approx 0$, meaning that the EuSN model is architecturally biased towards the critical dynamical regime near the edge of stability.

To concretely illustrate the result of Proposition 2, we numerically compute the LLEs of reservoir dynamics in EuSN with $N = 100$, using the estimator in Eq. (17), and considering as driving input a one-dimensional time-series of length 500 whose elements are randomly drawn from a uniform distribution over $[-0.5, 0.5]$, with $\omega_x = \omega_b = 1$. Varying the ε, γ and ω_r hyper-parameters as in Fig. 3, in Fig. 4 we show the resulting MLLE (averaged over 3 repetitions for every configuration). As results clearly show, the obtained MLLE is ≈ 0 in every configuration, confirming the analytical outcomes regarding the intrinsic critical regime of EuSN dynamics.

5. Experiments

The mathematical analysis provided in Section 4 pointed out that EuSN dynamics are naturally biased towards the edge of stability. In this section, we show the relevant impact that such a characterization can have in applications. First, in Section 5.1, we analyze the impact on tasks involving long-term memorization in comparison to RC models. Then, in Section 5.2 we analyze the resulting performance (in terms of accuracy, required times, and energy consumption) on a large pool of time-series classification benchmarks, in comparison to both RC models and trainable RNN architectures. Finally, even though the proposed EuSN approach is introduced in the context of long-range propagation of temporal information, in Section 5.3 we analyze its performance on (next-step) time-series modeling, comparatively to the standard ESN.

The experiments described in this section were run in single-GPU mode on a system equipped with 2 × 20 Intel(R) Xeon(R) CPU E5-2698

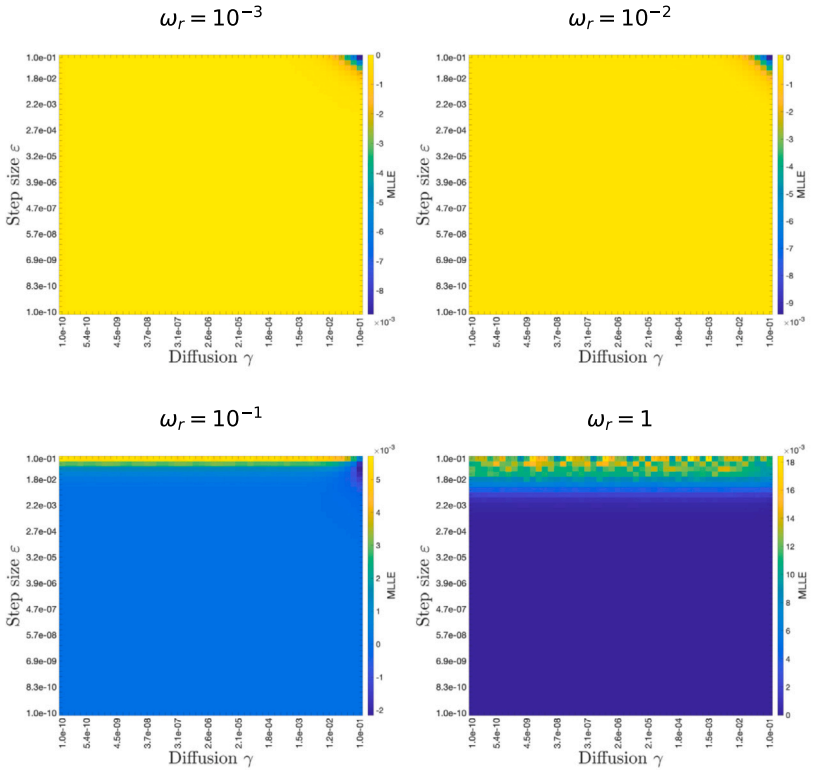


Fig. 4. MLLE of EuSN with $N = 100$ reservoir neurons, for varying values of the step size (ε, vertical axis in each plot), of the diffusion (γ, horizontal axis in each plot), and of the scaling of the recurrent weight matrix $\mathbf{W}_h$ (ω_r, different plots). For each configuration, the results are averaged over 3 repetitions. See details in the text.

v4 @ 2.20 GHz and 4 T P100-PCIE-16 GB GPU. The code used for our experiments is written in Keras and Scikit-learn, and is made publicly available online.[7]

5.1. Long-term memorization

We introduce a family of time-series classification tasks to assess the long-term memorization (LTM) capabilities of recurrent layers. The general aim is to exercise the ability of neural networks to correctly classify an increasingly long input time-series based on the presence of specific patterns injected into the sequence at arbitrary points in time. The recurrent layer must be able to effectively latch input information into the state representations over long time spans, as the information relevant to the target becomes increasingly distant from the input suffix, posing a relevant challenge for RC-based systems. [25].

We designed LTM experiments with 4 tasks. The first one is based on a *Synthetic* dataset. Specifically, we created two sequential patterns, p_1 and p_2, of 10 time-steps each, and both containing 1-dimensional signals randomly sampled from a normal distribution. Each time-series for the task was obtained by first generating a prefix with 1-dimensional normal noise of random length, uniformly drawn between 0 a 20. Then, one of two patterns (either p_1 or p_2) was added to the prefix, and then a suffix with normal noise was added until a total length of T was reached, varying T from 30 to 400. Half of the input time-series contain the first pattern p_1, and they are assigned the target class +1. The remaining time-series contain the second pattern p_2, and they are assigned the target class 0. For every choice of the total length of the padded sequences T, we generated a dataset of 1000 time-series, equally divided (with stratification) into training set and test set. Data intended for training have been divided into a training and a validation set by a further (stratified) 50%–50% split. The other tasks were obtained similarly, as padded versions of *LSST* [50], *PenDigits* [51], and *RacketSports*, a selection of classification benchmarks from the UEA & UCR time-series classification repository [52], featured by short input time-series. In these cases, the original time-series from the respective datasets were used as injected patterns into longer sequences constructed as for the Synthetic task, with the difference that the random prefix and suffix had the same dimensionality as the original input signals. Each time series constructed in this way was associated with the same target class of the corresponding sample in the original dataset. For each task, we used the original training-test splitting, with a further 67%−33% training–validation stratified splitting. Finally, notice that for all the tasks, the difficulty in the classification is increased by design for larger values of the total length of padded sequences (i.e., T), since the input pattern relevant to the target is positioned in a progressively more distant past.

Table 1 summarizes the main characteristics of the datasets used in the LTM experiments.

5.1.1. Experimental settings

We initialized EuSNs with a number of N reservoir neurons chosen from a discrete uniform distribution between 10 and 200 (with step 10), exploring values of the other network hyper-parameters in the following ranges: ω_r, ω_x and ω_b from a uniform distribution over $\{10^j : j = -3, \dots, 1\}$; ε and γ from a uniform distribution over $\{10^j : j = -5, \dots, 0\}$.

For comparison, we ran experiments with ESNs and R-ESNs considering a number of N reservoir neurons chosen in the same way as for EuSNs, i.e., from a discrete uniform distribution between 10 and 200 (with step 10). Moreover, we explored values of the hyper-parameters $\rho(\mathbf{W}_h)$ from a uniform distribution over $\{0.1, 0.2, \dots, 1.5\}$, ω_x and ω_b from a uniform distribution over $\{10^j : j = -3, \dots, 1\}$, and α from a uniform distribution over in $\{10^j : j = -5, \dots, 0\}$. Regarding $\mathbf{W}_h$ initialization, for ESNs we used the fast initialization strategy described in [24], while for R-ESN all the non-zero elements in $\mathbf{W}_h$ shared the same weight value, as in [33].

[7] https://github.com/gallicch/EulerStateNetworks

Table 1
Information on the time-series classification tasks used for the experiments on LTM, namely: size of the training (#Seq Tr) and of the test set (#Seq Ts); length of the pattern relevant to the target classification, i.e., length of the original time-series for LSST, RacketSports, and PenDigits (Pattern Length); min–max total length of the padded time-series T (Padded Length); number of input features, i.e., input dimensionality X (Feat.); number of class labels (Classes).

Name	#Seq Tr	#Seq Ts	Pattern length	Padded length	Feat.	Classes
Synthetic	500	500	10	30–400	1	2
LSST	2459	2466	36	60–400	6	14
PenDigits	7494	3498	8	30–400	2	10
RacketSports	151	152	30	50–400	6	4

In all the cases (i.e., for EuSN, ESN and R-ESN), the readout was implemented by a dense layer fed by the last state computed by the reservoir for each time-series, and trained in closed-form by ridge regression.[8]

The hyper-parameters were fine-tuned by a rigorous model selection on the validation set using random search with 200 iterations, individually for each model. After model selection, for each model, we generated 10 instances (random guesses) of the network with the selected configuration. The 10 instances were trained on the training set and evaluated on the test set. Dealing with classification tasks, we used accuracy as the performance metric reported in our results (the higher the better). The reported results have been finally achieved by computing averages and standard deviations over the 10 instances.

5.1.2. Results

The results on the LTM tasks are reported in Fig. 5, which shows the test set accuracy achieved by EuSN, ESN and R-ESN, for increasing values of the total length of padded sequences. Each plot in Fig. 5 corresponds to one of the LTM tasks considered in this study. As it is evident, the performance shown by EuSN is consistently better than that of ESN and R-ESN. In general, it can be observed that the level of accuracy obtained by all models decreases as T increases. Crucially, the accuracy of EuSN decreases slowly and remains high even for high values of T, indicating an effective propagation process of task-relevant information even after hundreds of time-steps. In contrast, consistent with its fading memory characterization, ESN (and R-ESN) shows a rapidly degrading performance. For example, in the case of the Synthetic task, ESN's accuracy is close to the chance level (i.e., ≈ 0.5) after a few hundred time-steps.

Overall, results in Fig. 5 indicate that EuSN can propagate the input information effectively in tasks requiring long-term memorization abilities, overcoming the fading memory limitations that are typical of ESN variants.

5.2. Time-series classification

We assessed the performance of EuSN on several real-world time-series classification benchmarks of diverse nature. The first 17 datasets were taken from the UEA & UCR time-series classification repository [52], namely: *Adiac* [53], *Blink* [54], *CharacterTrajectories* [55], *ECG5000* [56], *Epilepsy* [57], *FordA*, *HandOutlines* [58], *Hearthbeat* [56], *Libras* [59], *Mallat* [60], *MotionSenseHAR* [61] *ShapesAll* [62], *SpokenArabicDigits* [63], *Trace*, *UWaveGestureLibraryAll* [64], *Wafer* [65], and *Yoga*. Moreover, we have considered the *IMDB* movie review sentiment classification dataset [66], and the *Reuters* newswire classification dataset from UCI [67]. These two datasets were utilized in the parsed version that is made publicly available online.[9] In addition, individually for the two tasks, we have applied a pre-processing phase to represent each sentence by a time-series of 32-dimensional word embeddings.[10] Finally, we performed experiments with a pixel-by-pixel (i.e., sequential) version of the *MNIST* dataset [68], in which each 28×28 pixels image is reshaped into a 784-long time-series of gray levels, rescaled to the $0-1$ range.

For all datasets, we have used the original data splitting into training and test, with a further stratified splitting of the original training data into training (67%) and validation (33%) sets. A summary of the relevant information for each dataset is given in Table 2.

5.2.1. Experimental settings

We have run experiments with our introduced EuSNs, as well as with the ESNs and R-ESNs baselines, on the above-mentioned time-series classification tasks following the same base experimental setting already introduced in Section 5.1.1 for the LTM tasks. In addition, to provide even more complete experimental validation within the RC landscape, we also considered experiments on these benchmarks with DeepESN [69], a deep variant of the standard ESN model. In this latter case, the reservoir is organized into a number of layers chosen from a discrete uniform distribution between 2 and 5, and the total number of N reservoir neurons (chosen as in Section 5.1.1) were divided evenly among the layers.[11] As in [69], the readout of the DeepESN is fed by the concatenation of the states computed in each layer of the reservoir.

Moreover, we extended the extent of the comparative analysis to a pool of fully trainable neural models for time-series, including (vanilla) RNN, A-RNN [34], Gated Recurrent Unit (GRU) [70], and 1-D Convolutional Neural Network (1D-CNN). For RNN, A-RNN, and GRU, the model architecture includes a recurrent layer of the correspondent type, followed by an output layer. For 1D-CNN, the architecture includes a linear 1D convolutional layer, followed by a batch normalization layer, a ReLu nonlinearity, a global 1D average pooling layer, and finally an output layer. For all of these fully trainable models, the output (readout) layer was implemented by a dense layer with 1 sigmoid unit for the binary classification tasks, and a number of softmax units equal to the number of target classes for multi-classification tasks. Analogously, we used binary or categorical cross-entropy as a loss function during training, depending on the number of target classes. To train these models we used Adam [71] with a maximum number of 5000 epochs, learning rate chosen from a uniform distribution over $\{10^j : j = -5, \ldots, -1\}$, batch size chosen from a uniform distribution over $\{2^j : j = 5, \ldots, 8\}$ and early stopping with patience 50. In line with the experiments on the RC models, the total number of internal

[8] For all the RC models, the reservoir part was implemented in Keras as a custom RNN layer, while the readout component used Scikit-learn RidgeClassifier with the default regularization strength value of 1.

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

[10] Each sentence was represented by a sequence of words among the 10000 most frequent words in the database, with truncation to the maximum length of 200. Word embeddings have been obtained by training an MLP architecture with a preliminary embedding layer with 32 units, a hidden layer containing 128 units with ReLU activation, and a final dense output layer (with 1 sigmoidal unit for IMDB, and 46 softmax units for Reuters). This architecture was trained on the training set with RMSProp for 100 epochs, using early stopping with patience 10 on a validation set containing the 33% of the original training data. The output of the embedding layer on the sentences in the dataset is then used as the input feature for our experiments.

[11] In cases where the total number of units in the reservoir was not divisible by the number of layers, the excess units were added to the first layer.

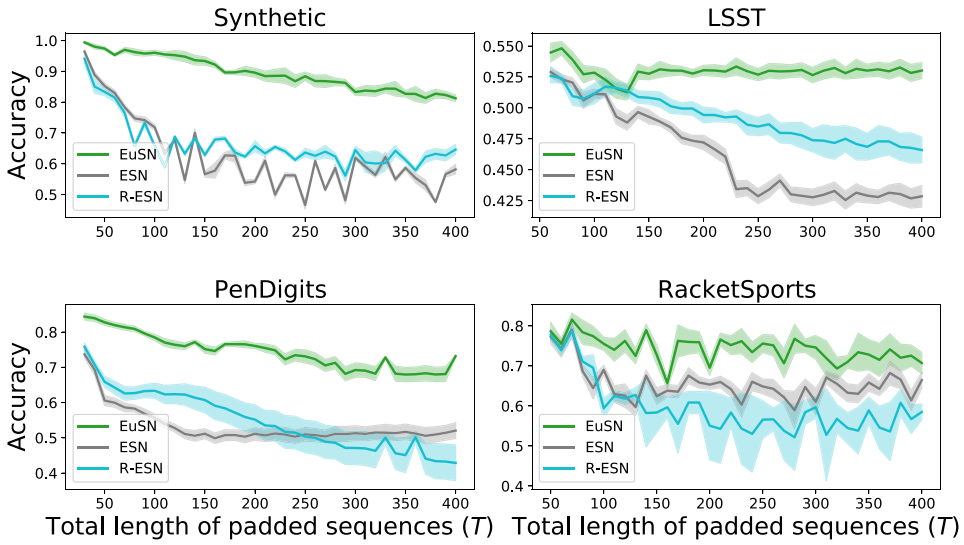


Fig. 5. Accuracy achieved on the LTM tasks by EuSN for increasing the length of the total length of padded sequences T (higher is better). Results are compared to standard ESN and R-ESN. For each value of T, the plots report the accuracy values averaged over the 10 random instances and the corresponding standard deviation as a shaded area.

Table 2
Information on the time-series classification benchmarks used in the paper, namely: size of the training (#Seq Tr) and of the test set (#Seq Ts), length of the time-series (Length), number of input features, i.e., input dimensionality X (Feat.), and number of class labels (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
ECG5000	500	4500	140	1	5
Epilepsy	137	138	206	3	4
FordA	3601	1320	500	1	2
HandOutlines	1000	370	2709	1	2
Heartbeat	204	205	405	61	2
IMDB	25 000	25 000	200	32	2
Libras	180	180	45	2	15
Mallat	55	2345	1024	1	8
MNIST	60 000	10 000	784	1	10
MotionSenseHAR	966	265	1000	12	6
Reuters	8982	2246	200	32	46
ShapesAll	600	600	512	1	60
SpokenArabicDigits	6599	2199	93	13	10
Trace	100	100	275	1	4
UWaveGestureLibraryAll	896	3582	945	1	8
Wafer	1000	6164	152	1	2
Yoga	300	3000	426	1	2

neurons N for the fully trainable models was chosen from a discrete uniform distribution between 10 and 200, with step 10.

Similarly to the experimental setup reported in Section 5.1.1, for these experiments, we performed model selection on the validation set individually for each model. For this, we used random search up to 200 iterations or 10 hours of computation time. For each task, after model selection, the performance of each model (in the selected configuration) was assessed through 10 random instances (i.e., guesses), trained on the training set, and evaluated on the test set.

Given the classification nature of the tasks under consideration, we have used accuracy as the performance metric (the higher the better). In addition to the accuracy results, for every task and every model we measured the required computational times. Moreover, to evaluate the energy footprint, we measured the energy consumed for running all the experiments, integrating the power draw of the GPU (measured in W), sampled every 10 s using the NVIDIA System Management Interface.

5.2.2. Results

For the sake of the cleanliness of the presentation of the results, here we report a graphical illustration of the main results of the performed experiments. The reader can find full details on the obtained results reported in Appendix A. Fig. 6 shows the test set accuracy achieved by the considered models on the time-series classification benchmarks. Figs. 7, and 8, which follow a similar trend with each other, respectively illustrate the times needed for model selection (in minutes), and the energy consumption required for running the experiments (in kWh). In all three figures, for simplicity's sake, we report a single value for the baseline RC models (i.e., ESN, R-ESN, and DeepESN), and a single value for the fully trainable models (i.e., RNN, A-RNN, GRU, and 1D-CNN). The reported value for each group is the one corresponding to the model that, among those in the group, is found to have the highest accuracy performance. Please refer to Appendix A for a detailed report on the results obtained individually for all the RC baselines and all the fully trainable models.

Two major observations can be drawn from the results. First, it is evident from Fig. 6 that EuSN in general considerably outperforms the RC baselines in terms of accuracy. On the one hand, in tasks where a clear performance gap in favor of fully trained models over standard RC models is appreciable (e.g., Adiac, HandOutlines, Libras, Mallat, MNIST, MotionSenseHAR, Reuters, ShapesAll, Trace), EuSNs

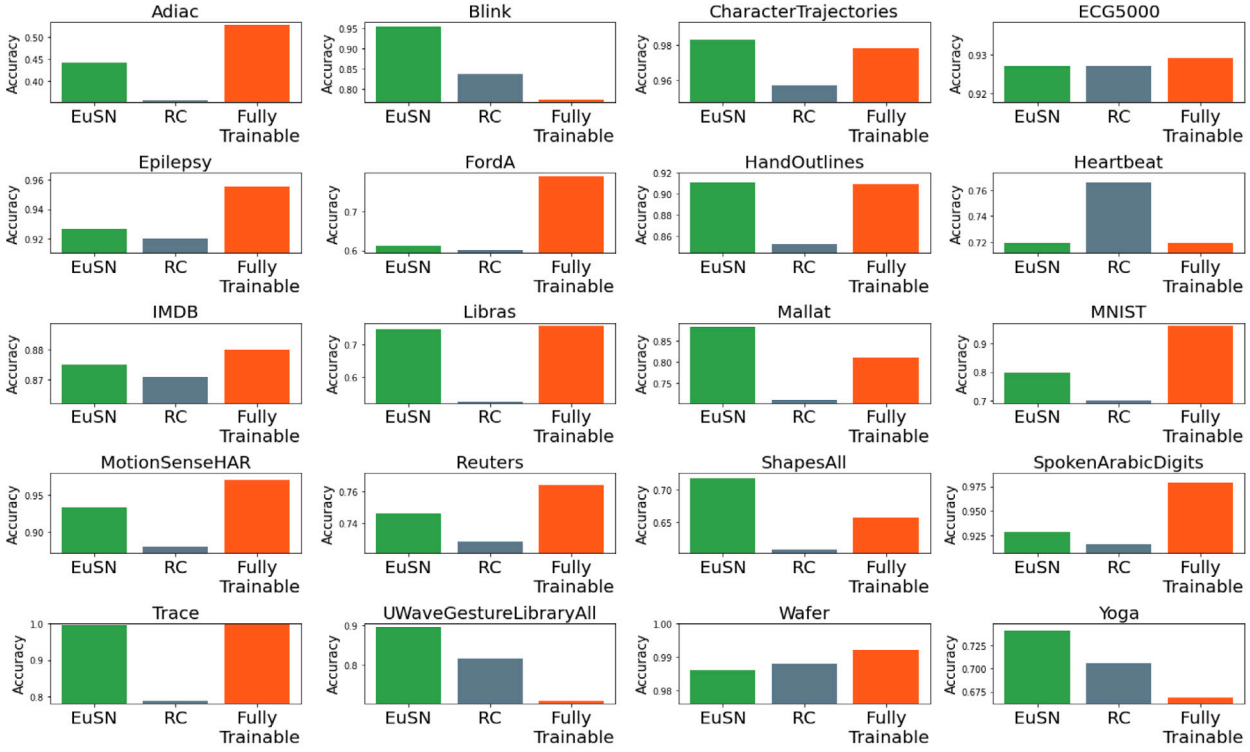


Fig. 6. Averaged test set accuracy on the time-series classification benchmarks (higher is better). The "RC" results refer to the reservoir-based approach (among ESN, R-ESN, and DeepESN) that achieves the highest accuracy on each task. The "Fully Trainable" results refer to the trainable model (among RNN, A-RNN, GRU, and 1D-CNN) that achieves the highest accuracy on each task. Further details on the numerical results can be found in Appendix A.

allow to largely bridge this gap, in some cases even exceeding the performance of fully trained networks. On the other hand, in tasks where standard RC models perform better than fully trained ones (Blink, UWaveGestureLibraryAll, Yoga), EuSNs generally allow an even higher performance. The only two exceptions are the Heartbeat and Wafer tasks, where the performance of the standard RC models is better than that of EuSNs. In these cases, however, we can see that EuSNs reach or even exceed the performance of the fully trained models.

Second, looking at Figs. 7 and 8, we can see that by leveraging untrained dynamics, EuSN proved to be at least as efficient as RC baselines, and far more efficient than fully trainable networks. Overall, compared to fully trainable models, while showing a competitively high accuracy, EuSN allowed a dramatic advantage in terms of required computational resources, enabling an average of $\approx$ 218x and up to an $\approx$ 464x reduction in computational times, and an average of $\approx$ 389x and up to an $\approx$ 1750x reduction in energy consumption. In contrast, baseline RC models, while providing a similar computational advantage (although reduced due to the higher computational resources required by DeepESN in some of the tasks), do not achieve the same competitive level of classification performance.

5.3. Time-series modeling

To complete the experimental analysis of the proposed model in heterogeneous contexts, we conducted additional experiments on tasks of a potentially opposite nature to the architectural bias of the EuSN model dynamics toward long-range temporal propagation. Therefore, we considered a pool of 10 time-series modeling tasks on different temporal sequences. The datasets used for this purpose are the following: *Adiac*, *ECG5000*, *FordA*, *HandOutlines*, *Mallat*, *ShapesAll*, *Trace*, *UWaveGestureLibraryAll*, *Wafer*, and *Yoga* (the reader is referred to Table 2 for further information on the used datasets). Differently from the time-series classification setup adopted in Section 5.2, here each task consists of predicting the next time-step of each given sequence, given its previous values. In other words, for each sequence and time step t, the network is trained to predict the value of $\mathbf{x}(t+1)$, given $\mathbf{x}(1), \ldots, \mathbf{x}(t)$. Also in this case, we have used the original training/test splitting of the datasets, with an inner level of training (67%)/validation (33%) splitting applied to the original training data.

5.3.1. Experimental settings

We have run experiments on the above-defined time-series modeling tasks with both EuSN and ESN, for baseline comparison. In addition, for both models, we have extended the analysis to include standard architectural variants in the case of regression tasks with RC approaches [8,23], i.e., direct and trainable input-to-readout connections. In this case, at each time-step t, the readout is fed by $[\mathbf{h}(t); \mathbf{x}(t)]$, i.e., by the concatenation of the current reservoir state and the driving input, and it is trained to predict the next value of the input $\mathbf{x}(t+1)$. In what follows, we use EuSN_{I-R} and ESN_{I-R} to refer to the models' configurations with input-to-readout connections.

For both EuSN and ESN, we have explored reservoir configurations with 200 neurons. The rest of the hyper-parameters were explored as in the previous experiments on long-term memorization (see Section 5.1.1). Individually for each model and each dataset, the values of the hyper-parameters were fine-tuned by model selection on the validation set, using a random search with 100 iterations. After that, the selected configurations were instantiated in a number of 10 random guesses, trained on the training set, and tested on the test set. Given the regression nature of the tasks under consideration, here we have used the root mean squared error (RMSE) as a performance metric for the models (the lower the better). The reported results are obtained by computing the average (and the standard deviation) over the 10 random guesses.

5.3.2. Results

The results obtained on the time-series modeling tasks are given in Fig. 9. The reader can find full details on the numerical results

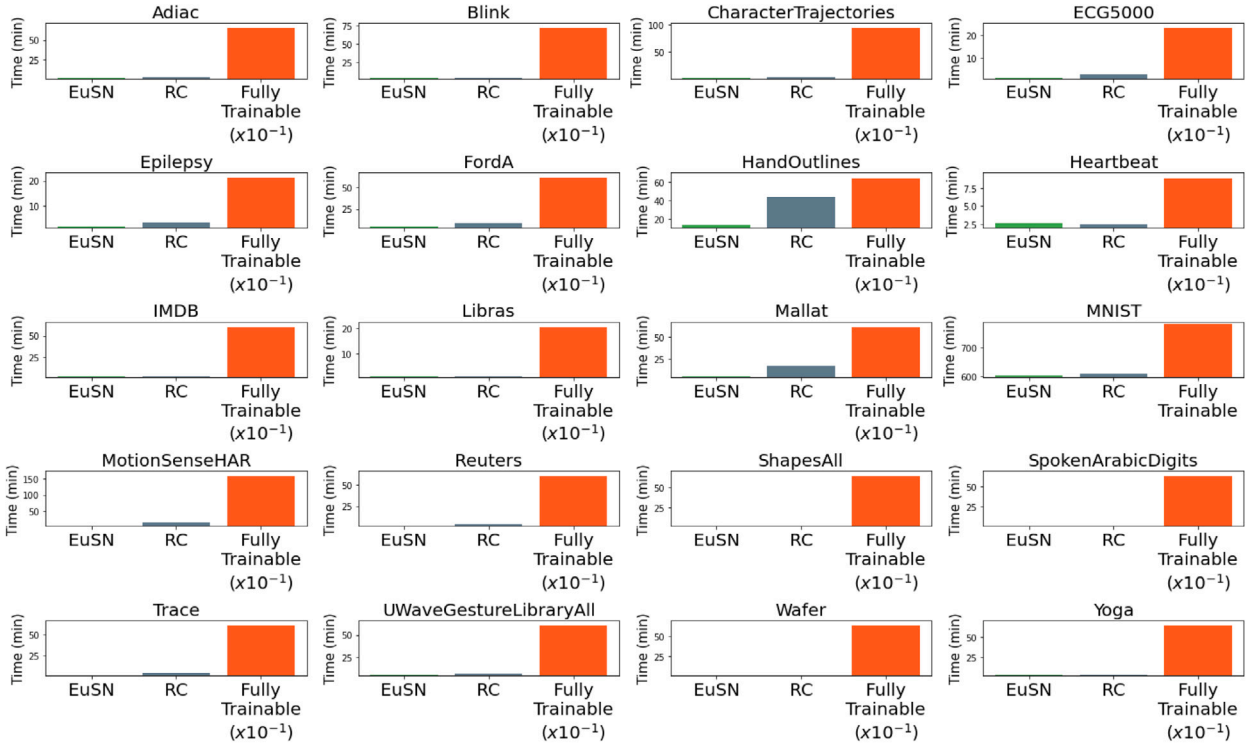


Fig. 7. Time required for model selection (in minutes) on the time-series classification benchmarks (lower is better). The "RC" results refer to the reservoir-based approach (among ESN, R-ESN, and DeepESN) that achieves the highest accuracy on each task. The "Fully Trainable" results refer to the trainable model (among RNN, A-RNN, GRU, and 1D-CNN) that achieves the highest accuracy on each task. For ease of visualization, the results of fully trainable models are shown downscaled by a factor of 10 in all tasks except MNIST. Further details on the numerical results can be found in Appendix A.

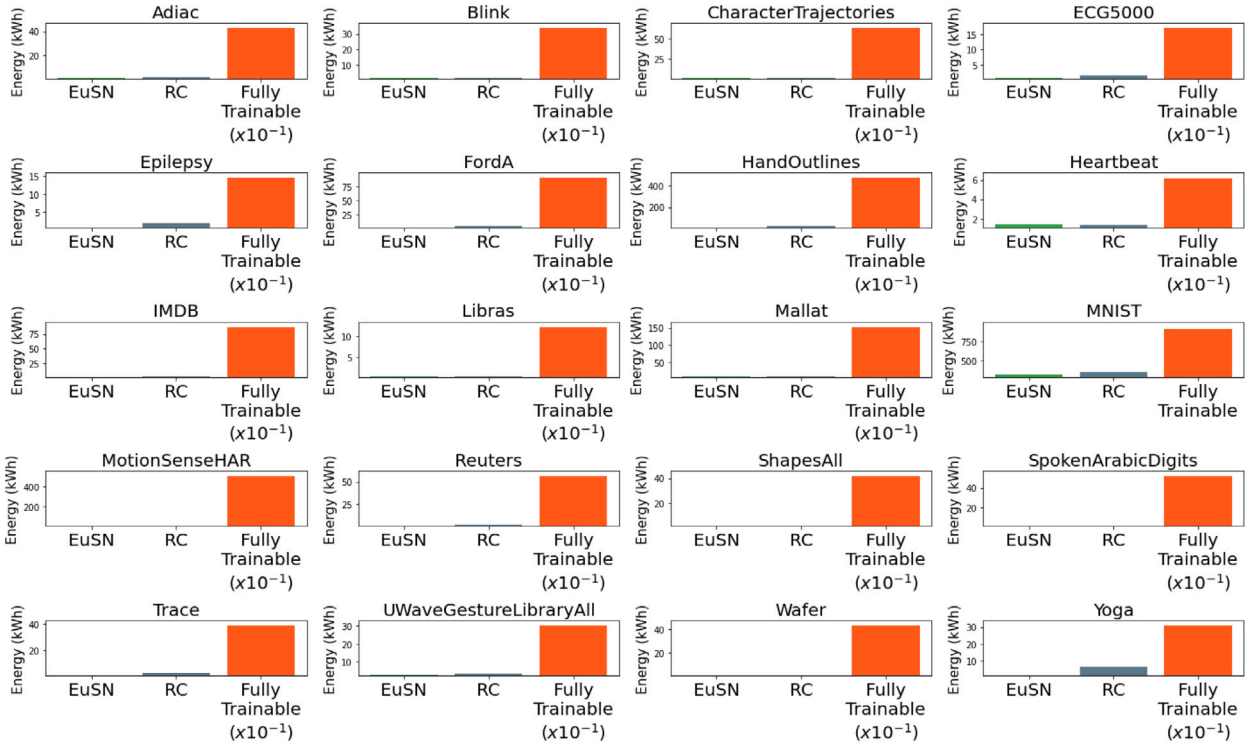


Fig. 8. Energy consumption (in kWh) required for the experiments on the time-series classification benchmarks (lower is better). The "RC" results refer to the reservoir-based approach (among ESN, R-ESN, and DeepESN) that achieves the highest accuracy on each task. The "Fully Trainable" results refer to the trainable model (among RNN, A-RNN, GRU, and 1D-CNN) that achieves the highest accuracy on each task. For ease of visualization, the results of fully trainable models are shown downscaled by a factor of 10 in all tasks except MNIST. Further details on the numerical results can be found in Appendix A.

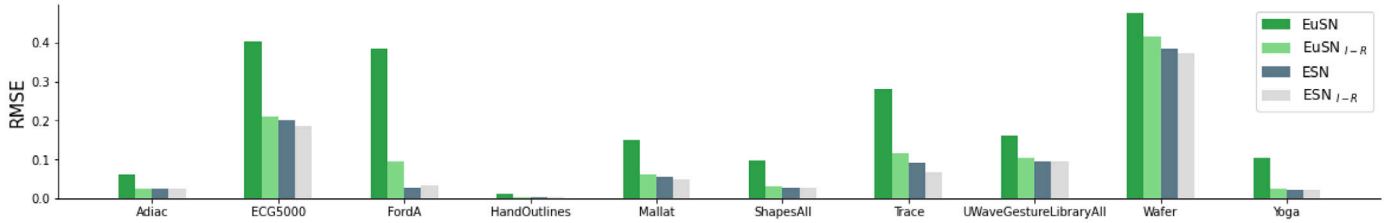


Fig. 9. Averaged test set RMSE on the time-series modeling tasks (lower is better). Further details on the numerical results can be found in Appendix B.

in Appendix B. The results indicate that the standard EuSN achieves a lower performance on these tasks compared to the ESN baseline. This observation adds an interesting insight to our analysis and is not surprising. In fact, the results confirm the lower suitability of non-dissipatively biased reservoir dynamics in next-step prediction tasks where the long-range propagation of the input information is not crucial (unlike the time-series classification tasks analyzed in Sections 5.1 and 5.2). Moreover, we can see that the inclusion of direct connections between the input and the readout in EuSN$_{I-R}$ allows to significantly reduce the error, reaching a performance level that is comparable to the ESN variants. Indeed, for ESNs, the addition of the current input information results in only a small improvement in predictive performance, since the reservoir dynamics already provide a short-range representation of the input history. Interestingly, in the case of EuSNs, since the reservoir alone provides a purely long-range representation of the past, adding the information related to the current input actually enriches the representation provided as input to the readout, and performance improves significantly. Overall, these results offer additional empirical support for the mathematical properties of the architectural bias of EuSNs as analyzed in Section 4. Furthermore, they assist in clarifying the natural effectiveness boundaries of the proposed model.

6. Conclusions

In this paper, we have introduced the Euler State Network (EuSN) model, a novel approach to design RC neural networks, based on the numerical discretization of ODEs by the forward Euler method. The proposed approach leverages the idea of constraining the recurrent reservoir weight matrix to have an antisymmetric structure, resulting in dynamics that are neither lossy nor unstable, and allowing for an effective transmission of input signals over time. Moreover, being featured by untrained recurrent neurons, EuSNs combine the ability to model long-term propagation of input signals with the efficiency typical of RC systems.

Our mathematical analysis of reservoir dynamics revealed that EuSNs are biased towards unitary effective spectral radius and zero local Lyapunov exponents, indicating an intrinsically critical dynamical regime near the edge of stability. Experiments on long-term memorization tasks have shown the effectiveness of EuSN in latching the input information into the state representations over long time spans, overcoming the inherent limitations of baseline RC models. Furthermore, through experiments on time-series classification benchmarks, we found that EuSN provides a formidable trade-off between accuracy and efficiency, compared to state-of-the-art neural models. While reaching – or even outperforming – the highest level of accuracy achieved by the fully trainable models, the proposed EuSN allows a tremendous reduction in computation time and energy consumption. While standard RC approaches can provide similar computational gains, the accuracy levels these achieve are generally considerably lower. Finally, our time-series modeling experiments have completed the spectrum of experimental analyses of the model, contributing to critically delineating the contours of its practical applicability and confirming mathematical intuitions about the long-range nature of the representations developed by the reservoir of a EuSN. These latter results revealed the inherent difficulties of non-dissipative reservoirs in dealing with next-step prediction tasks while highlighting how the addition of direct connections between input and readout can help alleviate these difficulties in tasks that go against the nature of the model's dynamics.

The work presented in this paper demonstrated the validity of an RC-based approach that relies on a profoundly different architectural bias than that of conventional ESNs. While ESNs represent dynamical systems with fading memory controlled by the spectral properties of the recurrent weight matrix, EuSNs represent dynamical systems with dynamics controllably close to the critical behavior (i.e., to the edge of stability) by construction. Consequently, while ESNs are naturally able to distinguish input sequences in a suffix-based way, EuSNs are inherently able to distinguish input sequences based on their entire temporal evolution. The investigation carried out in this article showed clear experimental evidence regarding the effectiveness of the proposed method in time-series classification tasks. However, it is important to emphasize that EuSNs are not proposed here as a replacement for ESNs. Instead, they intend to represent a new model in the RC landscape, suitable for problems involving some form of long-term memorization rather than fading memory. For instance, our findings on next-step prediction of time-series support the idea that, without the use of input-readout connections, EuSNs appear unlikely to be suitable for dealing with problems in which it is necessary to model a fast reaction of the target with respect to changes in the driving input. Investigations into the Markovian properties of the dynamics of EuSNs represent insights for future work.

Overall, the work presented in this paper is seminal and represents a first step in the study of ODE-inspired RC architectures and their theoretical properties. Although the presented results are already very encouraging, new directions for future works can be envisaged, for example, deepening the study of the differences between ESN and EuSN in terms of reservoir state space organization and the nature of the memory structure. Other examples of future studies may involve an in-depth analysis of the reservoir topology in EuSN. For example, it might be useful to investigate ways to simplify the architectural construction of the reservoir, while maintaining its computational characteristics, or to expand the diversity of generated dynamics by randomizing the diffusion term among different reservoir neurons. Future work would also deserve to explore implementations in neuromorphic hardware, a possibility at once intriguing and challenging, for example in the treatment of noise and the consequences it might have in the realization of non-dissipative physical EuSN reservoirs. Finally, we consider of great interest the extensions of the methodology proposed in this paper to graph processing. RC-based approaches have previously shown great potential in this regard, resulting in an unparalleled trade-off between application effectiveness and computational efficiency in graph classification [11,72,73] and spatio-temporal information processing [74,75]. In this context, the adoption of an architectural bias based on stable and non-dissipative dynamics could have a decisive impact on tasks that require effective propagation of information through long paths on a graph. The recent developments described in [76] already show theoretical and empirical evidence of possible benefits, and pave the way to further developments in the field of deep graph networks.

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

Model	*Avg rank*
EuSN (ours)	2.30
ESN	5.20
R-ESN	5.25
DeepESN	4.30
RNN	6.05
A-RNN	3.75
GRU	4.15
1D-CNN	5.00

CRediT authorship contribution statement

Claudio Gallicchio: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Claudio Gallicchio reports financial support was provided by the Italian Ministry of University and Research, by EU Horizon research and innovation programme, and by the EC H2020 programme.

Data availability

Data will be made available on request.

Acknowledgments

This work is partially supported by NEURONE, a project funded by the Italian Ministry of University and Research (PRIN 20229JRTZA), by EMERGE, a project funded by EU Horizon research and innovation programme (GA n. 101070918), and by TEACHING, a project funded by EC H2020 programme (GA n. 871385).

Appendix A. Detailed time-series classification results

Here, we provide the full details of the experimental outcomes on the time-series classification datasets described in Section 5.2.

First, in Table A.3 we provide a general overview of the achieved levels of accuracy, by reporting the averaged ranking across the accuracy results on all the time-series classification benchmarks considered. It can be noticed that EuSNs overall result in the best-performing model, followed by A-RNNs and GRUs. DeepESNs perform on average better than ESNs and R-ESNs. 1D-CNNs and RNNs are generally characterized by the worst results in terms of accuracy.

In Tables B.4–B.23, we report the full details of the experimental outcomes on the time-series classification datasets described in Section 5.2. For every task and for every model, the results in the tables include the following data:

- *Acc* - the test set accuracy (averaged over the 10 random guesses, and indicating the standard deviations);
- *#Par* - the number of trainable parameters of the selected configuration;
- *Time (min.)* - the time in minutes required for training and testing the selected configuration (averaged over the 10 random guesses, and indicating the standard deviations);
- *MS Time (min.)* - the time in minutes required for model selection;
- *Energy (kWH)* - the energy in kWH consumed to perform all the experiments.

Notice that, differently from the other models, the GRU implementation could exploit the optimized cuDNN code available through Keras (resulting in a demand for computational resources that is far lower than that which would have resulted from a custom implementation).

Results in Tables B.4–B.23 provide further support to the analysis reported in Section 5.2. Considering also the standard deviation over the results, we can observe that EuSNs substantially outperform standard ESNs, R-ESNs and DeepESNs in terms of predicted accuracy in almost all tasks, at the same time requiring about the same cost in terms of both time and energy consumption. Widening the comparison to fully trained models, EuSNs achieve comparable, and in some cases better, accuracy at costs that are orders of magnitude lower.

Table B.4
Results on the Adiac dataset.

Adiac					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.442 (±0.009)	7437	0.011 (±0.001)	1.834	1.028
ESN	0.159 (±0.067)	5957	0.010 (±0.001)	1.921	1.046
R-ESN	0.254 (±0.007)	7067	0.010 (±0.001)	1.888	1.037
DeepESN	0.356 (±0.036)	5957	0.023 (±0.002)	3.476	1.801
RNN	0.243 (±0.105)	43 547	22.808 (±8.195)	621.018	484.170
A-RNN	0.528 (±0.068)	35 567	9.500 (±1.193)	648.365	424.210
GRU	0.201 (±0.146)	28 477	0.341 (±0.134)	479.226	351.676
1D-CNN	0.118 (±0.057)	5197	0.666 (±0.159)	44.458	33.076

Table B.5
Results on the Blink dataset.

Blink					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.955 (±0.010)	131	0.019 (±0.001)	3.530	1.587
ESN	0.837 (±0.028)	171	0.021 (±0.002)	3.539	1.587
R-ESN	0.557 (±0.033)	131	0.020 (±0.001)	3.499	1.666
DeepESN	0.700 (±0.066)	181	0.061 (±0.005)	8.902	4.842
RNN	0.478 (±0.044)	10 601	16.352 (±8.338)	602.812	349.893
A-RNN	0.775 (±0.044)	6881	2.788 (±0.783)	707.131	334.985
GRU	0.658 (±0.037)	53 171	6.670 (±1.485)	613.643	312.688
1D-CNN	0.603 (±0.147)	1921	0.203 (±0.103)	163.240	125.428

Table B.6
Results on the CharacterTrajectories dataset.

CharacterTrajectories					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.983 (±0.002)	3620	0.011 (±0.001)	2.012	1.153
ESN	0.918 (±0.012)	3220	0.011 (±0.000)	2.000	1.069
R-ESN	0.938 (±0.008)	3620	0.011 (±0.001)	1.782	1.093
DeepESN	0.957 (±0.004)	4020	0.017 (±0.001)	3.892	1.996
RNN	0.684 (±0.152)	14 760	6.087 (±2.627)	601.495	376.926
A-RNN	0.978 (±0.003)	1640	18.828 (±1.652)	932.676	633.714
GRU	0.978 (±0.004)	92 670	0.438 (±0.153)	603.785	512.404
1D-CNN	0.977 (±0.001)	3860	1.382 (±0.154)	568.496	426.929

Table B.7
Results on the ECG5000 dataset.

ECG5000					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.927 (±0.002)	855	0.009 (±0.001)	1.287	0.666
ESN	0.912 (±0.001)	1005	0.009 (±0.001)	1.266	0.664
R-ESN	0.913 (±0.002)	755	0.008 (±0.001)	1.326	0.750
DeepESN	0.927 (±0.003)	555	0.015 (±0.001)	2.830	1.467
RNN	0.923 (±0.011)	1885	1.557 (±0.441)	588.373	343.755
A-RNN	0.920 (±0.006)	30 095	1.992 (±0.158)	630.257	373.438
GRU	0.929 (±0.003)	60 765	0.257 (±0.020)	229.285	169.291
1D-CNN	0.894 (±0.038)	1325	0.570 (±0.135)	299.279	195.337

Table B.8
Results on the Epilepsy dataset.

Epilepsy					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.927 (±0.012)	684	0.008 (±0.001)	1.527	0.752
ESN	0.901 (±0.017)	684	0.009 (±0.001)	1.557	0.842
R-ESN	0.909 (±0.009)	764	0.008 (±0.001)	1.585	0.840
DeepESN	0.920 (±0.012)	804	0.014 (±0.000)	3.548	1.791
RNN	0.560 (±0.121)	23 704	1.500 (±0.541)	215.733	130.642
A-RNN	0.933 (±0.008)	7044	1.340 (±0.255)	639.625	371.542
GRU	0.704 (±0.152)	61 464	0.171 (±0.034)	181.629	121.691
1D-CNN	0.955 (±0.004)	1924	2.432 (±0.254)	212.452	145.788

Table B.9
Results on the FordA dataset.

FordA					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.612 (±0.015)	191	0.020 (±0.002)	3.766	2.350
ESN	0.576 (±0.012)	201	0.022 (±0.001)	3.843	2.515
R-ESN	0.551 (±0.018)	151	0.022 (±0.002)	3.977	2.554
DeepESN	0.601 (±0.008)	201	0.039 (±0.004)	8.600	5.044
RNN	0.782 (±0.140)	36 671	37.616 (±31.107)	616.725	596.350
A-RNN	0.494 (±0.013)	6641	29.183 (±13.680)	1780.866	1252.646
GRU	0.717 (±0.184)	52 001	16.305 (±9.991)	447.561	512.732
1D-CNN	0.792 (±0.009)	1191	15.400 (±3.653)	610.572	908.310

Table B.10
Results on the HandOutlines dataset.

HandOutlines					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.911 (±0.004)	191	0.067 (±0.003)	13.438	8.187
ESN	0.668 (±0.041)	71	0.071 (±0.003)	13.378	8.136
R-ESN	0.690 (±0.047)	61	0.069 (±0.005)	13.418	8.180
DeepESN	0.853 (±0.027)	161	0.316 (±0.034)	44.193	24.778
RNN	0.673 (±0.024)	2651	160.575 (±85.782)	792.408	1362.865
A-RNN	0.909 (±0.005)	32 941	706.254 (±305.364)	637.913	4789.139
GRU	0.659 (±0.015)	401	5.151 (±1.338)	845.320	806.848
1D-CNN	0.620 (±0.105)	1191	6.316 (±1.179)	604.309	1170.398

Appendix B. Detailed time-series modeling results

In Table B.24 we provide the full details on the experimental outcomes of the time-series modeling tasks described in Section 5.3. The findings demonstrate that using direct connections between the input and readout layers in EuSN architecture results in a clear enhancement in EuSNs' performance for predicting the next step of time-series for the analyzed tasks. EuSN$_{I-R}$ consistently outperforms EuSN, with an

Table B.11
Results on the Heartbeat dataset.

Heartbeat					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.719 (±0.021)	171	0.013 (±0.000)	2.544	1.414
ESN	0.753 (±0.008)	121	0.013 (±0.001)	2.490	1.425
R-ESN	0.766 (±0.010)	191	0.013 (±0.001)	2.460	1.322
DeepESN	0.680 (±0.020)	81	0.041 (±0.005)	7.278	3.853
RNN	0.670 (±0.026)	48 071	9.610 (±2.240)	614.315	399.403
A-RNN	0.633 (±0.054)	16 301	11.321 (±4.801)	619.311	413.130
GRU	0.682 (±0.040)	96 001	0.419 (±0.031)	399.335	272.900
1D-CNN	0.719 (±0.040)	22 441	0.115 (±0.004)	87.854	61.588

Table B.12
Results on the IMDB dataset.

IMDB					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.875 (±0.002)	201	0.016 (±0.001)	2.087	2.001
ESN	0.871 (±0.007)	111	0.010 (±0.000)	2.127	2.637
R-ESN	0.856 (±0.020)	111	0.011 (±0.001)	2.092	2.317
DeepESN	0.870 (±0.003)	81	0.013 (±0.002)	4.393	3.451
RNN	0.872 (±0.006)	2961	207.990 (±130.754)	761.358	1613.372
A-RNN	0.863 (±0.000)	11 161	272.227 (±92.466)	783.339	1973.787
GRU	0.880 (±0.000)	93 281	38.173 (±5.213)	603.641	872.405
1D-CNN	0.878 (±0.000)	4001	1.328 (±0.054)	600.094	1197.343

Table B.13
Results on the Libras dataset.

Libras					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.746 (±0.031)	2565	0.006 (±0.001)	0.768	0.392
ESN	0.479 (±0.017)	2415	0.006 (±0.001)	0.752	0.384
R-ESN	0.526 (±0.013)	2715	0.006 (±0.001)	0.794	0.378
DeepESN	0.467 (±0.028)	1665	0.007 (±0.000)	1.145	0.537
RNN	0.597 (±0.039)	22 135	0.406 (±0.106)	341.168	195.617
A-RNN	0.721 (±0.017)	39 535	0.500 (±0.036)	600.094	344.389
GRU	0.723 (±0.175)	113 445	0.132 (±0.015)	170.894	100.965
1D-CNN	0.757 (±0.043)	2895	0.613 (±0.066)	204.841	121.053

Table B.14
Results on the Mallat dataset.

Mallat					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.885 (±0.014)	1528	0.029 (±0.001)	4.930	9.306
ESN	0.562 (±0.020)	1368	0.028 (±0.002)	4.959	8.097
R-ESN	0.642 (±0.054)	1448	0.026 (±0.001)	5.074	3.029
DeepESN	0.709 (±0.133)	808	0.106 (±0.012)	16.726	9.095
RNN	0.203 (±0.079)	27 208	9.314 (±2.879)	627.158	656.499
A-RNN	0.810 (±0.017)	11 008	4.631 (±0.716)	617.484	1525.189
GRU	0.179 (±0.032)	70 058	0.430 (±0.046)	183.117	656.499
1D-CNN	0.123 (±0.000)	1688	0.256 (±0.042)	51.276	35.952

Table B.15
Results on the MNIST dataset.

MNIST					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.800 (±0.004)	1910	3.498 (±0.038)	603.901	330.898
ESN	0.640 (±0.009)	1110	3.676 (±0.138)	600.850	329.289
R-ESN	0.665 (±0.009)	810	3.643 (±0.161)	603.340	329.365
DeepESN	0.700 (±0.017)	810	7.204 (±0.131)	609.023	364.008
RNN	0.885 (±0.153)	21 290	5485.678 (±1714.600)	7529.925	22 767.684
A-RNN	0.867 (±0.008)	7370	2178.702 (±463.826)	2486.857	15 664.276
GRU	0.961 (±0.049)	8460	79.697 (±17.154)	782.909	902.842
1D-CNN	0.519 (±0.005)	1930	189.917 (±13.900)	756.519	5661.165

average improvement of ≈ 57% and up to ≈ 81%. When compared to baseline ESNs, the inclusion of connections between input and output helps to reduce the performance gap with EuSN by an average of ≈ 87% on the tasks considered.

Table B.16
Results on the MotionSenseHAR dataset.

MotionSenseHAR					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.934 (±0.007)	1026	0.034 (±0.002)	6.153	2.884
ESN	0.875 (±0.021)	1026	0.035 (±0.005)	5.956	2.809
R-ESN	0.853 (±0.028)	1146	0.034 (±0.003)	6.339	2.961
DeepESN	0.880 (±0.021)	1206	0.072 (±0.006)	16.710	8.984
RNN	0.700 (±0.102)	16 686	14.150 (±7.962)	610.699	341.362
A-RNN	0.970 (±0.006)	19 376	71.593 (±6.250)	1569.631	5047.365
GRU	0.935 (±0.035)	129 606	0.508 (±0.066)	614.305	1527.228
1D-CNN	0.966 (±0.006)	5406	0.410 (±0.042)	336.060	425.731

Table B.17
Results on the Reuters dataset.

Reuters					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.746 (±0.002)	8326	0.011 (±0.001)	2.111	1.226
ESN	0.699 (±0.003)	7866	0.009 (±0.000)	2.215	1.624
R-ESN	0.706 (±0.003)	8786	0.012 (±0.001)	2.086	1.502
DeepESN	0.728 (±0.004)	8326	0.029 (±0.004)	4.113	2.577
RNN	0.486 (±0.095)	4806	324.440 (±230.949)	710.351	2242.565
A-RNN	0.745 (±0.003)	27 216	119.179 (±2.328)	1035.142	1343.872
GRU	0.724 (±0.047)	37 666	2.759 (±0.016)	630.515	583.792
1D-CNN	0.764 (±0.002)	24 696	1.357 (±0.015)	603.212	561.465

Table B.18
Results on the ShapesAll dataset.

ShapesAll					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.718 (±0.007)	10 860	0.016 (±0.001)	3.257	1.900
ESN	0.526 (±0.015)	10 260	0.015 (±0.001)	3.083	1.724
R-ESN	0.608 (±0.020)	10 860	0.019 (±0.002)	3.107	1.825
DeepESN	0.574 (±0.016)	9660	0.066 (±0.010)	9.391	5.078
RNN	0.182 (±0.053)	18 980	7.641 (±2.683)	600.022	386.331
A-RNN	0.656 (±0.018)	2820	9.672 (±0.709)	632.344	416.894
GRU	0.588 (±0.028)	59 730	1.072 (±0.131)	622.661	540.201
1D-CNN	0.254 (±0.069)	7980	2.236 (±0.761)	57.853	62.523

Table B.19
Results on the SpokenArabicDigits dataset.

SpokenArabicDigits					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.929 (±0.009)	1910	0.010 (±0.001)	1.605	0.927
ESN	0.916 (±0.009)	710	0.008 (±0.001)	1.634	0.964
R-ESN	0.781 (±0.085)	1810	0.011 (±0.002)	1.703	1.074
DeepESN	0.916 (±0.010)	2010	0.021 (±0.003)	2.620	1.470
RNN	0.875 (±0.053)	40 670	3.906 (±0.587)	610.520	373.028
A-RNN	0.845 (±0.003)	12 410	475.437 (±11.460)	700.678	3117.685
GRU	0.979 (±0.003)	85 610	0.392 (±0.112)	629.330	521.702
1D-CNN	0.957 (±0.004)	2610	5.641 (±1.117)	621.477	443.938

Table B.20
Results on the Trace dataset.

Trace					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.994 (±0.008)	764	0.009 (±0.000)	1.969	0.911
ESN	0.492 (±0.097)	484	0.009 (±0.001)	1.902	0.759
R-ESN	0.776 (±0.052)	724	0.009 (±0.000)	1.863	0.835
DeepESN	0.790 (±0.070)	644	0.028 (±0.000)	4.876	2.560
RNN	0.427 (±0.161)	20 444	2.878 (±1.284)	257.409	130.565
A-RNN	0.997 (±0.005)	6884	25.065 (±0.466)	603.131	390.523
GRU	0.515 (±0.301)	11 584	1.041 (±1.815)	182.036	87.252
1D-CNN	0.323 (±0.226)	1204	0.274 (±0.417)	154.188	92.464

Table B.21
Results on the UWaveGestureLibraryAll dataset.

UWaveGestureLibraryAll					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.896 (±0.002)	1448	0.034 (±0.003)	6.336	2.961
ESN	0.815 (±0.016)	1368	0.033 (±0.002)	6.751	3.188
R-ESN	0.785 (±0.006)	1128	0.033 (±0.002)	6.331	4.408
DeepESN	0.788 (±0.042)	808	0.096 (±0.012)	16.910	9.171
RNN	0.379 (±0.075)	9008	25.029 (±9.022)	623.929	399.263
A-RNN	0.488 (±0.008)	208	211.754 (±4.353)	766.737	1313.035
GRU	0.710 (±0.306)	111 538	5.681 (±4.757)	601.381	299.809
1D-CNN	0.498 (±0.061)	1688	3.142 (±0.363)	391.341	391.272

Table B.22
Results on the Wafer dataset.

Wafer					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.986 (±0.002)	101	0.009 (±0.001)	1.705	0.870
ESN	0.988 (±0.002)	201	0.011 (±0.000)	1.756	1.048
R-ESN	0.960 (±0.009)	61	0.009 (±0.001)	1.759	0.954
DeepESN	0.984 (±0.003)	151	0.027 (±0.002)	3.281	1.727
RNN	0.965 (±0.032)	26 081	9.288 (±4.885)	601.445	397.676
A-RNN	0.992 (±0.001)	10 301	11.337 (±2.300)	636.482	430.376
GRU	0.981 (±0.012)	1401	0.739 (±0.339)	318.245	267.189
1D-CNN	0.929 (±0.090)	981	1.837 (±0.440)	600.293	394.139

Table B.23
Results on the Yoga dataset.

Yoga					
	Acc	*# Par*	*Time* (min.)	*MS Time* (min.)	*Energy* (kWh)
EuSN (ours)	0.742 (±0.010)	171	0.017 (±0.001)	3.126	1.367
ESN	0.702 (±0.011)	161	0.019 (±0.001)	2.973	1.367
R-ESN	0.706 (±0.011)	151	0.018 (±0.001)	3.099	6.512
DeepESN	0.684 (±0.023)	111	0.050 (±0.006)	8.118	4.378
RNN	0.610 (±0.035)	3781	3.236 (±1.095)	486.777	1381.872
A-RNN	0.669 (±0.003)	2651	3.032 (±0.231)	645.536	309.413
GRU	0.603 (±0.051)	11 401	0.334 (±0.124)	157.973	73.558
1D-CNN	0.544 (±0.043)	841	4.649 (±3.694)	125.104	113.863

Table B.24
Averaged test set RMSE achieved on the time-series modeling tasks by EuSN, EuSN$_{I-R}$ (i.e., EuSN including direct input-readout connections), ESN, and ESN$_{I-R}$ (i.e., ESN including direct input-readout connections).

Dataset	EuSN	EuSN$_{I-R}$	ESN	ESN$_{I-R}$
Adiac	5.971e−2 (± 5.728e−4)	2.379e−2 (± 2.242e−5)	2.522e−2 (± 2.452e−4)	2.327e−2 (± 4.820e−5)
ECG5000	4.027e−1 (± 3.071e−3)	2.088e−1 (± 1.392e−3)	2.007e−1 (± 1.200e−3)	1.871e−1 (± 9.550e−4)
FordA	3.837e−1 (± 3.335e−3)	9.361e−2 (± 4.363e−4)	2.800e−2 (± 5.612e−4)	3.321e−2 (± 9.351e−4)
HandOutlines	1.282e−2 (± 3.147e−4)	2.427e−3 (± 3.812e−6)	2.449e−3 (± 2.755e−5)	2.331e−3 (± 1.015e−5)
Mallat	1.490e−1 (± 4.806e−4)	6.120e−2 (± 1.015e−3)	5.521e−2 (± 6.847e−4)	4.765e−2 (± 7.146e−4)
ShapesAll	9.652e−2 (± 2.043e−3)	3.089e−2 (± 1.090e−4)	2.832e−2 (± 3.631e−5)	2.815e−2 (± 1.475e−5)
Trace	2.802e−1 (± 5.211e−4)	1.168e−1 (± 2.864e−3)	9.023e−2 (± 1.470e−3)	6.672e−2 (± 1.983e−3)
UWaveGestureLibraryAll	1.601e−1 (± 1.474e−3)	1.031e−1 (± 2.583e−5)	9.371e−2 (± 8.661e−4)	9.447e−2 (± 8.923e−4)
Wafer	4.746e−1 (± 4.858e−4)	4.153e−1 (± 3.961e−3)	3.839e−1 (± 1.141e−3)	3.728e−1 (± 7.287e−4)
Yoga	1.019e−1 (± 4.528e−4)	2.521e−2 (± 8.675e−5)	2.182e−2 (± 8.451e−5)	2.076e−2 (± 7.010e−5)

References

[1] C. Gallicchio, Reservoir computing by discretizing ODEs, in: Proceedings of ESANN, 2021.
[2] R.T. Chen, Y. Rubanova, J. Bettencourt, D.K. Duvenaud, Neural ordinary differential equations, Adv. Neural Inf. Process. Syst. 31 (2018).
[3] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, E. Holtham, Reversible architectures for arbitrarily deep residual neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, (1) 2018.
[4] B. Chang, L. Meng, E. Haber, F. Tung, D. Begert, Multi-level residual networks from dynamical systems view, 2017, arXiv preprint arXiv:1710.10348.
[5] Y. Lu, A. Zhong, Q. Li, B. Dong, Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations, in: International Conference on Machine Learning, PMLR, 2018, pp. 3276–3285.
[6] E. Haber, L. Ruthotto, Stable architectures for deep neural networks, Inverse Problems 34 (1) (2017) 014004.
[7] D. Verstraeten, B. Schrauwen, M. d'Haene, D. Stroobandt, An experimental unification of reservoir computing methods, Neural Net. 20 (3) (2007) 391–403.
[8] M. Lukoševičius, H. Jaeger, Reservoir computing approaches to recurrent neural network training, Comp. Sci. Rev. 3 (3) (2009) 127–149.
[9] M. Dragone, G. Amato, D. Bacciu, S. Chessa, S. Coleman, M. Di Rocco, C. Gallicchio, C. Gennaro, H. Lozano, L. Maguire, et al., A cognitive robotic ecology approach to self-configuring and evolving AAL systems, Eng. Appl. Artif. Intell. 45 (2015) 269–280.
[10] D. Bacciu, P. Barsocchi, S. Chessa, C. Gallicchio, A. Micheli, An experimental characterization of reservoir computing in ambient assisted living applications, Neural Comput. Appl. 24 (6) (2014) 1451–1464.

[11] S. Wang, Y. Li, D. Wang, W. Zhang, X. Chen, D. Dong, S. Wang, X. Zhang, P. Lin, C. Gallicchio, et al., Echo state graph neural networks with analogue random resistive memory arrays, Nat. Mach. Intell. (2023) 1–10.

[12] G. Milano, G. Pedretti, K. Montano, S. Ricci, S. Hashemkhani, L. Boarino, D. Ielmini, C. Ricciardi, In materia reservoir computing with a fully memristive architecture based on self-organizing nanowire networks, Nat. Mater. 21 (2) (2022) 195–202.

[13] 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 (2019) 100–123.

[14] J. Torrejon, M. Riou, F.A. Araujo, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, K. Yakushiji, A. Fukushima, et al., Neuromorphic computing with nanoscale spintronic oscillators, Nature 547 (7664) (2017) 428–431.

[15] D. Marković, J. Grollier, Quantum neuromorphic computing, Appl. Phys. Lett. 117 (15) (2020) 150501.

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

[17] H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks - with an erratum note, Tech. rep., GMD - German National Research Institute for Computer Science, 2001.

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

[19] I. Yildiz, H. Jaeger, S. Kiebel, Re-visiting the echo state property, Neural Net. 35 (2012) 1–9.

[20] P. Verzelli, C. Alippi, L. Livi, Learn to synchronize, synchronize to learn, Chaos 31 (8) (2021) 083119.

[21] A. Ceni, P. Ashwin, L. Livi, C. Postlethwaite, The echo index and multistability in input-driven recurrent neural networks, Physica D 412 (2020) 132609.

[22] G. Manjunath, H. Jaeger, Echo state property linked to an input: Exploring a fundamental characteristic of recurrent neural networks, Neural Comput. 25 (3) (2013) 671–696.

[23] M. Lukoševičius, A practical guide to applying echo state networks, in: Neural Networks: Tricks of the Trade, Springer, 2012, pp. 659–686.

[24] C. Gallicchio, A. Micheli, L. Pedrelli, Fast spectral radius initialization for recurrent neural networks, in: INNS Big Data and Deep Learning Conference, Springer, 2019, pp. 380–390.

[25] C. Gallicchio, A. Micheli, Architectural and markovian factors of echo state networks, Neural Netw. 24 (5) (2011) 440–456.

[26] P. Tiňo, B. Hammer, M. Bodén, Markovian bias of neural-based architectures with feedback connections, in: Perspectives of Neural-Symbolic Integration, Springer, 2007, pp. 95–133.

[27] B. Hammer, P. Tiňo, Recurrent neural networks with small weights implement definite memory machines, Neural Comput. 15 (8) (2003) 1897–1929.

[28] T. Strauss, W. Wustlich, R. Labahn, Design strategies for weight matrices of echo state networks, Neural Comput. 24 (12) (2012) 3246–3276.

[29] O.L. White, D.D. Lee, H. Sompolinsky, Short-term memory in orthogonal neural networks, Phys. Rev. Lett. 92 (14) (2004) 148102.

[30] M. Henaff, A. Szlam, Y. LeCun, Recurrent orthogonal networks and long-memory tasks, in: International Conference on Machine Learning, PMLR, 2016, pp. 2034–2042.

[31] P. Verzelli, C. Alippi, L. Livi, P. Tino, Input representation in recurrent neural networks dynamics, 2020, arXiv preprint arXiv:2003.10585.

[32] P. Tino, Dynamical systems as temporal feature spaces, J. Mach. Learn. Res. 21 (2020) 1649–1690.

[33] A. Rodan, P. Tiňo, Minimum complexity echo state network, IEEE Trans. Neural Netw. 22 (1) (2010) 131–144.

[34] B. Chang, M. Chen, E. Haber, E.H. Chi, Antisymmetricrnn: A dynamical system view on recurrent neural networks, 2019, arXiv preprint arXiv:1902.09689.

[35] U.M. Ascher, R.M. Mattheij, R.D. Russell, Numerical Solution of Boundary Value Problems for Ordinary Differential Equations, SIAM, 1995.

[36] P. Glendinning, Stability, Instability and Chaos: An Introduction To the Theory of Nonlinear Differential Equations, Cambridge University Press, 1994.

[37] L.N. Trefethen, D. Bau III, Numerical Linear Algebra, vol. 50, Siam, 1997.

[38] G. Strang, Introduction To Linear Algebra, vol. 3, Wellesley-Cambridge Press, Wellesley, MA, 1993.

[39] E. Süli, D.F. Mayers, An Introduction To Numerical Analysis, Cambridge University Press, 2003.

[40] C. Gallicchio, S. Scardapane, Deep randomized neural networks, Recent Trends Learning Data (2020) 43–68.

[41] R.T. Rockafellar, R.J.-B. Wets, Variational Analysis, vol. 317, Springer Science & Business Media, 2009.

[42] B. Bailey, Local Lyapunov exponents: predictability depends on where you are, in: Nonlinear Dynamics and Economics, Cambridge University Press, 1996, pp. 345–359.

[43] H.D. Abarbanel, R. Brown, M.B. Kennel, Local Lyapunov exponents computed from observed data, J. Nonlinear Sci. 2 (3) (1992) 343–365.

[44] D. Verstraeten, B. Schrauwen, On the quantification of dynamics in reservoir computing, in: International Conference on Artificial Neural Networks, Springer, 2009, pp. 985–994.

[45] F. Bianchi, L. Livi, C. Alippi, Investigating echo state networks dynamics by means of recurrence analysis, 2016, pp. 1–25, arXiv preprint arXiv:1601.07381.

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

[47] R. Legenstein, W. Maass, What makes a dynamical system computationally powerful, in: New directions in statistical signal processing: From systems to brain, 2007, pp. 127–154.

[48] J. Boedecker, O. Obst, J. Lizier, N. Mayer, M. Asada, Information processing in echo state networks at the edge of chaos, Theory Biosci. 131 (3) (2012) 205–213.

[49] F.L. Bauer, C.T. Fike, Norms and exclusion theorems, Numer. Math. 2 (1) (1960) 137–141.

[50] T. Allam Jr., A. Bahmanyar, R. Biswas, M. Dai, L. Galbany, R. Hložek, E.E. Ishida, S.W. Jha, D.O. Jones, R. Kessler, et al., The photometric lsst astronomical time-series classification challenge (plasticc): Data set, 2018, arXiv preprint arXiv:1810.00001.

[51] F. Alimoglu, E. Alpaydin, Combining multiple representations and classifiers for pen-based handwritten digit recognition, in: Proceedings of the Fourth International Conference on Document Analysis and Recognition, vol. 2, IEEE, 1997, pp. 637–640.

[52] A. Bagnall, J.L.W. Vickers, E. Keogh, The UEA & UCR time series classification repository, www.timeseriesclassification.com.

[53] A.C. Jalba, M.H. Wilkinson, J.B. Roerdink, Automatic segmentation of diatom images for classification, Microsc. Res. Tech. 65 (1–2) (2004) 72–85.

[54] K.O. Chicaiza, M.E. Benalcázar, A brain-computer interface for controlling IoT devices using eeg signals, in: 2021 IEEE Fifth Ecuador Technical Chapters Meeting, ETCM, IEEE, 2021, pp. 1–6.

[55] B.H. Williams, M. Toussaint, A.J. Storkey, Extracting motion primitives from natural handwriting data, in: International Conference on Artificial Neural Networks, Springer, 2006, pp. 634–643.

[56] A.L. Goldberger, L.A. Amaral, L. Glass, J.M. Hausdorff, P.C. Ivanov, R.G. Mark, J.E. Mietus, G.B. Moody, C.-K. Peng, H.E. Stanley, PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals, Circulation 101 (23) (2000) e215–e220.

[57] J.R. Villar, P. Vergara, M. Menéndez, E. de la Cal, V.M. González, J. Sedano, Generalized models for the classification of abnormal movements in daily life and its applicability to epilepsy convulsion recognition, Int. J. Neural Syst. 26 (06) (2016) 1650037.

[58] L.M. Davis, B.-J. Theobald, J. Lines, A. Toms, A. Bagnall, On the segmentation and classification of hand radiographs, Int. J. Neural Syst. 22 (05) (2012) 1250020.

[59] D.B. Dias, R.C. Madeo, T. Rocha, H.H. Biscaro, S.M. Peres, Hand movement recognition for brazilian sign language: a study using distance-based neural networks, in: 2009 International Joint Conference on Neural Networks, IEEE, 2009, pp. 697–704.

[60] S. Mallat, A Wavelet Tour of Signal Processing, Elsevier, 1999.

[61] M. Malekzadeh, R.G. Clegg, A. Cavallaro, H. Haddadi, Mobile sensor data anonymization, in: Proceedings of the International Conference on Internet of Things Design and Implementation, 2019, pp. 49–58.

[62] L.J. Latecki, R. Lakamper, T. Eckhardt, Shape descriptors for non-rigid shapes with a single closed contour, in: Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662), vol. 1, IEEE, 2000, pp. 424–429.

[63] N. Hammami, M. Bedda, Improved tree model for arabic speech recognition, in: 2010 3rd International Conference on Computer Science and Information Technology, vol. 5, IEEE, 2010, pp. 521–526.

[64] J. Liu, L. Zhong, J. Wickramasuriya, V. Vasudevan, Uwave: Accelerometer-based personalized gesture recognition and its applications, Pervasive Mob. Comput. 5 (6) (2009) 657–675.

[65] R.T. Olszewski, Generalized Feature Extraction for Structural Pattern Recognition in Time-Series Data, Carnegie Mellon University, 2001.

[66] A.L. Maas, R.E. Daly, P.T. Pham, D. Huang, A.Y. Ng, C. Potts, Learning word vectors for sentiment analysis, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Portland, Oregon, USA, 2011, pp. 142–150, URL http://www.aclweb.org/anthology/P11-1015.

[67] C. Apté, F. Damerau, S.M. Weiss, Automated learning of decision rules for text categorization, ACM Trans. Inform. Syst. 12 (3) (1994) 233–251.

[68] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[69] C. Gallicchio, A. Micheli, L. Pedrelli, Deep reservoir computing: A critical experimental analysis, Neurocomputing 268 (2017) 87–99, http://dx.doi.org/10.1016/j.neucom.2016.12.089.

[70] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014, arXiv preprint arXiv:1412.3555.

[71] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[72] C. Gallicchio, A. Micheli, Fast and deep graph neural networks, in: Proceedings of AAAI, 2020, arXiv preprint arXiv:1911.08941.

[73] C. Gallicchio, A. Micheli, Graph echo state networks, in: The 2010 International Joint Conference on Neural Networks, IJCNN, IEEE, 2010, pp. 1–8.

[74] A. Cini, I. Marisca, F.M. Bianchi, C. Alippi, Scalable spatiotemporal graph neural networks, in: Proceedings of AAAI, 2023, arXiv preprint arXiv:2209.06520.
[75] A. Micheli, D. Tortorella, Discrete-time dynamic graph echo state networks, Neurocomputing 496 (2022) 85–95.
[76] A. Gravina, D. Bacciu, C. Gallicchio, Anti-Symmetric DGN: a stable architecture for Deep Graph Networks, in: Proceedings of ICLR, 2023, arXiv preprint arXiv:2210.09789.

Claudio Gallicchio received the Ph.D. degree in computer science from the University of Pisa, Pisa, Italy, in 2011. He is currently an Assistant Professor at the Department of Computer Science, University of Pisa, Italy. His research interests include the fusion of concepts from deep learning, recurrent neural networks, and dynamical systems. He is the founder and former chair of the IEEE CIS Task Force on Reservoir Computing, and the founder and vicechair of the IEEE Task Force on Randomization-Based Neural Networks and Learning Systems.