A memristive computational neural network model for time-series processing

RESEARCH ARTICLE | MARCH 20 2025

A memristive computational neural network model for time-series processing

Special Collection: Neuromorphic Technologies for Novel Hardware AI

Veronica Pistolesi; Andrea Ceni ; Gianluca Milano ; Carlo Ricciardi ; Claudio Gallicchio

APL Mach. Learn. 3, 016117 (2025)
https://doi.org/10.1063/5.0255168

A memristive computational neural network model for time-series processing

Cite as: APL Mach. Learn. 3, 016117 (2025); doi: 10.1063/5.0255168
Submitted: 27 December 2024 • Accepted: 5 March 2025 •
Published Online: 20 March 2025

Veronica Pistolesi,[1] Andrea Ceni,[1] Gianluca Milano,[2] Carlo Ricciardi,[3] and Claudio Gallicchio[1,a)]

AFFILIATIONS

[1] Department of Computer Science, University of Pisa, 56127 Pisa, Italy
[2] Advanced Materials Metrology and Life Sciences Division, INRiM (Istituto Nazionale di Ricerca Metrologica), 10135 Torino, Italy
[3] Department of Applied Science and Technology, Politecnico di Torino, 10129 Torino, Italy

Note: This paper is part of the APL Machine Learning Special Topic on Neuromorphic Technologies for Novel Hardware AI.
[a)]**Author to whom correspondence should be addressed:** claudio.gallicchio@unipi.it

ABSTRACT

In this work, we introduce a novel computational framework inspired by the physics of memristive devices and systems, which we embed into the context of Recurrent Neural Networks (RNNs) for time-series processing. Our proposed memristive-friendly neural network architecture leverages both the principles of Reservoir Computing (RC) and fully trainable RNNs, providing a versatile platform for sequence learning. We provide a mathematical analysis of the stability of the resulting neural network dynamics, identifying the role of crucial RC-based architectural hyper-parameters. Through numerical simulations, we demonstrate the effectiveness of the proposed approach across diverse regression and classification tasks, showcasing performance that is competitive with both traditional RC and fully trainable RNN systems. Our results highlight the scalability and adaptability of memristive-inspired computational architectures, offering a promising path toward efficient neuromorphic computing for complex sequence-based applications.

I. INTRODUCTION

Neuromorphic computing, inspired by the structure and function of the human brain, aims to emulate biological information processing systems for the development of efficient and brain-like computational frameworks.[1] In a context where a hardware revolution is required to sustain the ever-growing demand for computing power boosted by the development of artificial intelligence,[2] a promising approach relies on exploiting the inherent physics of "intelligent materials" for computation.[3,4] For this purpose, a wide range of emerging technologies has been demonstrated for *in materia* computing by emulating hardware neuromorphic functionalities.[5,6]

Even if a generalized theory of computing with whatever physics offers is still missing,[3] Reservoir Computing (RC) has been reported to be a theoretical framework capable of leveraging the dynamic properties of materials for computing directly at the matter level.[7–9] RC offers a theoretically founded approach to realize efficiently trainable Recurrent Neural Networks (RNNs).[10–12] In RC, the recurrent layer, known as the reservoir, remains fixed during training, while only the readout layer is trained. This reduces training complexity and allows efficient mapping of input signals into high-dimensional spaces for tasks such as time-series processing. The fixed nature of the reservoir offers a natural avenue for integrating physical systems, enabling their use as computational reservoirs.[7] Among emerging hardware technologies, memristive devices and systems have been demonstrated as versatile platforms for hardware implementation of reservoir computing.[13–19] These implementations are based on the capability of memristive devices and systems to efficiently process temporal-dependent input signals by exploiting the intrinsic physics of the device.

In this paper, we introduce a physics-inspired computational model of an RNN system based on the physics underlying the working principle of memristive devices, with the aim of bridging the worlds of *in materia* computing and deep learning for time-series. Two specific computational models are presented: *Memristive-friendly Echo State Network* (MF-ESN), an RC approach, and *Memristive-friendly Recurrent Neural Network*

(MF-RNN), a fully trainable variant. We mathematically characterize the dynamical behavior of our introduced approach, studying conditions for asymptotic stability irrespective of training of the internal connections. Moreover, through computational experiments, we show the validity of our approach in the context of learning tasks on time-series, considering both regression and classification problems. Through these experiments, we demonstrate that our memristive-inspired models can achieve competitive results compared to traditional neural network methods. The experimental results showcase the adaptability of the memristive-based models to complex sequence processing, pointing toward their scalability and robustness. Our findings highlight the potential of these models in advancing next-generation intelligent systems and robust, adaptive computational frameworks. The remainder of this paper is organized as follows. Section II describes memristive dynamics from a physics perspective, including the mathematical model describing the dynamics of their internal state. After that, Sec. III introduces the preliminary core aspects of RNN systems and RC-based computational models. In Sec. IV, we detail the architecture of the proposed memristive-friendly neural networks, discussing the transformation of physical properties into computational mechanisms. Section V offers a mathematical analysis of the proposed approach, focusing on its stability analysis in the RC context. This includes conditions for stability and the exploration of architectural parameters critical for information representation. Section VI describes the computational experiments conducted to validate the proposed methodology, including a discussion of the datasets used, experimental setup, and results. Finally, the paper concludes in Sec. VII with a summary of our findings and a discussion of potential future research directions.

II. MEMRISTIVE DYNAMICS: A PHYSICS PERSPECTIVE

Memristive devices are two-terminal nanoscale devices where the internal state of resistance depends on the history of applied voltage and current.[20,21] By exploiting their inherent physics, these devices have been explored for the realization of artificial neurons and synapses[22] and represent promising candidates for bridging biological and artificial neural networks[23] as well as for the development of novel computing architectures.[24] In redox-based memristive cells, dynamics are regulated by nanoionic effects involving the dissolution and migration of ions,[25] resembling somehow the transfer of ions through the membrane of neuronal cells.[26] In the context of physical RC, memristive dynamics have been exploited for temporal processing of the input signal by exploiting nonlinear dynamics and short-term memory effects achievable in memristive devices working in the volatile regime, characterized by a spontaneous relaxation of the internal state toward the ground state after the end of stimulation.[27–29] By exploiting these memristive dynamics, physical reservoirs have been implemented in both top-down memristive devices and crossbar architectures,[13–16] as well as in bottom-up self-assembled memristive systems[17–19] (see Fig. 1).

In this context, nonlinear and short-term memristive dynamics can be described by a potentiation–depression rate balance equation[30]

$$\frac{d}{dt}g = \kappa_P(V)\cdot(1-g) - \kappa_D(V)\cdot g. \tag{1}$$

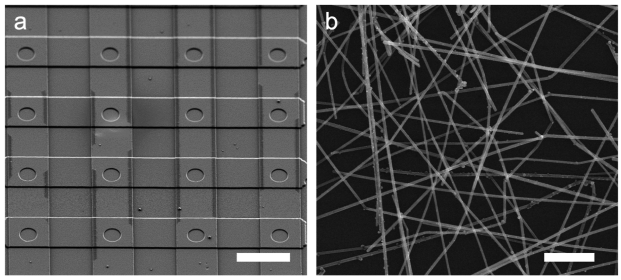

FIG. 1. Scanning electron microscopy image of (a) top-down memristive architecture (crossbar array) (scale bar: 10 μm). Here, each intersection between top and bottom electrode lines is a memristive device, and (b) bottom-up memristive architecture based on a self-assembled memristive nanowire network (scale bar: 2 μm). Each junction between nanowires behaves as a memristive element, exhibiting dynamics that depend on the history of applied voltage and current.

In Eq. (1), g is the normalized conductance (memory state) that assumes values in between 0 and 1, while $\kappa_P(V)$ and $\kappa_D(V)$ are the potentiation and depression rate coefficients that are assumed to be a function of the applied stimulation voltage V through exponential relations, as expected for diffusion of ions, mathematically described as follows:

$$\kappa_P(V) = \kappa_{P0}\ \exp(\eta_P V), \quad \kappa_D(V) = \kappa_{D0}\ \exp(-\eta_D V), \tag{2}$$

where κ_{P0}, $\kappa_{D0} > 0$ are constants, and η_P, $\eta_D > 0$ are transition rates. Note that Eq. (1), coupled with Ohm's law, is able to describe the evolution of current (the physical observable) flowing across the memristive device. This equation has been exploited for modeling short-term memristive dynamics in memristive cells based on single nanowires,[30] as well as for modeling memristive behavior in self-assembled nanowire networks[17,18,31–33] and neuromorphic networks.[34,35] Notably, this equation has also been exploited to model the internal dynamics of all-optical memitters (i.e., emitters with memory) that emulate short-term plasticity effects.[36]

III. RECURRENT AND RESERVOIR NEURAL NETWORKS

Recurrent Neural Networks (RNNs) are a class of computational neural architectures specifically designed to model sequential and time-series data by maintaining internal states that evolve over time. These networks incorporate feedback connections, enabling them to process variable-length input sequences and retain information from previous time steps (Fig. 2).

The fundamental operation of a recurrent layer is described by the following discrete-time dynamical system:

$$\mathbf{h}(t+1) = \phi(\mathbf{W}_h\mathbf{h}(t) + \mathbf{W}_x\mathbf{x}(t+1) + \mathbf{b}), \tag{3}$$

where $\mathbf{h}(t) \in \mathbb{R}^{N_h}$ represents the hidden state vector at time t, $\mathbf{x}(t) \in \mathbb{R}^{N_x}$ is the input vector, $\mathbf{W}_h \in \mathbb{R}^{N_h \times N_h}$ is the recurrent weight matrix, $\mathbf{W}_x \in \mathbb{R}^{N_h \times N_x}$ is the input weight matrix, $\mathbf{b} \in \mathbb{R}^{N_h}$ is the bias vector, and $\phi(\cdot)$ is a pointwise non-linearity, typically in the form of hyperbolic tangent, i.e., $\phi := \tanh$. The output signal is then given by

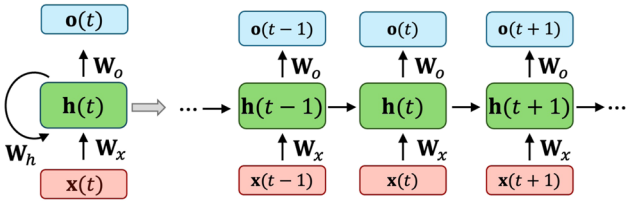

FIG. 2. Diagram of an RNN (left) and its operation unfolded over time (right). At each time t, the internal state of the network, denoted as $\mathbf{h}(t)$ in the figure, is used to encode the history of received input signals up to time step t. The output $\mathbf{o}(t)$ is computed from $\mathbf{h}(t)$ as a feed-forward operation. See more details in the text [Eqs. (3) and (4)].

$$\mathbf{o}(t) = \mathbf{W}_o \mathbf{h}(t), \tag{4}$$

where $\mathbf{o}(t)$ is the output signal and $\mathbf{W}_o$ is the output matrix.

The training of RNNs is typically performed using backpropagation through time (BPTT), an extension of the backpropagation algorithm that accounts for the temporal dependencies in the data. While BPTT enables the optimization of RNN parameters for a wide range of tasks, it suffers from high computational costs and difficulties in training due to the vanishing and exploding gradient problems.[37] These challenges have motivated the development of alternative approaches, such as Reservoir Computing (RC).

RC is a framework that simplifies the training of RNNs by leveraging the fixed dynamics of a high-dimensional recurrent layer, referred to as the *reservoir*, while restricting training to a linear output layer.[10,38] In the RC paradigm, the reservoir serves as a dynamical system that projects the input sequence into a high-dimensional space, enabling efficient representation of temporal patterns. In the context of discrete-time systems, the Echo State Network (ESN) model[39] implements the reservoir dynamics by using the same equation as in Eq. (3), but using fixed weight values in $\mathbf{W}_h$, $\mathbf{W}_x$, and $\mathbf{b}$. A crucial aspect of RC is ensuring the stability of the reservoir dynamics, often referred to as the Echo State Property (ESP). The ESP requires that the influence of initial conditions on the reservoir state vanishes over time, allowing the system to reliably encode the input sequence. The spectral radius of $\mathbf{W}_h$ (i.e., its largest eigenvalue in modulus) plays a key role in determining the stability of the reservoir. Accordingly, $\mathbf{W}_h$ is initialized with random values, typically from a uniform distribution over $[-1, 1]$, and then re-scaled to have a specific value of the spectral radius, denoted as ρ and treated as a hyper-parameter. Often, ρ is set to a value less than or equal to 1 in order to control the asymptotic stability of the resulting dynamics while preserving the nonlinear dynamics necessary for effective temporal processing.[40] The input weight matrix $\mathbf{W}_x$ is initialized with random values from a uniform distribution on $[-\omega, \omega]$, and similarly, the values in the bias vector $\mathbf{b}$ are drawn from a uniform distribution on $[-\beta, \beta]$. The values of ω and β are treated as input scaling and bias scaling hyper-parameters, respectively. The resulting approach is characterized by its simplicity and computational efficiency. In particular, the fixed nature of the reservoir eliminates the need for iterative training of the recurrent weights, drastically reducing training complexity. Instead, only the weights of the linear readout layer are trained, typically using ridge regression in closed form. RC has gained significant attention as a practical framework for time-series processing due to its ability to integrate physical systems as reservoirs. These systems, which can include optical, mechanical, or memristive devices, inherently satisfy the requirements of the RC paradigm, enabling efficient hardware implementations of RNN-like architectures.[7] In this paper, we extend the principles of RC to develop a memristive-inspired recurrent neural network architecture, which combines the physical dynamics of memristive networks with the computational efficiency of RC models. Notably, while in previous studies authors focused on the implementation of reservoir computing in memristive hardware,[13,16,17,41] in the following we focus on embedding dynamics typical of memristive systems in a recurrent neural network, providing new insights on how these dynamics affect computing capabilities.

IV. FROM PHYSICS TO A COMPUTATIONAL MODEL FOR MEMRISTIVE-FRIENDLY RECURRENT NEURAL NETWORKS

In the following, we develop the fundamental mathematical description of our memristive-friendly recurrent neural model. We exploited memristive dynamics described by Eq. (1) as the internal dynamics of a recurrent neuron in a RNN architecture. In this context, the normalized conductance g plays the role of memory state and can be reflected in the concept of neuron state from a deep learning perspective. The applied voltage V in Eq. (1) plays the role of an externally applied input that has the effect of driving the evolution of the state. Note that Eq. (1) endows in a single equation both (i) recurrence (i.e., the dependence of the neuron state on previous stimulation) and (ii) high non-linearity [given by $\kappa_P(V)$ and $\kappa_P(V)$]. In the proposed computational model, synaptic connections between recurrent neurons (memristive elements) are represented by means of the recurrent weight matrix $\mathbf{W}_h$, while connections from the external input are represented by the input weight matrix $\mathbf{W}_x$.

In the following, we use the symbols h and x to, respectively, denote the state and the received input of a recurrent neuron in an RNN architecture. The dynamics of h can then be described as follows: by suitably adapting Eqs. (1) and (2),

$$\frac{d}{dt}h = K_p(x) \cdot (1 - h) - K_d(x) \cdot h, \tag{5}$$

$$K_p(x) = K_{p_0} \cdot \exp(\eta_p x), \quad K_d(x) = K_{d_0} \cdot \exp(-\eta_d x), \tag{6}$$

where $h \in [0, 1]$, and $K_p(\cdot)$ and $K_d(\cdot)$ play the role of exponential-like non-linearities that process the input x. Notice that K_{p_0}, K_{d_0}, η_p, and η_d in Eq. (6) act as memristive-based hyper-parameters of the K_p and K_d non-linearities. In order to develop the discrete-time dynamics of a recurrent neuron, in line with conventional RNN and ESN approaches, we discretize Eq. (5) using the forward Euler method, leading to the following formulation:

$$\begin{aligned} h(t+1) = h(t) + \Delta_t [K_p(x(t+1)) \\ - (K_p(x(t+1)) + K_d(x(t+1)))h(t)], \end{aligned} \tag{7}$$

where $\Delta_t > 0$ is the step size used for the numerical integration, which we treat as a discretization time hyper-parameter for the

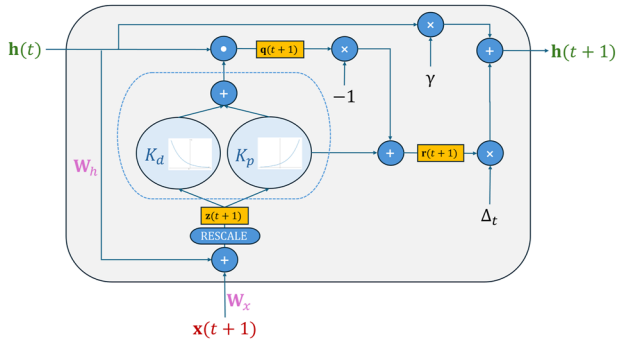

FIG. 3. Computational graph of the memristive-friendly recurrent layer, where $\mathbf{h}(t)$ and $\mathbf{h}(t+1)$, in green, are the current and the next hidden state vectors, respectively, $\mathbf{x}(t+1)$, in red, is the external input vector, $\mathbf{W}_h$ is the recurrent weight matrix (or recurrent kernel), and $\mathbf{W}_x$, in magenta, is the input weight matrix (or kernel). The input bias is omitted for simplicity. $\Delta_t > 0$ and $\gamma \in (0, 1]$ denote the informatics-based hyper-parameters, respectively, representing the discretization time and the stabilizer coefficient. Non-linearity operations are marked by a dashed line.

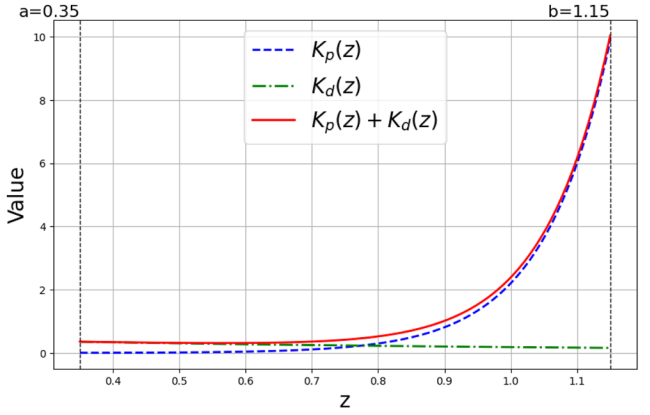

FIG. 4. Effective range of the non-linearities involved in the memristive-friendly recurrent neuron calculations after the rescaling operation. The plot assumes values of the memristive-based hyper-parameters as given in Table II.

resulting computational model. We use the discrete-time recurrent neuron model in Eq. (7) as the building block of a recurrent layer in an RNN architecture, whose computational graph is depicted graphically in Fig. 3. Specifically, we consider N_h recurrent neurons whose dynamics are described by Eq. (7), with a few additions, as detailed below.

First, each neuron is fed by the external input at the current time, the activation of all other recurrent neurons at the previous time, and a constant bias. Each of these sets of connections is modulated by its own set of weights, thereby enabling a flexible computation of the total effective input that stimulates the dynamics of each recurrent neuron. Although particular connectivity patterns can be modeled through the structure of the recurrent weight matrix $\mathbf{W}_h$, in our computational model we assume a densely connected network topology. This means that each recurrent neuron receives the (delayed) output of all others in the same layer, enabling an RC-like setup with random weights as detailed below when introducing the memristive-friendly reservoir model. Second, to address potential numerical instabilities arising from the interplay between the exponential-like non-linearities in the system, we apply a rescaling operation to the effective input of each recurrent neuron. This operation ensures that the input values fall within a range where the functions $K_p(\cdot)$ and $K_d(\cdot)$ exhibit well-behaved dynamics, avoiding extreme growth. Specifically, the rescaling is defined as

$$\text{RESCALE}(z) = \frac{(b-a)}{1+\exp(-z \cdot s)} + a, \tag{8}$$

where $a = 0.35$, $b = 1.15$, and s is a slope parameter. By confining the argument of these functions to such a range, the rescaling step enhances the robustness of the numerical integration and the stability of the resulting discrete-time dynamical system (as empirically analyzed in Sec. V). Moreover, the rescaling contributes to preserving the expressive power of the non-linearities, as it prevents the system from being dominated by either extreme linearity or extreme exponential growth, as graphically shown in Fig. 4. Third, in order to enhance the asymptotic stability of the memristive-friendly recurrent layer, we introduce a stabilizer hyper-parameter $\gamma \in (0, 1]$, whose effect is to facilitate the development of stable dynamics from the collective operation of the neurons in the recurrent layer (see Sec. V).

To provide a mathematical description of the resulting memristive-friendly recurrent network, we generalize the notation adopted so far, introducing $\mathbf{h}(t) \in \mathbb{R}^{N_h}$ to collectively denote the state of the N_h neurons in our recurrent layer and $\mathbf{x}(t) \in \mathbb{R}^{N_x}$ to represent an N_x-dimensional input signal. The dynamics of the memristive-friendly recurrent layer is then described by the following set of equations:

$$\mathbf{z}(t+1) = \text{RESCALE}(\mathbf{W}_h \mathbf{h}(t) + \mathbf{W}_x \mathbf{x}(t+1) + \mathbf{b}), \tag{9}$$

$$\mathbf{q}(t+1) = \text{diag}(K_p(\mathbf{z}(t+1)) + K_d(\mathbf{z}(t+1)))\mathbf{h}(t), \tag{10}$$

$$\mathbf{r}(t+1) = K_p(\mathbf{z}(t+1)) - \mathbf{q}(t+1), \tag{11}$$

$$\mathbf{h}(t+1) = \mathbf{r}(t+1)\Delta_t + \gamma\, \mathbf{h}(t), \tag{12}$$

where $\mathbf{W}_h \in \mathbb{R}^{N_h \times N_h}$ is the recurrent weight matrix (modulating the influence of the previous state of the layer on the new one), $\mathbf{W}_x \in \mathbb{R}^{N_h \times N_x}$ is the input weight matrix (modulating the influence of the new external input on the network state), $\mathbf{b} \in \mathbb{R}^{N_h}$ is a bias vector, and γ is the stabilizer coefficient.

Note that our model admits a number of hyper-parameters. Of these, K_{p_0}, K_{d_0}, η_p, and η_d represent physical quantities, i.e., they are memristive-based hyper-parameters, which in the following are fixed to physically plausible values.[31] In contrast, Δ_t, γ, and s represent informatics-based hyper-parameters of the computational model only, and in the following, they are appropriately explored in the computational experiments according to conventional fine-tuning strategies by model selection.

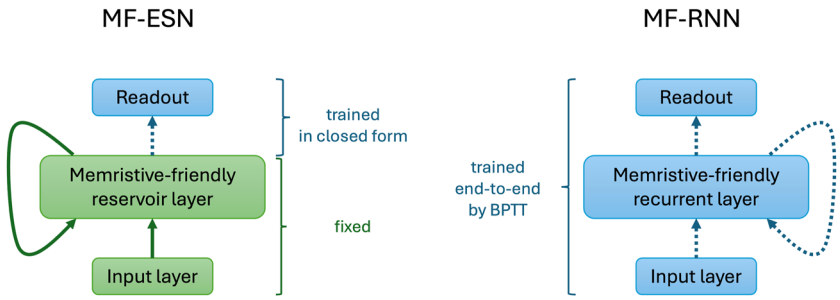

FIG. 5. Architectural organization of MF-ESN (left), in which the memristive-friendly recurrent layer is used as a reservoir system that is left untrained after initialization, and MF-RNN (right), where the memristive-friendly recurrent layer is included in a fully trainable recurrent neural model trained end-to-end. Green boxes denote untrained components, and blue boxes indicate trained ones. Dashed blue arrows indicate trainable connections.

In the following, we give two specific instances of the approach introduced so far, by framing it in the context of RC and fully trainable RNNs, as graphically illustrated in Fig. 5.

A. Memristive-friendly Echo State Network (MF-ESN)

In this case, the recurrent layer described by Eqs. (9)–(12) is used as the reservoir of an ESN-like model. Accordingly, the weight values in $\mathbf{W}_h$, $\mathbf{W}_x$, and $\mathbf{b}$ are left untrained after initialization. More specifically, aligning to the RC methodology described in Sec. III, the recurrent weight matrix $\mathbf{W}_h$ is randomly initialized and then re-scaled to have a specific spectral radius ρ (i.e., the largest length of an eigenvalue of $\mathbf{W}_h$), which is treated as a hyper-parameter. Moreover, the values in $\mathbf{W}_x$ and $\mathbf{b}$ are randomly generated from uniform distributions, respectively, on $[-\omega, \omega]$ and $[-\beta, \beta]$. The values of ω and β are treated as input scaling and bias scaling hyper-parameters, respectively. As in ESNs, training is restricted to a linear readout layer, which is trained in closed form using ridge-regression.

B. Memristive-friendly Recurrent Neural Network (MF-RNN)

In this case, we explore a fully trainable version of the recurrent layer described in Eqs. (9)–(12). The neural architecture is complemented by a linear readout dense layer, and it is trained end-to-end by BPTT (including the weights in $\mathbf{W}_h$, $\mathbf{W}_x$, and $\mathbf{b}$).

V. MATHEMATICAL ANALYSIS

In this section, we perform an analysis of the stability of the input-driven dynamical system defined by Eqs. (9)–(12). Linear stability analysis is a foundational mathematical approach used to investigate the behavior of a dynamical system near an equilibrium point. This method typically focuses on local stability by analyzing the eigenvalues of the Jacobian matrix, which represents the linearized system of equations around the equilibrium. If all eigenvalues lie inside the unit circle in the complex plane (for discrete-time systems), the equilibrium is locally stable, meaning small perturbations decay over time. Therefore, preventing the system from diverging and our computational model from producing unstable computations. Conversely, eigenvalues outside the unit circle indicate instability. However, this local perspective does not guarantee stability throughout the entire state space. Global stability, on the other hand, is often established using tools like contraction mappings, which assess whether the system as a whole converges to an equilibrium regardless of the initial condition. While eigenvalue-based analysis is sufficient for local behavior, global stability requires a more comprehensive examination of the system's nonlinear properties. This distinction highlights the complementary roles of local eigenvalue analysis and global contractivity in understanding system dynamics. For input-driven systems, the analysis becomes more intricate, as the external input can influence stability over time; however, similar tools can be employed with appropriate modifications to account for the system's input-dependent dynamics. In particular, equilibrium points may not exist in input-driven systems, and the stability is usually assessed for input-driven trajectories. In this section, we provide two stability results. First, a sufficient condition for global stability and the existence of a unique asymptotically stable input-drive solution for any input, based on the (spectral) norm of the Jacobian matrix.[42,43] Second, a necessary condition for local stability, based on the estimation of the location of the eigenvalues of the Jacobian.[40] Specifically, the Jacobian of the system Eqs. (9)–(12) is

$$\frac{\partial \mathbf{h}(t+1)}{\partial \mathbf{h}(t)} = \gamma \mathbf{I} + \Delta_t \mathbf{R}(t), \tag{13}$$

where $\mathbf{R}(t)$ represents the matrix whose entries are the derivatives $\frac{\partial(\mathbf{r}(t+1))_i}{\partial(\mathbf{h}(t))_j}$, for each subscript $i, j = 1, \ldots, N_h$, denoting the vector's components, and it has the following form:

$$\begin{aligned}&\{\eta_p \mathrm{diag}(K_p(\mathbf{z}(t+1)))\mathbf{W}_h \\ &\quad + [\mathrm{diag}(\eta_d K_d(\mathbf{z}(t+1)) - \eta_p K_p(\mathbf{z}(t+1)))]\mathbf{W}_h \mathrm{diag}(\mathbf{h}(t))\} \\ &\quad \times \mathrm{diag}(\sigma'(\mathbf{z}(t+1))) - \mathrm{diag}(K_p(\mathbf{z}(t+1)) + K_d(\mathbf{z}(t+1))),\end{aligned} \tag{14}$$

where diag is the operator transforming a vector in a diagonal matrix, and $\sigma :=$ RESCALE for the sake of conciseness.

A. Sufficient condition for stability

Informally speaking, a contraction mapping operating on a region of state space where trajectories are trapped inside implies global stability and the existence of a unique asymptotically stable solution for any input. Therefore, the strategy we follow to prove

the sufficient condition for global stability can be divided into three steps: (i) prove that initial conditions inside the closed unit disk evolve inside the closed unit disk for any input (i.e., the closed unit disk is positively invariant for any input), (ii) prove that the dynamical system of Eqs. (9)–(12) is a contraction (i.e., the spectral norm of the Jacobian is less than 1 for any input and at any point in state space), and (iii) exploit Theorem 4.1 in Ref. 42 to prove the existence and uniqueness of an asymptotically stable solution inside the closed unit disk for any input.

In the remainder of this paper, we denote $\|\mathbf{A}\|$ the spectral norm (or more simply, the norm) of a matrix $\mathbf{A}$.

We start proving in Lemma V.1 that the closed unit disk is a positively invariant set of the dynamics for any input. A proof can be found in Appendix A.

Lemma V.1. If $\Delta_t[2K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a}] + \gamma \le 1$, *then the closed unit disk* $\{\mathbf{h} : \|\mathbf{h}\| \le 1\}$ *is a positively invariant set of the dynamics of the MF model of Eqs. (9)–(12), for any input,* i.e., *for any input sequence* $\{\mathbf{x}(t+1)\}_{t\ge 0}$, *if the initial state* $\mathbf{h}(0)$ *lies in the closed unit disk, then the sequence of hidden states* $\{\mathbf{h}(t)\}_{t\ge 0}$ *lies inside the closed unit disk.*

We aim to find regions of hyper-parameters where the norm of the Jacobian is less than 1, i.e., a contractive dynamical system. For this purpose, we decompose the matrix $\mathbf{R}(t)$ of Eq. (14) in the sum of two terms, one dense and the other diagonal, $\mathbf{R}(t) = \mathbf{M}(t) - \mathrm{diag}(K_p(\mathbf{z}(t+1)) + K_d(\mathbf{z}(t+1)))$. Below, in Lemma V.2, we use Lemma V.1 to provide an upper bound on the spectral norm of the matrix $\mathbf{M}(t)$, which is independent of t. A Proof of Lemma V.2 can be found in Appendix B.

Lemma V.2. If $\Delta_t[2K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a}] + \gamma \le 1$ *and* $\|\mathbf{h}(0)\| \le 1$, *then the norm of the matrix* $\mathbf{M}(t)$ *admits the following upper bound:*

$$\|\mathbf{M}(t)\| \le \underbrace{s\frac{b-a}{4}}_{\text{rescaling}} \underbrace{\left(\eta_p K_{p_0} e^{\eta_p b} + \eta_d K_{d_0} e^{-\eta_d a} - \eta_p K_{p_0} e^{\eta_p a}\right)}_{\text{physics}} \underbrace{\|\mathbf{W_h}\|}_{\text{topology}}. \tag{15}$$

Lemma V.2 highlights three different groups of hyper-parameters involved in upper-bounding the norm of $\mathbf{M}(t)$. The first term involves a, b, s, which are directly related to the rescaling function of Eq. (9). The second term is linked to the physics of the memristive material with hyper-parameters $\eta_p, K_{p_0}, \eta_d, K_{d_0}$. Meanwhile, the third term correlates with the strength of the recurrent connections and the network topology.

We exploit the upper bound of Eq. (15) to provide a sufficient condition for the existence and uniqueness of an asymptotically stable input-driven solution of the proposed MF model that lies within the closed unit disk, as stated in the following theorem, swhose proof can be found in Appendix C.

Theorem V.3. *Assume* $\Delta_t[2K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a}] + \gamma \le 1$, *and* $\|\mathbf{h}(0)\| \le 1$. *Denote the constants* $C = \eta_p K_{p_0} e^{\eta_p b} + \eta_d K_{d_0} e^{-\eta_d a} - \eta_p K_{p_0} e^{\eta_p a}$, *and* $\mu = \max\left(\left|\gamma - \Delta_t(K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a})\right|, \gamma - \Delta_t(K_{p_0}e^{\eta_p a} + K_{d_0}e^{-\eta_d b})\right)$.

If $\mu + \Delta_t \frac{s(b-a)}{4}\|\mathbf{W}_h\|C < 1$, *then the MF model defined by Eqs. (9)–(12) admits a unique asymptotically stable input-driven solution for any input that lies in the closed unit disk.*

As usual in RC applications, sufficient conditions for global stability are often too restrictive. Therefore, necessary conditions for stability are usually provided to guide the hyper-parameter search. Those necessary conditions are usually based on the estimation of the largest in modulo eigenvalues of the Jacobian, and in particular of the spectral radius, rather than the norm of the Jacobian.

B. Necessary condition for stability

In this section, we seek ways to approximate the location of the eigenvalues of the Jacobian that allow us to derive necessary conditions for the stability of the MF model. The following theorem establishes an estimation of the location of the eigenvalues of the Jacobian of an MF model (see Appendix D for a proof).

Theorem V.4. *Denote the time-varying real values* $p_i(t) = K_p\big((\mathbf{z}(t+1))_i\big) + K_d\big((\mathbf{z}(t+1))_i\big)$. *For all* t, *each eigenvalue* $\chi(t)$ *of the Jacobian of the MF model must lie in a ball centered at* $\gamma - \Delta_t p_i(t)$, *for some* $i \in \{1, \dots, N_h\}$, *of radius* $\Delta_t\|\mathbf{M}(t)\|$. *Moreover, assuming* $\Delta_t[2K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a}] + \gamma \le 1$ *and* $\|\mathbf{h}(0)\| \le 1$, *we can estimate an upper bound for the radius as* $\Delta_t \frac{s(b-a)}{4} C\|\mathbf{W}_h\|$, *where* $C = \eta_p K_{p_0} e^{\eta_p b} + \eta_d K_{d_0} e^{-\eta_d a} - \eta_p K_{p_0} e^{\eta_p a}$.

Remark V.5. Theorem V.4 indicates the center points, $\gamma - \Delta_t p_i(t)$, as references to locate the eigenvalues of the Jacobian, the smaller the norm of $\mathbf{M}(t)$ or Δ_t and the greater the accuracy of the location. Therefore, at least for the case of weak coupling (i.e., small $\|\mathbf{W}_h\|$) or small step Δ_t, the location of the eigenvalues of the Jacobian is well approximated by the center points.

This remark translates into the following necessary condition for the eigenvalues of the Jacobian to be inside the unit circle for any input, whose proof can be found in Appendix E.

Theorem V.6. *In the case of a small step* Δ_t *or weak coupling (i.e., small* $\|\mathbf{W}_h\|$*), if the MF model has all eigenvalues inside the unit circle for any input, then*

$$\begin{cases} \gamma - \Delta_t p_{\max} > -1, \\ \gamma - \Delta_t p_{\min} < 1, \end{cases} \tag{16}$$

where $p_{\min} := K_{p_0}e^{\eta_p a} + K_{d_0}e^{-\eta_d b}$ *and* $p_{\max} := K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a}$. *Moreover, the following approximation of the spectral radius of the Jacobian holds:*

$$\rho(\mathbf{J}(t)) \approx \max\left(|\gamma - \Delta_t p_{\max}|, \gamma - \Delta_t p_{\min}\right). \tag{17}$$

In Fig. 6, the eigenvalues of the Jacobian for a few combinations of hyper-parameters are represented in blue, while the center points $\gamma - \Delta_t p_i(t)$ of Theorem V.4 are depicted in green together with their bounds $\gamma - \Delta_t p_{\max}$ and $\gamma - \Delta_t p_{\min}$ of Theorem V.6. These bounds can be used as guidelines since Eq. (16) are just necessary conditions

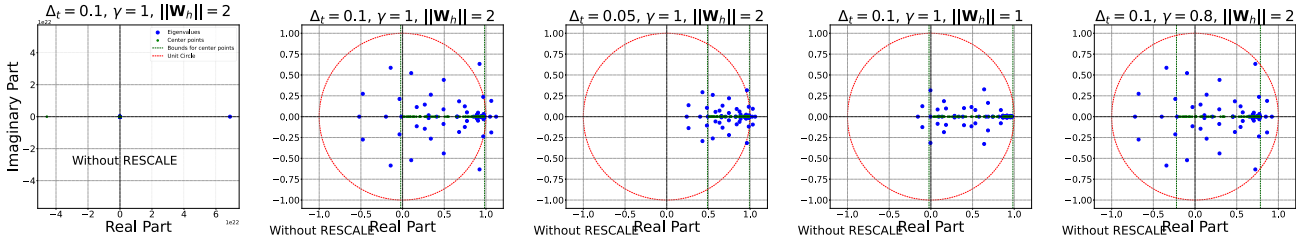

FIG. 6. Eigenspectrum of the Jacobian of Eq. (13) for a few combinations of hyper-parameters. Blue dots are eigenvalues, green dots are the center points $\gamma - \Delta_t p_i(t)$ of Theorem V.4, the red dashed line is the unit circle, and green dashed lines are the lower and upper bounds $\gamma - \Delta_t p_{\max}$ and $\gamma - \Delta_t p_{\min}$, respectively, of Theorem V.6. All Jacobians have been calculated using the same seed, just varying the hyper-parameters $\Delta_t, \gamma, \|\mathbf{W}_h\|$. The hidden state has dimension $N_h = 100$, and it is normally distributed with zero mean and unitary standard deviation. The input is a scalar value set to 1, matrix $\mathbf{W}_x$ is uniformly distributed in $(-5, 5)$, matrix $\mathbf{W}_h$ is uniformly distributed and re-scaled to a desired value of $\|\mathbf{W}_h\|$, and bias is zero. In the leftmost plot, the rescaling of Eq. (9) is not applied; the spectral radius of the Jacobian explodes to 10^{22}. Meanwhile, in all other cases, the rescaling is applied with $a = 0.35, b = 1.15, s = 1$. The hyper-parameters $\eta_p, \eta_d, K_{p_0}, K_{d_0}$ are those in Table II.

for the stability of the MF model; they are not sufficient to imply either global or local stability.

From Theorem V.4 we can deduce a few considerations to promote the stability of the MF model. Several hyper-parameters are involved in upper-bounding the exponential behavior characteristic of the MF model, namely $\eta_p, K_{p_0}, \eta_d, K_{d_0}$, but also the rescaling parameters a, b, s. In the following, we assume to have fixed values for the physical hyper-parameters $\eta_p, K_{p_0}, \eta_d, K_{d_0}$. Once these are fixed, the rescaling of Eq. (9) allows us to tame the numerical overflows that might arise when using exponential non-linearities. In fact, for certain combinations of hyper-parameters, as highlighted in the leftmost plot of Fig. 6, the eigenspectrum can explode without a proper rescaling. This stabilization effect allows us to explore wider regions of the hyper-parameters during model selection, thus enhancing the expressiveness of the model. Assuming fixed values for b, a, s for the rescaling function, we then focus on tuning the remaining software-like hyper-parameters γ, Δ_t, and the strength of the recurrent connectivity matrix $\mathbf{W}_h$. In this regard, the weaker the weights of the recurrent connectivity matrix and the smaller the $\|\mathbf{W}_h\|$ and, therefore, the greater the chance of having a stable MF model for any input. A similar stability effect can be enforced by selecting a small value of Δ_t. As can be seen from the center plot in Fig. 6, smaller values of Δ_t make the eigenspectrum contract uniformly but without directly affecting the recurrent connections. On the other hand, as evident from the rightmost plot in Fig. 6, γ acts as a translation on the eigenspectrum. Hence, the smaller the value of γ, the more the eigenvalues are shifted to the left, thus providing an additional flexibility to enforce stable dynamics for any input.

VI. COMPUTATIONAL EXPERIMENTS

To evaluate the performance of the proposed memristive-inspired neural network models, we conducted an extensive set of computational experiments focusing on time-series classification and regression tasks. The experiments were designed to benchmark the capabilities of both MF-ESN and MF-RNN models against conventional approaches, including ESNs and fully trainable RNNs. To this end, we have developed a software library that implements the MF-ESN and MF-RNN equations described in Sec. IV, in Python,

TABLE I. Overview of the datasets used in the computational experiments. Reported information includes the number of sequences in training (train size) and test (test size), the max length of a sequence in the dataset (length), the number of output classes (classes) for the time-series classification problems, the number of input features (input dim.), and the type of sequences (type).

Dataset	Train size	Test size	Length	Classes	Input dim.	Type
JapaneseVowels(↑)[44]	270	370	29	9	12	Audio
SyntheticControl(↑)[44]	300	300	60	6	1	Simulated
ECG5000(↑)[44]	500	4500	140	5	1	ECG
GunPoint(↑)[44]	50	150	150	2	1	HAR
Wafer(↑)[44]	1000	6164	152	2	1	Sensor
Epilepsy(↑)[44]	137	138	207	4	3	HAR
Coffee(↑)[44]	28	28	286	2	1	Spectro
FloodModeling1(↓)[45]	471	202	266	···	1	Environment Monitoring
FloodModeling2(↓)[45]	389	167	266	···	1	Environment Monitoring
FloodModeling3(↓)[45]	429	184	266	···	1	Environment Monitoring

TABLE II. Fundamental memristive-related hyper-parameters values used in the computational experiments.

Hyper-parameter	Value
K_{p0}	0.0001
K_{d0}	0.5
η_p	10
η_d	1

as custom Keras layers. Our code is made publicly available at https://github.com/VeronicaPistolesi/Memristive-Friendly-RNN.

The datasets used in this study span a variety of temporal dynamics and application domains, ensuring a comprehensive evaluation. Specifically, we considered the `JapaneseVowels`, `SyntheticControl`, `ECG5000`, `GunPoint`, `Wafer`, `Epilepsy`, and `Coffee` datasets from the UCR archive[44] for classification tasks. For regression tasks, we utilized `FloodModeling1`, `FloodModeling2`, and `FloodModeling3`, sourced from the flood modeling benchmark suite,[45] which consists of hourly rainfall event time series used as input to predict the maximum water depth in a given Digital Elevation Model (DEM) domain. A summary of the datasets, including key characteristics such as sequence length, number of classes and input features, data split sizes, and type of sequences, is provided in Table I.

For all datasets, the predefined splits into training and test sets were respected. To facilitate hyper-parameter tuning, we further partitioned the training set into a training subset and a validation subset, allocating 80% of the data for training and 20% for validation. This setup was consistently employed across all experiments to ensure robust model selection and reliable assessment of performance.

In all experiments, the architectures of the MF-ESN and MF-RNN models were compared with their traditional counterparts, ESN and RNN, respectively. For the ESN model, the leaky-integrator neuron formulation[46] was adopted, incorporating a leaking-rate hyper-parameter α. In every case, the output layer was applied to the final hidden state of the recurrent component (i.e., the state at the last time step of each sequence), providing a unified framework for evaluating the sequence processing capabilities of the models. For all experiments, we used networks with a number of $N_h = 100$ recurrent neurons. During the model selection, the memristive-related hyper-parameters of the memristive-based neural models were set to the values reported in Table II, taken from Ref. 31, which provided a set of physically meaningful values from a neuromorphic experimental setting. Table III(a)–(d) shows the ranges of values explored for the remaining hyper-parameters for MF-ESN, MF-RNN, ESN, and RNN, respectively (i.e., the informatics-based ones). For the RC models, we used an initialization process that regulates the strength of the recurrent connections using the spectral radius as Ref. 47, rather than the norm, as is more commonly adopted in the RC literature. For the MF models, we made sure the grid of hyper-parameters is coherent with the theoretical guidelines derived in Theorem V.6. With regard to the training algorithms, the readout of the RC models (i.e., MF-ESN and ESN) was trained in closed form by ridge regression (with Tikhonov regularizer $\lambda = 1$). The fully trainable computational models (i.e., MF-RNN and RNN), on the other hand, were trained with the Adam algorithm for 5000 epochs, using early stopping with a patience of 10 and a batch size of 64. After model selection, the model evaluation of all models was based on the average performance over five trials.

A. Results

The experimental results, summarized in Table IV, provide a comprehensive evaluation of the proposed MF-ESN and MF-RNN architectures across both classification and regression tasks. Details on the hyper-parameters selected through model selection can be

TABLE III. Values of the hyper-parameters explored during model selection for the different models used in the computational experiments.

Hyper-parameter	Values	Hyper-parameter	Values
Input scaling ω	{0.1, 1, 10}	Learning rate	{0.001, 0.01}
Bias scaling β	{0, 0.001, 0.1, 1}	Stabilizer γ	{0.5, 0.8, 0.95, 1}
Spectral radius ρ	{0.8, 0.9, 0.95, 0.99}	Slope s	{1, 5}
Stabilizer γ	{0.5, 0.8, 0.95, 1}	Discretization time Δ_t	{0.001, 0.01, 0.1}
Slope s	{1, 5}		
Discretization time Δ_t	{0.1, 0.01, 0.001}		
(a) MF-ESN		(b) MF-RNN	

Hyper-parameter	Values	Hyper-parameter	Values
Input scaling ω	{0.1, 1, 10}	Learning rate	{0.001, 0.01}
Bias scaling β	{0, 0.001, 0.1, 1}		
Spectral radius ρ	{0.8, 0.9, 0.95, 0.99}		
Leaking-rate α	{0.1, 0.3, 0.5, 0.7, 1}		
(c) ESN.		(d) RNN.	

TABLE IV. Experimental results. Reported metrics are the accuracy for classification tasks (↑) and the root mean squared error for regression ones (↓). The best results are highlighted in bold.

Task	ESN	MF-ESN (*ours*)	RNN	MF-RNN (*ours*)
JapaneseVowels(↑)	**0.984 ± 0.003**	0.974 ± 0.004	0.916 ± 0.011	0.951 ± 0.004
SyntheticControl(↑)	0.893 ± 0.010	**0.950 ± 0.007**	0.799 ± 0.119	0.927 ± 0.048
ECG5000(↑)	0.915 ± 0.002	**0.924 ± 0.002**	0.804 ± 0.247	0.909 ± 0.004
GunPoint(↑)	0.593 ± 0.023	**0.769 ± 0.070**	0.501 ± 0.007	0.751 ± 0.012
Wafer(↑)	**0.986 ± 0.002**	**0.986 ± 0.001**	0.804 ± 0.109	0.910 ± 0.037
Epilepsy(↑)	0.868 ± 0.015	**0.961 ± 0.006**	0.448 ± 0.106	0.884 ± 0.013
Coffee(↑)	0.586 ± 0.029	0.586 ± 0.017	0.543 ± 0.014	**0.950 ± 0.029**
FloodModeling1(↓)	0.019 ± 0.000	**0.008 ± 0.000**	0.022 ± 0.002	0.009 ± 0.001
FloodModeling2(↓)	0.018 ± 0.000	**0.015 ± 0.001**	0.019 ± 0.001	0.017 ± 0.000
FloodModeling3(↓)	0.023 ± 0.000	**0.010 ± 0.001**	0.027 ± 0.003	**0.010 ± 0.000**

found in Appendix F. Performance is benchmarked against traditional ESN and RNN models, with classification accuracy (↑) and root mean squared error (↓) used as evaluation metrics for classification and regression tasks, respectively.

The results reveal the competitive performance of memristive-based models, particularly the MF-ESN, which achieves the best performance on almost every task, often significantly surpassing the accuracy of the conventional ESN. For instance, in the `SyntheticControl` dataset, MF-ESN achieves an accuracy of 0.950 ± 0.007, compared to 0.893 ± 0.010 for the ESN, demonstrating its superior ability to encode and process temporal patterns. Similarly, MF-ESN shows an improvement over ESN on `Epilepsy`, with an accuracy of 0.961 ± 0.006 compared to 0.868 ± 0.015, highlighting the advantages of incorporating memristive-inspired dynamics in reservoir-based models. Moreover, on `GunPoint`, MF-ESN achieves 0.769 ± 0.070 against 0.593 ± 0.023 of ESN. In regression tasks, MF-ESN outperforms ESN by a substantial margin in all cases, achieving an order of magnitude lower root mean squared error (RMSE) on `FloodModeling1` and halving the RMSE on `FloodModeling3`, with respect to ESN.

The MF-RNN model exhibits robust performance, often comparable to or better than the standard RNN. On the `Coffee` dataset, MF-RNN achieves the highest classification accuracy of 0.950 ± 0.029, outperforming both RNN (0.543 ± 0.014) and MF-ESN (0.585 ± 0.029). These results suggest that the introduction of memristive-inspired dynamics into a fully trainable recurrent network can enhance its capacity to learn temporal dependencies while retaining the flexibility afforded by end-to-end training.

It is worth noting that, while MF-RNN offers slight gains over RNN, MF-ESN consistently outperforms the standard ESN in most cases. This highlights the efficiency of the proposed MF-based architecture in extracting and processing temporal features. Importantly, MF-ESN achieves these improvements without increasing computational complexity, as it retains the characteristic simplicity of the RC framework by keeping the reservoir weights fixed.

Overall, these findings underscore the potential of memristive-inspired architectures, particularly MF-ESN, as effective and scalable solutions for time-series processing. The observed improvements illustrate the adaptability and robustness of the MF-inspired approach.

VII. CONCLUSIONS

In this work, we introduced a novel computational framework inspired by the physics of memristive devices and systems, leveraging their memristive dynamics to design neural architectures suitable for time-series processing. We presented two specific computational models, the Memristive-friendly Echo State Network (MF-ESN) and the Memristive-friendly Recurrent Neural Network (MF-RNN), which combine the principles of reservoir computing and fully trainable recurrent networks, respectively. Our analysis provided a mathematical characterization of the stability of these models, and extensive computational experiments were conducted to evaluate their performance across a diverse set of time-series classification and regression tasks.

The results highlight the effectiveness of memristive-inspired computational models. In particular, MF-ESN demonstrated remarkable performance, outperforming traditional ESNs in nearly all tasks while maintaining the simplicity and computational efficiency characteristic of reservoir computing. Notably, MF-ESN achieved the best performance results in several benchmarks, showcasing its ability to capture and process complex temporal patterns efficiently. Similarly, MF-RNN offered improvements over standard RNNs in some tasks, illustrating the benefits of incorporating memristive-inspired dynamics into fully trainable architectures. The performance of memristive-based models in regression tasks further underscores their robustness and adaptability, pointing toward their potential in a wide range of applications.

These findings open up promising avenues for future research. One interesting direction involves exploring deeper architectures, such as deep recurrent structures (as in deep reservoirs[11]), to further enhance the representational power of memristive-based models. In this context, it is also important to note that the sparsity (and pruning) of connectivity patterns could play a relevant role, which is an emerging aspect in the general landscape of deep learning research,[48,49] in neural networks based on random weights,[50] and also in reservoir computing.[51,52] In addition, modular systems that integrate memristive dynamics with other neuromorphic components could provide a pathway to more flexible and specialized architectures. A further promising area is the development of hybrid models combining MF-ESN and MF-RNN elements, potentially leveraging the strengths of both approaches in a unified framework.

The RC-based models resulting from these lines of research could also be optimized by using advanced reservoir hyper-parameter optimization strategies, such as in the learn-to-learn framework.[53–55] Finally, the reported results can provide guidance on the development of new hardware architectures leveraging the physics of emerging devices for computation. In this context, the dynamics of a recurrent neuron can be directly mapped in the dynamics of a single memristive element, where the neuron input is in the form of a voltage stimulation. Note that, while in the computational model the dynamic response of the recurrent neuron is temporally discretized, memristive dynamics works in a continuum temporal domain. In this context, note that the discrete-time solution of Eq. (1) represents an approximation of the actual time-continuous response of the memristive element. In addition, it is worth remarking that memristive-based hyper-parameters represent physical quantities that depend on the memristive cell configuration in terms of physicochemical properties of involved materials. Therefore, in physical systems, these hyper-parameters are constrained by physicochemical properties governing device functioning. At the architecture level, the realization of a fully hardware RNN requires, in addition to recurrent neurons, the hardware implementation of synaptic connections here represented by the weight matrix $\mathbf{W}_h$ (recurrent weight matrix) and $\mathbf{W}_x$ (input weight matrix). These trainable weights can be implemented, for example, in memristive crossbar arrays[56] to realize a fully memristive RNN hardware architecture. While the here proposed theoretical computational approach assumes a dense connection between recurrent neurons with a consequent large number of (synaptic) weights, further work is required to understand the influence of the network topology on computational performances. Indeed, the realization of architectures with sparse connections will enable the reduction of the number of weights to be mapped in physical devices, largely simplifying the hardware architecture mapping the RNN. In this context, scalability studies should be performed. Finally, we envision that the hardware implementation of the proposed theoretical computational model can be explored in a wide range of real-world physical dynamical systems (not only memristive devices) characterized by an internal state that depends on the history of previous stimulation and endows high non-linearity.

ACKNOWLEDGMENTS

This work has been supported by NEURONE, a project funded by the European Union—Next Generation EU, M4C1 CUP No. I53D23003600006, under program PRIN 2022 (prj code 20229JRTZA), and by EU-EIC EMERGE (Grant No. 101070918). Part of this work has been supported by the European Research Council (ERC) under the European Union's ERC Starting Grant (ERC-2024-STG) Agreement "MEMBRAIN" No. 101160604.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Veronica Pistoles and Andrea Ceni contributed equally to this work.

Veronica Pistolesi: Conceptualization (equal); Investigation (equal); Methodology (equal); Software (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Andrea Ceni**: Conceptualization (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Software (equal); Supervision (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Gianluca Milano**: Conceptualization (equal); Funding acquisition (equal); Investigation (equal); Methodology (equal); Writing – original draft (equal); Writing – review & editing (equal). **Claudio Gallicchio**: Conceptualization (equal); Formal analysis (equal); Funding acquisition (equal); Investigation (equal); Methodology (equal); Project administration (equal); Software (equal); Supervision (equal); Writing – original draft (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data and the experiments that support the findings of this study are openly available. Datasets are publicly available at http://timeseriesclassification.com and http://tseregression.org. Our code is publicly available at https://github.com/VeronicaPistolesi/Memristive-Friendly-RNN.

APPENDIX A: PROOF OF LEMMA V.1

Lemma. If $\Delta_t[2K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a}] + \gamma \le 1$, then the closed unit disk $\{\mathbf{h} : \|\mathbf{h}\| \le 1\}$ is a positively invariant set of the dynamics of the MF model of Eqs. (9)–(12); for any input, i.e., for any input sequence $\{\mathbf{x}(t+1)\}_{t\ge 0}$, if the initial state $\mathbf{h}(0)$ lies in the closed unit disk, then the sequence of hidden states $\{\mathbf{h}(t)\}_{t\ge 0}$ lies inside the closed unit disk.

Proof. We start from Eqs. (9)–(12) by writing

$$\mathbf{h}(t+1) = \gamma\mathbf{h}(t) + \Delta_t\{K_p(\mathbf{z}(t+1)) - [\mathrm{diag}(K_p(\mathbf{z}(t+1))) + \mathrm{diag}(K_d(\mathbf{z}(t+1)))]\mathbf{h}(t)\},$$

therefore,

$$\begin{aligned}\|\mathbf{h}(t+1)\| &\le \gamma\|\mathbf{h}(t)\| + \Delta_t\big\{\|K_p(\mathbf{z}(t+1))\| \\ &\quad + \|\mathrm{diag}(K_p(\mathbf{z}(t+1))) + \mathrm{diag}(K_d(\mathbf{z}(t+1)))\|\|\mathbf{h}(t)\|\big\} \\ &= \Delta_t\|K_p(\mathbf{z}(t+1))\| + \|\mathbf{h}(t)\|\big\{\gamma + \Delta_t\|\mathrm{diag}(K_p(\mathbf{z}(t+1))) \\ &\quad + \mathrm{diag}(K_d(\mathbf{z}(t+1)))\|\big\} \le \Delta_t \max_{x\in(a,b)} K_p(x) \\ &\quad + \|\mathbf{h}(t)\|\{\gamma + \Delta_t[\max_{x\in(a,b)}(K_p(x) + K_d(x))]\} \\ &\le \Delta_t K_{p_0}e^{\eta_p b} + \|\mathbf{h}(t)\|\{\gamma + \Delta_t[K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a}]\},\end{aligned}$$

where, in the last inequality, we used the fact that the function $K_p(x)$ is monotonically increasing, hence $\max_{x\in(a,b)}K_p(x) = K_{p_0}e^{\eta_p b}$, and

the fact that the function $K_d(x)$ is monotonically decreasing, hence $\max_{x\in(a,b)} K_d(x) = K_{d_0}e^{-\eta_d a}$. Therefore, we derived that

$$\|\mathbf{h}(t+1)\| \le \Delta_t K_{p_0} e^{\eta_p b} + \|\mathbf{h}(t)\|\{\gamma + \Delta_t[K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a}]\}.$$

Now, starting with an initial condition $\|\mathbf{h}(0)\| \le 1$, we have $\|\mathbf{h}(1)\| \le \Delta_t[2K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a}] + \gamma$. Therefore, if $\Delta_t[2K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a}] + \gamma \le 1$, then by induction, we have that $\|\mathbf{h}(t)\| \le 1$, for all t. □

APPENDIX B: PROOF OF LEMMA V.2

Lemma. If $\Delta_t[2K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a}] + \gamma \le 1$ and $\|\mathbf{h}(0)\| \le 1$, then the norm of the matrix $\mathbf{M}(t)$ admits the following upper bound:

$$\|\mathbf{M}(t)\| \le \underbrace{s\frac{b-a}{4}}_{\text{rescaling}} \underbrace{\left(\eta_p K_{p_0}e^{\eta_p b} + \eta_d K_{d_0}e^{-\eta_d a} - \eta_p K_{p_0}e^{\eta_p a}\right)}_{\text{physics}} \underbrace{\|\mathbf{W_h}\|}_{\text{topology}}.$$

Proof. By definition of $\sigma :=$ RESCALE, we have $\|\mathrm{diag}(\sigma'(\mathbf{z}(t+1)))\| \le \|\mathrm{diag}(\sigma'(\mathbf{0}))\| = \frac{s(b-a)e^0}{(1+e^0)^2} = \frac{s(b-a)}{4}$. Since the function $K_p(x)$ is monotonically increasing, then $\|\mathrm{diag}(K_p(\mathbf{z}(t+1)))\| \le \|\mathrm{diag}(K_p(\mathbf{b}))\| = K_{p_0}e^{\eta_p b}$. Meanwhile, since the function $K_d(x)$ is monotonically decreasing, then $\|\mathrm{diag}(K_d(\mathbf{z}(t+1)))\| \le \|\mathrm{diag}(K_d(\mathbf{a}))\| = K_{d_0}e^{-\eta_d a}$. Now, we observe that by Lemma V.1 it holds that $\|\mathbf{h}(t)\| \le 1$, for all t, as long as $\|\mathbf{h}(0)\| \le 1$, which implies that $\max_i|(\mathbf{h}(t))_i| \le 1$, for all t. Therefore, $\|\mathrm{diag}(\mathbf{h}(t))\| = \max_i|(\mathbf{h}(t))_i| \le 1$.

Therefore, applying the triangle inequality and wrapping it all together, we have that

$$\|\mathbf{M}(t)\| \le s\frac{(b-a)}{4}\Big\{\eta_p K_{p_0}e^{\eta_p b}\|\mathbf{W}_h\| + \|\mathrm{diag}(\eta_d K_d(\mathbf{z}(t+1)) - \eta_p K_p(\mathbf{z}(t+1)))\|\|\mathbf{W}_h\|\Big\}.$$

Finally, we exploit the monotonic trends of K_p (increasing) and K_d (decreasing) to upper bound as follows:

$$\begin{aligned}&\|\mathrm{diag}(\eta_d K_d(\mathbf{z}(t+1)) - \eta_p K_p(\mathbf{z}(t+1)))\| \\ &\le \max_{x\in(a,b)}(\eta_d K_d(x)) - \min_{x\in(a,b)}(\eta_p K_p(x)) \\ &= \eta_d K_{d_0}e^{-\eta_d a} - \eta_p K_{p_0}e^{\eta_p a}.\end{aligned}$$

Concluding we can write the upper bound $\|\mathbf{M}(t)\| \le s\frac{(b-a)}{4}\left\{\eta_p K_{p_0}e^{\eta_p b} + \eta_d K_{d_0}e^{-\eta_d a} - \eta_p K_{p_0}e^{\eta_p a}\right\}\|\mathbf{W}_h\|$, which is the thesis. □

APPENDIX C: PROOF OF THEOREMs V.3

Theorem. Assume $\Delta_t[2K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a}] + \gamma \le 1$, and $\|\mathbf{h}(0)\| \le 1$. Denote the constants $C = \eta_p K_{p_0}e^{\eta_p b} + \eta_d K_{d_0}e^{-\eta_d a} - \eta_p K_{p_0}e^{\eta_p a}$, and $\mu = \max\left(\left|\gamma - \Delta_t(K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a})\right|, \gamma - \Delta_t(K_{p_0}e^{\eta_p a} + K_{d_0}e^{-\eta_d b})\right)$.

If $\mu + \Delta_t\frac{s(b-a)}{4}\|\mathbf{W}_h\|C < 1$, then the MF model defined by Eqs. (9)–(12) admits a unique asymptotically stable input-driven solution for any input that lies in the closed unit disk.

Proof. By Lemma V.1, if $\Delta_t[2K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a}] + \gamma \le 1$, then the closed unit disk is a positively invariant set of the dynamics. We will prove that the Jacobian $\mathbf{J}(t) = \frac{\partial \mathbf{h}(t+1)}{\partial \mathbf{h}(t)}$ is such that $\sup_t\|\mathbf{J}(t)\| < 1$. This, together with the fact that the closed unit disk is a convex and compact positively invariant set of the dynamics, implies that the system admits a unique asymptotically stable input-driven trajectory for any input (see Theorem 4.1 in Ref. 42). We write the Jacobian as

$$\begin{aligned}\mathbf{J}(t) &= \gamma\mathbf{I} + \Delta_t\mathbf{R}(t) \\ &= \gamma\mathbf{I} - \Delta_t\mathrm{diag}(K_p(\mathbf{z}(t+1)) + K_d(\mathbf{z}(t+1))) + \Delta_t\mathbf{M}(t).\end{aligned}$$

The maximum absolute value of the diagonal matrix $\gamma\mathbf{I} - \Delta_t\mathrm{diag}(K_p(\mathbf{z}(t+1)) + K_d(\mathbf{z}(t+1)))$ is either $\gamma - \Delta_t\min_{x\in(a,b)}(K_p(x) + K_d(x))$ or $\left|\gamma - \Delta_t\max_{x\in(a,b)}(K_p(x) + K_d(x))\right|$. Therefore,

$$\begin{aligned}&\|\gamma\mathbf{I} - \Delta_t\mathrm{diag}(K_p(\mathbf{z}(t+1)) + K_d(\mathbf{z}(t+1)))\| \\ &= \max\Big(\Big|\gamma - \Delta_t(K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a})\Big|, \\ &\quad \gamma - \Delta_t(K_{p_0}e^{\eta_p a} + K_{d_0}e^{-\eta_d b})\Big).\end{aligned}$$

Finally, if we denote the constants

$$\begin{aligned}C &= \eta_p K_{p_0}e^{\eta_p b} + \eta_d K_{d_0}e^{-\eta_d a} - \eta_p K_{p_0}e^{\eta_p a}, \\ \mu &= \max\Big(\Big|\gamma - \Delta_t(K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a})\Big|, \\ &\quad \gamma - \Delta_t(K_{p_0}e^{\eta_p a} + K_{d_0}e^{-\eta_d b})\Big),\end{aligned}$$

then by Lemma V.2, it holds

$$\begin{aligned}\|\mathbf{J}(t)\| &\le \|\gamma\mathbf{I} - \Delta_t\mathrm{diag}(K_p(\mathbf{z}(t+1)) + K_d(\mathbf{z}(t+1)))\| + \Delta_t\|\mathbf{M}(t)\| \\ &\le \mu + \Delta_t\frac{s(b-a)}{4}\,C\,\|\mathbf{W}_h\|.\end{aligned}$$

□

APPENDIX D: PROOF OF THEOREM V.4

Theorem. Denote the time-varying real values $p_i(t) = K_p\big((\mathbf{z}(t+1))_i\big) + K_d\big((\mathbf{z}(t+1))_i\big)$. For all t, each eigenvalue $\chi(t)$ of the Jacobian of the MF model must lie in a ball centered at $\gamma - \Delta_t p_i(t)$, for some $i \in \{1,\dots,N_h\}$, of radius $\Delta_t\|\mathbf{M}(t)\|$. Moreover, assuming $\Delta_t[2K_{p_0}e^{\eta_p b} + K_{d_0}e^{-\eta_d a}] + \gamma \le 1$ and $\|\mathbf{h}(0)\| \le 1$, we can estimate an upper bound for the radius as $\Delta_t\frac{s(b-a)}{4}C\|\mathbf{W}_h\|$, where $C = \eta_p K_{p_0}e^{\eta_p b} + \eta_d K_{d_0}e^{-\eta_d a} - \eta_p K_{p_0}e^{\eta_p a}$.

Proof. We write the Jacobian as $\mathbf{J}(t) = \gamma\mathbf{I} - \Delta_t\mathrm{diag}(K_p(\mathbf{z}(t+1)) + K_d(\mathbf{z}(t+1))) + \Delta_t\mathbf{M}(t)$. The eigenvalues of the diagonal matrix $\gamma\mathbf{I} - \Delta_t\mathrm{diag}(K_p(\mathbf{z}(t+1)) + K_d(\mathbf{z}(t+1)))$ are exactly the entries in the diagonal, i.e., the real values $\gamma - \Delta_t p_i(t)$. We can consider the dense matrix $\Delta_t\mathbf{M}(t)$ as a perturbative term of the diagonal matrix $\gamma\mathbf{I} - \Delta_t\mathrm{diag}(K_p(\mathbf{z}(t+1)) + K_d(\mathbf{z}(t+1)))$ and apply the Bauer–Fike theorem[57] to get the following estimation on the location of the eigenvalues $\chi(t)$ of $\mathbf{J}(t)$:

$$|\chi(t) - (\gamma - \Delta_t p_i(t))| \le \Delta_t\|\mathbf{M}(t)\|.$$

In other words, for each eigenvalue $\chi(t)$ of $\mathbf{J}(t)$, there exists an index i, such that $\chi(t)$ is at a distance at most $\Delta_t \|\mathbf{M}(t)\|$ from $\gamma - \Delta_t p_i(t)$. Moreover, by exploiting the upper bound of Eq. (15) on the norm of $\mathbf{M}(t)$, we can conclude the proof. □

APPENDIX E: PROOF OF THEOREM V.6

Theorem. In the case of a small step Δ_t or weak coupling (i.e., small $\|\mathbf{W}_h\|$), if the MF model has all eigenvalues inside the unit circle for any input, then

$$\begin{cases} \gamma - \Delta_t p_{\max} > -1, \\ \gamma - \Delta_t p_{\min} < 1, \end{cases}$$

where $p_{\min} := K_{p_0} e^{\eta_p a} + K_{d_0} e^{-\eta_d b}$ and $p_{\max} := K_{p_0} e^{\eta_p b} + K_{d_0} e^{-\eta_d a}$. Moreover, the following approximation of the spectral radius of the Jacobian holds:

$$\rho(\mathbf{J}(t)) \approx \max\left(|\gamma - \Delta_t p_{\max}|, \gamma - \Delta_t p_{\min}\right).$$

Proof. A necessary condition to have all eigenvalues inside the unit circle is to impose that the center points, $\gamma - \Delta_t p_i(t)$, lie inside the unit circle. Recalling that $p_i(t) = K_p\big((\mathbf{z}(t+1))_i\big) + K_d\big((\mathbf{z}(t+1))_i\big)$, we can bound the $p_i(t)$ locations within $p_{\min} = K_{p_0} e^{\eta_p a} + K_{d_0} e^{-\eta_d b} \le \min_{x \in (a,b), i=1,\ldots,N_h} p_i(t)$ and $p_{\max} = K_{p_0} e^{\eta_p b} + K_{d_0} e^{-\eta_d a} \ge \max_{x \in (a,b), i=1,\ldots,N_h} p_i(t)$. Therefore, if $\gamma - \Delta_t p_{\max} > -1$, then all the center points are greater than -1, and if $\gamma - \Delta_t p_{\min} < 1$, then all the center points are less than 1. Moreover, the greater absolute value between $\gamma - \Delta_t p_{\max}$ and $\gamma - \Delta_t p_{\min}$ represents an approximation of the spectral radius of the Jacobian for the case of weak coupling or small step Δ_t. □

APPENDIX F: DETAILS ON EXPERIMENTAL RESULTS

Here, we provide detailed information on the hyper-parameters selected for each model after performing a model selection through grid search. These details complement the analysis presented in Sec. VI and offer a comprehensive view of the configurations used in our experiments. Please refer to Table III(a)–(d) for the explored value ranges, while the final selected hyper-parameters for MF-ESN, MF-RNN, ESN, and RNN are reported in Tables V–VIII, respectively.

TABLE V. Selected hyper-parameters after model selection via grid search for MF-ESN. Reported values refer to input scaling (ω), bias scaling (β), spectral radius (ρ), stabilizer (γ), slope (s), and discretization time (Δ_t).

Task	ω	β	ρ	γ	s	Δ_t
JapaneseVowels(↑)	1	0.001	0.8	1	5	0.01
SyntheticControl(↑)	1	0.1	0.99	0.95	5	0.1
ECG5000(↑)	1	0	0.9	1	5	0.01
GunPoint(↑)	1	0	0.99	0.95	5	0.1
Wafer(↑)	1	0.001	0.95	1	5	0.1
Epilepsy(↑)	1	0.001	0.9	1	1	0.01
Coffee(↑)	10	1	0.9	0.95	5	0.01
FloodModeling1(↓)	0.1	0	0.8	1	5	0.01
FloodModeling2(↓)	0.1	1	0.95	1	5	0.01
FloodModeling3(↓)	0.1	0.001	0.95	1	5	0.01

TABLE VI. Selected hyper-parameters after model selection via grid search for MF-RNN. Reported values refer to learning rate, stabilizer (γ), slope (s), and discretization time (Δ_t).

Task	Learning rate	γ	s	Δ_t
JapaneseVowels(↑)	0.01	0.8	1	0.001
SyntheticControl(↑)	0.001	0.8	5	0.1
ECG5000(↑)	0.001	0.95	1	0.1
GunPoint(↑)	0.001	0.8	5	0.1
Wafer(↑)	0.001	0.8	5	0.01
Epilepsy(↑)	0.001	1	5	0.001
Coffee(↑)	0.001	0.8	5	0.001
FloodModeling1(↓)	0.001	1	5	0.001
FloodModeling2(↓)	0.001	1	1	0.001
FloodModeling3(↓)	0.001	1	1	0.001

TABLE VII. Selected hyper-parameters after model selection via grid search for ESN. Reported values refer to input scaling (ω), bias scaling (β), spectral radius (ρ), and leaking-rate (α).

Task	ω	β	ρ	α
JapaneseVowels(↑)	1	0.001	0.99	0.1
SyntheticControl(↑)	1	1	0.99	0.3
ECG5000(↑)	1	1	0.99	0.5
GunPoint(↑)	10	0.001	0.95	0.1
Wafer(↑)	1	1	0.99	0.1
Epilepsy(↑)	1	1	0.9	0.1
Coffee(↑)	0.1	0.1	0.99	0.3
FloodModeling1(↓)	1	0.1	0.99	0.1
FloodModeling2(↓)	2	0.1	0.95	0.1
FloodModeling3(↓)	1	0.1	0.99	0.1

TABLE VIII. Selected hyper-parameters after model selection via grid search for RNN. Reported values refer to the learning rate.

Task	Learning rate
JapaneseVowels(↑)	0.001
SyntheticControl(↑)	0.001
ECG5000(↑)	0.001
GunPoint(↑)	0.01
Wafer(↑)	0.001
Epilepsy(↑)	0.001
Coffee(↑)	0.001
FloodModeling1(↓)	0.001
FloodModeling2(↓)	0.001
FloodModeling3(↓)	0.001

REFERENCES

[1] D. V. Christensen, R. Dittmann, B. Linares-Barranco, A. Sebastian, M. Le Gallo, A. Redaelli, S. Slesazeck, T. Mikolajick, S. Spiga, S. Menzel *et al.*, "2022 roadmap on neuromorphic computing and engineering," Neuromorphic Comput. Eng. **2**, 022501 (2022).

[2] A. Mehonic and A. J. Kenyon, "Brain-inspired computing needs a master plan," Nature **604**, 255–260 (2022).

[3] H. Jaeger, B. Noheda, and W. G. Van Der Wiel, "Toward a formal theory for computing machines made out of whatever physics offers," Nat. Commun. **14**, 4911 (2023).

[4] C. Kaspar, B. J. Ravoo, W. G. van der Wiel, S. V. Wegner, and W. H. P. Pernice, "The rise of intelligent matter," Nature **594**, 345–355 (2021).

[5] D. Marković, A. Mizrahi, D. Querlioz, and J. Grollier, "Physics for neuromorphic computing," Nat. Rev. Phys. **2**, 499–510 (2020).

[6] M.-K. Song, J.-H. Kang, X. Zhang, W. Ji, A. Ascoli, I. Messaris, A. S. Demirkol, B. Dong, S. Aggarwal, W. Wan *et al.*, "Recent advances and future prospects for memristive materials, devices, and systems," ACS Nano **17**, 11994–12039 (2023).

[7] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: A review," Neural Networks **115**, 100–123 (2019).

[8] K. Nakajima, "Physical reservoir computing—An introductory perspective," Jpn. J. Appl. Phys. **59**, 060501 (2020).

[9] M. Yan, C. Huang, P. Bienstman, P. Tino, W. Lin, and J. Sun, "Emerging opportunities and challenges for the future of reservoir computing," Nat. Commun. **15**, 2056 (2024).

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

[11] C. Gallicchio, A. Micheli, and L. Pedrelli, "Deep reservoir computing: A critical experimental analysis," Neurocomputing **268**, 87–99 (2017).

[12] L. Grigoryeva and J.-P. Ortega, "Echo state networks are universal," Neural Networks **108**, 495–508 (2018).

[13] C. Du, F. Cai, M. A. Zidan, W. Ma, S. H. Lee, and W. D. Lu, "Reservoir computing using dynamic memristors for temporal information processing," Nat. Commun. **8**, 2204 (2017).

[14] R. Midya, Z. Wang, S. Asapu, X. Zhang, M. Rao, W. Song, Y. Zhuo, N. Upadhyay, Q. Xia, and J. J. Yang, "Reservoir computing using diffusive memristors," Advanced Intelligent Systems **1**, 1900084 (2019).

[15] Y. Zhong, J. Tang, X. Li, B. Gao, H. Qian, and H. Wu, "Dynamic memristor-based reservoir computing for high-efficiency temporal signal processing," Nat. Commun. **12**, 408 (2021).

[16] J. Moon, W. Ma, J. H. Shin, F. Cai, C. Du, S. H. Lee, and W. D. Lu, "Temporal data classification and forecasting using a memristor-based reservoir computing system," Nat. Electron. **2**, 480–487 (2019).

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

[18] G. Milano, K. Montano, and C. Ricciardi, "In materia implementation strategies of physical reservoir computing with memristive nanonetworks," J. Phys. D: Appl. Phys. **56**, 084005 (2023).

[19] R. Zhu, S. Lilak, A. Loeffler, J. Lizier, A. Stieg, J. Gimzewski, and Z. Kuncic, "Online dynamical learning and sequence memory with neuromorphic nanowire networks," Nat. Commun. **14**, 6697 (2023).

[20] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," Nature **453**, 80–83 (2008).

[21] L. Chua, "Memristor-the missing circuit element," IEEE Trans. Circuit Theory **18**, 507–519 (1971).

[22] R. Yang, H.-M. Huang, and X. Guo, "Memristive synapses and neurons for bioinspired computing," Adv. Electron. Mater. **5**, 1900287 (2019).

[23] J. Tang, F. Yuan, X. Shen, Z. Wang, M. Rao, Y. He, Y. Sun, X. Li, W. Zhang, Y. Li *et al.*, "Bridging biological and artificial neural networks with emerging neuromorphic devices: Fundamentals, progress, and challenges," Adv. Mater. **31**, 1902761 (2019).

[24] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," Nat. Nanotechnol. **8**, 13–24 (2013).

[25] Z. Wang, H. Wu, G. W. Burr, C. S. Hwang, K. L. Wang, Q. Xia, and J. J. Yang, "Resistive switching materials for information processing," Nat. Rev. Mater. **5**, 173–195 (2020).

[26] M. Nelson and J. Rinzel, "The Hodgkin-Huxley model," in *The Book of Genesis*, 2 (Springer, 1995).

[27] Z. Wang, S. Joshi, S. E. Savel'ev, H. Jiang, R. Midya, P. Lin, M. Hu, N. Ge, J. P. Strachan, Z. Li *et al.*, "Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing," Nat. Mater. **16**, 101–108 (2017).

[28] G. Milano, M. Luebben, Z. Ma, R. Dunin-Borkowski, L. Boarino, C. F. Pirri, R. Waser, C. Ricciardi, and I. Valov, "Self-limited single nanowire systems combining all-in-one memristive and neuromorphic functionalities," Nat. Commun. **9**, 5151 (2018).

[29] W. Wang, M. Wang, E. Ambrosi, A. Bricalli, M. Laudato, Z. Sun, X. Chen, and D. Ielmini, "Surface diffusion-limited lifetime of silver and copper nanofilaments in resistive switching devices," Nat. Commun. **10**, 81 (2019).

[30] E. Miranda, G. Milano, and C. Ricciardi, "Modeling of short-term synaptic plasticity effects in zno nanowire-based memristors using a potentiation-depression rate balance equation," IEEE Trans. Nanotechnol. **19**, 609–612 (2020).

[31] G. Milano, E. Miranda, and C. Ricciardi, "Connectome of memristive nanowire networks through graph theory," Neural Networks **150**, 137–148 (2022).

[32] G. Milano, M. Agliuzza, N. De Leo, and C. Ricciardi, "Speech recognition through physical reservoir computing with neuromorphic nanowire networks," in *2022 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2022), pp. 1–6.

[33] G. Milano, T. Chakrabarty, and C. Ricciardi, "Mackey-glass time series forecasting by nanowire networks," in *2023 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE)* (IEEE, 2023), pp. 989–994.

[34] D. Cipollini and L. R. B. Schomaker, "Conduction and entropy analysis of a mixed memristor-resistor model for neuromorphic networks," Neuromorphic Comput. Eng. **3**, 034001 (2023).

[35] D. Cipollini, H. Greatorex, M. Mastella, E. Chicca, and L. Schomaker, "FusedMemBrain: A spiking processor combining CMOS and self-assembled memristive networks," arXiv:2411.19353 (2024).

[36] F. Ferrarese Lupi, G. Milano, A. Angelini, M. Rosero-Realpe, B. Torre, E. Kozma, C. Martella, and C. Grazianetti, "Synaptic plasticity and visual memory in a neuromorphic 2d memitter based on WS_2 monolayers," Adv. Funct. Mater. **34**, 2403158 (2024).

[37] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," Int. J. Uncertainty, Fuzziness Knowl.-Based Syst. **6**, 107–116 (1998).

[38] K. Nakajima and I. Fischer, *Reservoir Computing* (Springer, 2021).

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

[40] I. B. Yildiz, H. Jaeger, and S. J. Kiebel, "Re-visiting the echo state property," Neural Networks **35**, 1–9 (2012).

[41] X. Liang, J. Tang, Y. Zhong, B. Gao, H. Qian, and H. Wu, "Physical reservoir computing with emerging electronics," Nat. Electron. **7**, 193–206 (2024).

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

[43] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks-with an erratum note," GMD Technical Report (German National Research Center for Information Technology, Bonn, Germany, 2001), Vol. **148**, p. 13.

[44] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, "The UCR time series archive," IEEE/CAA J. Autom. Sin. **6**, 1293–1305 (2019).

[45] C. W. Tan, C. Bergmeir, F. Petitjean, and G. I. Webb, "Time series extrinsic regression," Data Mining Knowl. Discovery **35**, 1032–1060 (2021).

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

[47]C. Gallicchio, A. Micheli, and L. Pedrelli, "Fast spectral radius initialization for recurrent neural networks," in *Recent Advances in Big Data and Deep Learning: Proceedings of the INNS Big Data and Deep Learning Conference INNSBDDL2019, held at Sestri Levante, Genova, Italy, 16–18 April 2019* (Springer, 2020), pp. 380–390.
[48]T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," arXiv:1902.09574 (2019).
[49]J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations* (2019), see https://openreview.net/forum?id=rJl-b3RcF7.
[50]V. Ramanujan, M. Wortsman, A. Kembhavi, A. Farhadi, and M. Rastegari, "What's hidden in a randomly weighted neural network?," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, 2020), pp. 11893–11902.
[51]C. Gallicchio, "Sparsity in reservoir computing neural networks," in *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)* (IEEE, 2020), pp. 1–7.
[52]B. Li, R. S. Fong, and P. Tino, "Simple cycle reservoirs are universal," J. Mach. Learn. Res. **25**, 1–28 (2024).
[53]R. Zhu, J. Eshraghian, and Z. Kuncic, "Memristive reservoirs learn to learn," in *Proceedings of the 2023 International Conference on Neuromorphic Systems* (ACM, 2023), pp. 1–7.
[54]A. Subramoney, F. Scherr, and W. Maass, "Reservoirs learn to learn," in *Reservoir Computing: Theory, Physical Implementations, and Applications* (Springer Nature, 2021), pp. 59–76.
[55]M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in Neural Information Processing Systems*, 29 (Curran Associates, Red Hook, NY, 2016).
[56]Q. Xia and J. J. Yang, "Memristive crossbar arrays for brain-inspired computing," Nat. Mater. **18**, 309–323 (2019).
[57]F. L. Bauer and C. T. Fike, "Norms and exclusion theorems," Numer. Math. **2**, 137–141 (1960).