Complex Recurrent Spectral Network

Chicchi, Lorenzo; Giambagli, Lorenzo; Buffoni, Lorenzo; Marino, Raffaele; Fanelli, Duccio

Published in:
Chaos, Solitons & Fractals

DOI:
10.1016/j.chaos.2024.114998

Publication date:
2024

Document Version
Publisher's PDF, also known as Version of record

Chaos, Solitons and Fractals 184 (2024) 114998

Contents lists available at ScienceDirect

Chaos, Solitons and Fractals

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

Complex Recurrent Spectral Network

Lorenzo Chicchi [a,*], Lorenzo Giambagli [a,b], Lorenzo Buffoni [a], Raffaele Marino [a], Duccio Fanelli [a]

[a] Department of Physics and Astronomy, University of Florence, INFN and CSDC, Sesto Fiorentino, Italy
[b] naXys - Namur Center for Complex Systems, University of Namur, rue Grafé 2, 5000 Namur, Belgium

ARTICLE INFO

Keywords:
Neural networks
Dynamical systems
Discrete maps
Machine learning
Recurrent Networks
Attractors

ABSTRACT

This paper presents a novel approach to advancing artificial intelligence (AI) through the development of the Complex Recurrent Spectral Network (ℂ-RSN), an innovative variant of the Recurrent Spectral Network (RSN) model. The ℂ-RSN model introduces localized non-linearity, complex fixed eigenvalues, and a distinct separation of memory and input processing functionalities. These features enable the ℂ-RSN to evolve towards a dynamic, oscillating final state that bear some degree of similarity with biological cognition. The model's ability to classify data through a time-dependent function, and the localization of information processing, is demonstrated by using the MNIST dataset. Remarkably, distinct items supplied as a sequential input yield patterns in time which bear the indirect imprint of the insertion order (and of the separation in time between contiguous insertions).

1. Introduction

Artificial intelligence (AI) stands as a cornerstone of innovation and progress. The surge in AI research and its applications across various domains, from healthcare [1,2] and finance [3] to autonomous systems [4,5] and beyond [6–8], underscores its growing importance. The quest is to replicate, possibly enhance, human cognitive abilities through computational means [9,10]. AI's significance is particularly evident in its ability to process vast amounts of data, uncover patterns [11,12], and make decisions, often surpassing human capabilities in speed and efficiency [13].

Central to the evolution of AI are neural networks [14], which draw inspiration from the biological circuits [15] that constitute animal brains. These networks are systems of interconnected nodes, or neurons, that work in concert to solve complex problems. Among these, Recurrent Neural Networks (RNNs) [16–18] have been proven effective in handling sequential data. Unlike traditional neural networks, RNNs possess an inherent memory kernel, as an additional degree of freedom. This allows RNNs to process not just individual data points, but entire sequences of data, making them particularly adept at tasks like language processing [19], time series analysis [20], and even music composition [21,22].

Despite the impressive performance displayed, a significant gap remains in our understanding of the actual functioning of AI, Machine Learning (ML), and Deep Learning (DL) technologies [23]. This opacity often limits their broader application, especially in scenarios demanding transparency and reliability [24,25]. Interpretability [26,27] represents still a critical challenge in the field. To shed novel light on AI intricacies, different theoretical frameworks have been proposed that seek at tackling the problem from distinct, though complementary directions [28–36].

In this paper, and to contribute along these lines, we delve deeper into the concept of Recurrent Neural Networks (RNNs) and, more specifically, expand upon the variant of the Recurrent Spectral Methods (RSN), as introduced in [18]. In this latter paper, a novel machine learning strategy was proposed wherein the evaluation process follows the dynamics of a specifically engineered system. The spectral parametrization of the adjacency matrix within a fully connected network allows to effectively steer the system's evolution towards a pre-defined vector subspace, spanned by a subset of select eigenvectors. During the learning phase, parameters are self-consistently chosen so as to ensure that the system converges to a final stationary state, which points to the class the provided input data belongs to. This methodology has revealed several notable properties. A striking observation is the independence of the evaluation process from the number of steps (or the system's integration time) as employed during training. This results in a robust convergence of accuracy (or measured loss) to a stable asymptotic value, which remains consistent across successive iterations, beyond the chosen training's temporal horizon. This characteristic markedly contrasts with traditional Recurrent Neural Networks (RNNs), where the iteration count is a critical parameter for accurately categorizing test data.

The RSN is constructed so as to asymptotically align along specific stationary directions, which collectively define the underlying stable

* Corresponding author.
E-mail address: lorenzo.chicchi@unifi.it (L. Chicchi).

https://doi.org/10.1016/j.chaos.2024.114998
Received 12 December 2023; Received in revised form 23 April 2024; Accepted 10 May 2024
Available online 22 May 2024

manifold. As an alternative possibility, we will here explore the setting where the system under exam is made to evolve towards a time dependent final state, which is localized on just one (in the actual example hereafter provided) inspection node. The temporal embedding of the exit node contains sufficient information to yield an accurate classification of the supplied items. More concretely, static inputs will be provided as an input to a time discrete map, each iteration in time being associated to a transfer between adjacent layers within the feedforward arrangement of the network. The produced output will be a time evolving function that we shall use for pattern recognition (via proper embedding within the loss). Hence, we shall use a dynamical output to classify static input, rather then classifying dynamical input as often done with the RSN realm.

In doing so, we seek at unlocking new horizons in the application of machine learning, via time-sensitive outputs that are some extent reminiscent of the degree of plasticity as displayed by neurons in real biological systems. The proposed algorithm is termed Complex Recurrent Spectral Network (ℂ-RSN), for reasons that will become transparent in the following.

Building on the foundation of the RSN model, the Complex Recurrent Spectral Network (ℂ-RSN) introduces several key innovations, each serving a distinct role in the overall functionality:

- **Localized Non-linearity in a Subset of Nodes**: In the ℂ-RSN model, non-linearity is not uniformly distributed across the entire network. Instead, it is strategically localized within a specific subset of nodes, referred to as the *non-linear part*.
- **Complex Fixed Eigenvalues**: Incorporation of complex fixed eigenvalues of the form $e^{2\pi i/T_m}$. This complex eigenvalue structure imbues the network with a rhythmic, oscillatory dynamic to some extent reminiscent of the cyclical processes observed in biological neural activities [37,38]. This feature allows for the generated output to be condensed in a sufficiently expressive function of time, localized on just one nodes.
- **Memory Confined in the Linear Part**: In the ℂ-RSN architecture, memory functionality is associated to the linear part of the network. This segregation of memory into a dedicated linear subsystem ensures a more stable retention of information. The system is also able to handle sequential inputs by keeping track of the relative insertion order, as we shall demonstrate in the following.
- **Input Processing in the Non-Linear Part**: The design of the ℂ-RSN ensures that the input is primarily processed in the non-linear part of the network (the non-linear filters being localized on the nodes that are matching the input pixels).

These additional features make the ℂ-RSN model to transcend the limitations of the static RSN scenario, offering a more dynamic framework. The integration of localized nonlinearity, complex eigenvalues, dedicated memory storage, and specialized input processing marks a step forward in the development of a class of neural network models which possess an inherent degree of dynamical complexity, as that displayed by actual biological neural processes.

As we will clarify, the oscillatory nature of the final state is intricately linked to the complex eigenvalues, specifically those of the form $e^{2\pi i/T_m}$. As such, these eigenvalues play a crucial role in defining the ensuing temporal dynamics. The final state emerges as a linear superposition of eigenvectors associated with these latter eigenvalues, leading to a periodic behavior where the period is contingent upon the values of the constants T_m. To state it differently, the emergent wavefront at the exit node reflects the specific combination of eigenvectors that remain asymptotically active. The ability to represent the system's state as a function of time through a combination of time frequencies, on just one definite spatial location, yields an efficient embedding for the data to be eventually classified. In practical terms, by initiating the network with an activity corresponding to a specific dataset, the network is tasked with generating a time-dependent function $f_C(t)$, which is defined by the activity within a sub-portion of the network. This function effectively becomes a temporal signature of the classified data. As a byproduct, ℂ-RSN possesses the capability to sequentially process multiple sets of input: the generated wavefront reflect indeed the order and time of relative insertion of distinct items.

As remarked, a pivotal distinction from the RSN model is the localization of information storage. In the ℂ-RSN, the information acquired during the evaluation process is concentrated within a minimal portion of the network, specifically within a single neuron. This allows the rest of the network to remain unencumbered, i.e. ready to engage in other evaluation processes, with an effective division of labor which emulate biological cognition and processing [39].

The paper is structured as follows: In Section 2, we elaborate on the architecture of the Neural Network and delve into the intricacies of the learning process. Section 3 is dedicated to describing the forward evolution of the map underlying the ℂ-RSN. Our numerical results, specifically pertaining to the MNIST dataset, are presented in Section 4. The manuscript concludes with Section 5, where we summarize our results and discuss potential avenues for future research.

2. ℂ-RSN neural network architecture

In this section, we conceptualize a neural network as a discrete map, thereby establishing the mathematical underpinnings of the ℂ-RSN. We represent the neural activity by a vector $\vec{x} \in \mathbb{C}^N$. Simultaneously, the network's architecture is modeled by a weighted adjacency matrix, $\mathbf{W} \in \mathbb{C}^{N\times N}$, characterizing a fully connected neural network of order N, with complex inter-linked connections. It is pertinent to recall from graph theory that the order of a graph refers to its number of nodes, while the size denotes the number of edges.

The architecture includes two neuron types: *linear* and *non-linear*. With linear we mean that on such neurons an activation function acts linearly, while with non-linear we mean that on such neurons an activation function acts non-linearly. To this end, we categorize the neuron sets as $\mathcal{NL} = \{1, 2, \dots, L-1, L\}$ (with cardinality $|\mathcal{NL}| = L$) and $\mathcal{L} = \{L+1, L+2, \dots, N-1, N\}$ (with cardinality $|\mathcal{L}| = N-L$), corresponding to the *non-linear* and *linear* segments of the network, respectively.

The neural network, in its entirety, is characterized by a singular activation function $\tilde{f}$, which assumes a non-linear form for $1 \le i \le L$ and transitions to a linear profile for indices beyond L. For the non-linearity within $\tilde{f}$, we opt for a sigmoidal structure, specifically a tanh function. This design choice is elucidated through the following mathematical expression:

$$\tilde{f}(z_l) = \begin{cases} \tanh(z_l) & \text{if} \quad l \le L, \\ z_l & \text{if} \quad l > L, \end{cases} \tag{1}$$

where $\vec{z}$ is a generic activity vector.

In our model, we postulate that the matrix $\mathbf{W}$, belonging to the complex space $\mathbb{C}^{N\times N}$, is derived through its spectral decomposition:

$$\mathbf{W} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^{-1} \tag{2}$$

Here, Λ is a diagonal matrix composed of eigenvalues λ_l, and $\boldsymbol{\Phi}$ encompasses the corresponding eigenvectors. Essentially, $\boldsymbol{\Phi}$ represents the basis for this decomposition. The choice of dealing with the above decomposition of the coupling matrix follows the spectral approach to machine learning discussed in [18,40–43]. When it come to the eigenvalues λ_l we posit:

$$\lambda_l = \begin{cases} e^{2\pi i/T_l} & \text{if} \quad l < M, \\ \lambda_l \in \mathbb{R} \quad \text{where} \quad |\lambda_l| < 1 & \text{if} \quad l > M. \end{cases} \tag{3}$$

This formulation entails that the first M, with $M \le L$, eigenvalues are predetermined to be complex values. Their magnitudes are set to equal 1, with a phase contingent on the parameter T_l, termed the *period*. This specification imbues the model with a rhythmic, cyclical

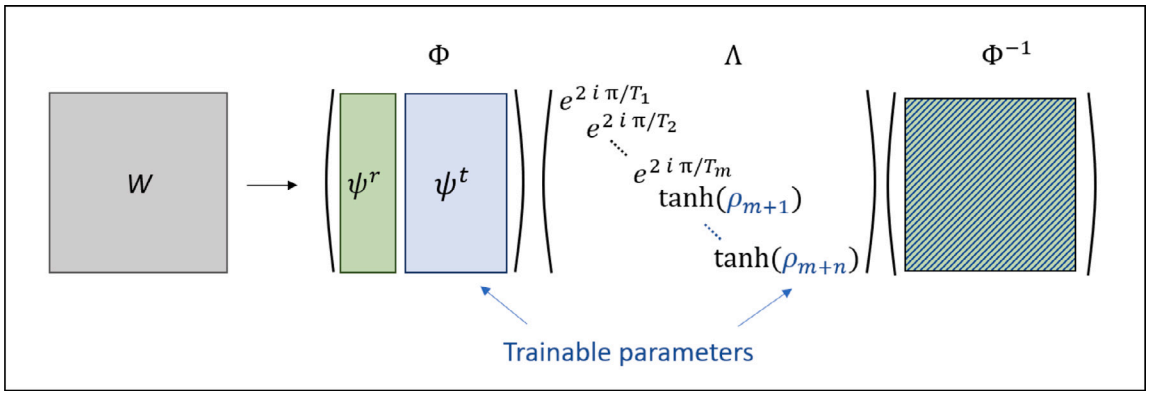

Fig. 1. Spectral decomposition used to describe the linear transformation. The trainable parameters are highlighted in blue and contained in the set ψ^t. The fixed parameters are highlighted in green and contained in the set ψ^r. The third matrix is the inverse matrix of the basis Φ and depend to both trainable elements and fixed elements of the matrix Φ. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

dynamism, reflecting in the oscillatory behavior of these eigenvalues. Conversely, the second set of eigenvalues (to be trained upon optimization) is constrained to have magnitudes less than 1, ensuring a damping effect that stabilizes the network over time. This condition is technically implemented by filtering with $\tanh(\cdot)$ the trained parameters. It is important to note that, despite the use of complex phases, this approach is completely different from the Complex-Valued Neural Networks in [44], as our system involves a dynamical evolution instead of static network.

To provide a more intuitive grasp, Fig. 1 offers a schematic visual depiction of the spectral decomposition, as applied in our study. This diagrammatic representation aids in comprehending how the eigenvalues are spatially and numerically orchestrated within the matrix, highlighting the dichotomy nature of their arrangement and the distinct roles they play in the network's functionality.

To clarify, the columns of the matrix Φ correspond to the eigenvectors $\vec{\psi}^{(k)}$ of the decomposition, where k ranges from 1 to N. Specifically, the first M eigenvectors, denoted as $m = 1, \dots, M$, are fixed and primarily localized within the linear part of the network. In Fig. 1, this particular subset of M eigenvectors is identified by the label ψ^r. The remaining $N - M$ eigenvectors, encapsulated collectively under the set ψ^t, are distinct from the first M and have different characteristics or localization within the network.

When $m < M$, each component of an eigenvector $\vec{\psi}^{(m)}$ is set to zero, with the exception of the Nth and the $(N - m)$th components, which are set to one. In other words, for a given eigenvector $\vec{\psi}^{(m)}$, only two components are set to one: the last component and the $(N - m)$th component, while the others are set to zero. For example, for the first three values of m, with N and M fixed, one should have:

$$\vec{\psi}^{(m=1)} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \quad \vec{\psi}^{(m=2)} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \quad \vec{\psi}^{(m=3)} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{4}$$

These eigenvectors, i.e the ones in ψ^r, and the respective eigenvalues, are fixed and, therefore, not learnable parameters during the training process. The ones in ψ^t, with the respective eigenvalues of the form:

$$\lambda_l = \tanh(\rho_l) \qquad l = M + 1, \dots, N, \qquad \rho_l \in \mathbb{R}. \tag{5}$$

are, instead, parameters that can be learned during the training process. The functional choice of the trainable eigenvalues enables for the constraints on the eigenvalues' magnitude to be satisfied during every single step of the training procedure. During training, the values of λ_l get hence modified by acting on the parameters ρ_l. The non linear filter $\tanh(\cdot)$ makes sure that λ_l is indeed confined within the assigned interval of pertinence. In conclusion, the Complex Recurrent Spectral Network ($\mathbb{C}$-RSN) is defined as the discrete map:

$$\vec{x}_{t+1} = \tilde{f}(\boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^{-1}\vec{x}_t). \tag{6}$$

The above system will be trained by employing a suitably defined loss that pivots on the real part of the time dependent signal produced at the exit node. We will elaborate further on this point in the following section.

3. Forward evolution of the map

Here, we focus on the forward evolution of the map within our model, i.e, Eq. (6). Let us begin by defining $\vec{x}_0$ as the initial condition for our map. After t iterations of the discrete map, we obtain a vector $\vec{x}_t$.

For the purposes of this analysis and to give qualitative hints on the processing flow, we will solely focus on the linear dynamics. To this end we will deliberately neglect the contributions that stem from the imposed (and spatially localized) non-linearities. This assumption simplifies our understanding of the system's behavior, thus allowing for analytical progress to be made.

Under linear dynamics and given the specific constraints on the eigenvalues described in the previous section, we observe that the long-term behavior of the system, i.e., its asymptotic dynamics, is confined to a subspace. This subspace is defined by the first M eigenvectors of the system. In principle, non-linearities could hold the potential to disrupt the system's convergence towards its expected asymptotic manifold — the subspace spanned by the eigenvectors linked to the fixed (and complex) eigenvalues. Interestingly, our investigations reveal that such deviations from the expected behavior — which we might term *divergent behaviors* somewhat loosely — are rare even for a system that did not undergo training and that has been initialized at random. Therefore, in our further discussions, we will assume, for the sake of the argument, that the dynamics of our system steadily progress, upon training, towards the final subspace defined by the fixed eigenvectors.

Let us consider a scenario where, after t^* iterations, the dynamics of the map (6) becomes confined within the subspace formed by the first M eigenvectors. This situation can be mathematically expressed as follows:

$$\vec{x}_{t^*} = \sum_{m=1}^{M} \alpha_m \vec{\psi}^{(m)}. \tag{7}$$

Here, $\vec{x}_{t^*}$ represents the state of the system after t^* iterations. The expression on the right-hand side is a linear combination of the first M eigenvectors $\vec{\psi}^{(m)}$, where each eigenvector is scaled by a coefficient

α_m. In other words, we are expressing $\vec{x}_{t^*}$ into a basis composed by first M eigenvectors. Remark that the effect of the non linearities is structurally embedded in the aforementioned coefficients. To carry out the analysis, there is no need to explicitly compute the coefficients α_m, that will in general reflect the specific items to be processed, as well as the characteristics of the employed non-linearities.

This formulation aligns with our earlier discussion that the system, influenced by its inherent constraints and the learning process, naturally gravitates towards this particular subspace. The coefficients α_m in the equation are indicative of the extent to which each of the first M eigenvectors influences the state of the system at the iteration t^*.

From (4) and (1), the dynamics on the subspace is linear and the activity remains confined on it. In particular after a new application of the map (6), the activity becomes:

$$\vec{x}_{t^*+1} = \sum_{m=1}^{M} \alpha_m e^{2\pi i/T_m} \vec{\psi}^{(m)}. \tag{8}$$

Making more iterations, the above equation transforms into:

$$\vec{x}_{t^*+t} = \sum_{m=1}^{M} \alpha_m (e^{2\pi i/T_m})^t \vec{\psi}^{(m)} = \sum_{m=1}^{M} \alpha_m e^{2\pi i t/T_m} \vec{\psi}^{(m)}. \tag{9}$$

Next, we turn our focus to the coefficients α_m in the system's dynamics. These coefficients are typically complex numbers, a result of dealing with complex eigenvalues. Moreover, and as already remarked, they are functions of the training parameters, which means their values are influenced by these parameters. To bring this into evidence, we make explicit the dependence of α_m on the vector of trainable parameters, here denoted as $\vec{\beta}$, as :

$$\alpha_m(\vec{\beta}) = r(m, \vec{x}_0, \vec{\beta}) + \mathrm{i}c(m, \vec{x}_0, \vec{\beta}), \tag{10}$$

where $r(m, \vec{x}_0, \vec{\beta})$ and $c(m, \vec{x}_0, \vec{\beta})$ are two real scalar functions where the dependence on the initial condition $\vec{x}_0$ has been brought into evidence. Under this setting, Eq. (9) becomes:

$$\begin{aligned}\vec{x}_{t^*+t} &= \sum_{m=1}^{M} \alpha_m(\vec{\beta}) e^{2\pi i t/T_m} \vec{\psi}^{(m)} = \sum_{m=1}^{M} (r(m, \vec{x}_0, \vec{\beta}) + \mathrm{i}c(m, \vec{x}_0, \vec{\beta})) e^{2\pi i t/T_m} \vec{\psi}^{(m)} \\ &= \sum_{m=1}^{M} \left(r(m, \vec{x}_0, \vec{\beta}) + \mathrm{i}c(m, \vec{x}_0, \vec{\beta})\right) \left(\cos\left(\frac{2\pi t}{T_m}\right) + \mathrm{i}\sin\left(\frac{2\pi t}{T_m}\right)\right) \vec{\psi}^{(m)}.\end{aligned} \tag{11}$$

From the form of eigenvalues in (4), and recalling that $(\vec{\psi}^{(m)})^N = 1$ $\forall m = 1, \dots, M$, we can explicitly compute the Nth component of $\vec{x}_{t^*+t}$:

$$\begin{aligned}\left(\vec{x}_{t^*+t}\right)_N = \sum_{m=1}^{M} &\left(r(m, \vec{x}_0, \vec{\beta}) \cos\left(\frac{2\pi t}{T_m}\right) - c(m, \vec{x}_0, \vec{\beta}) \sin\left(\frac{2\pi t}{T_m}\right)\right) \\ &+ \mathrm{i}\left(c(m, x_0, \vec{\beta}) \cos\left(\frac{2\pi t}{T_m}\right) + r(m, \vec{x}_0, \vec{\beta}) \sin\left(\frac{2\pi t}{T_m}\right)\right),\end{aligned} \tag{12}$$

and taking the real part of the above equation, we end up with:

$$\begin{aligned}\mathcal{R}(t, \vec{x}_0, \vec{\beta}) = \mathrm{Re}\left((\vec{x}_{t^*+t})_N\right) = \sum_{m=1}^{M} &\left(r(m, \vec{x}_0, \vec{\beta}) \cos\left(\frac{2\pi t}{T_m}\right)\right. \\ &\left. - c(m, \vec{x}_0, \vec{\beta}) \sin\left(\frac{2\pi t}{T_m}\right)\right).\end{aligned} \tag{13}$$

This latter Eq. (13) plays a pivotal role in defining the loss function for our model, as we will discuss in the following. In the context of our neural network model, the real part of the complex state vector, $\vec{x}_{t^*+t}$, encapsulates critical information about the system's dynamics. By integrating the real part into the loss function, we align the training process with the goal of optimizing the network's performance based on tangible, observable outcomes.

Expanding upon this foundation, we apply our model to classification tasks. Our dataset D consists of pairs $(\vec{x}_0, \hat{y})^\eta$ for $\eta = 1, \dots, |D|$. Here, $\vec{x}_0$ serves as both the input vector and the initial condition for our discrete map, while $\hat{y}$ represents the label, ranging from 1 to C, corresponding to different classes.

For classification purposes, we define C discrete time functions, $f_C(t)$, each associated with a distinct class. The classification challenge involves training the network to minimize a loss function, formulated as:

$$\mathcal{L} = \sum_{\eta \in D} \mathcal{L}(\vec{x}_0^{\,\eta}, \vec{\beta}) \tag{14}$$

where

$$\mathcal{L}(\vec{x}_0^{\,\eta}, \vec{\beta}) = \sum_{t=0}^{T} (\mathcal{R}(t, \vec{x}_0^{\,\eta}, \vec{\beta}) - f_{\hat{y}^\eta}(t))^2. \tag{15}$$

As noticed, the loss function hinges on the real part of the network's output, $\mathcal{R}(t, \vec{x}_0, \vec{\beta})$, at time t, with the aim of minimizing it through the adjustment of $\vec{\beta}$. The network is thus trained to align the real part of its output with the target time function $f_{\hat{y}}(t)$ of the corresponding class $\hat{y}$, over a time span T, starting from the input $\vec{x}_0$. We remark that the target functions $f_{\hat{y}}(t)$ are set arbitrarily. Indeed, the results obtained in terms of measured performance and for the class of experiments carried out, are not affected by the specific form of the employed embedding function $f_{\hat{y}}(t)$. The only constraint is that it should be possible to decompose this latter function on the basis of eigenvectors associated to the fixed eigenvalues, with a sufficiently small approximation error. In future perspective, we aim at exploring the possibility of autonomously selecting the best target functions during the training stage.

The minimization of the loss function is accomplished through the well-known Adam optimization algorithm [45] propagating the gradient backward in time (BPTT) [46].

In the following subsections, we will delve into the results obtained from applying this model to the renowned MNIST dataset [47]. The MNIST dataset serves as an excellent platform to demonstrate our model's capability in classifying different items, based on their produced temporal dynamics.

4. Results

The MNIST dataset, an acronym for *Modified National Institute of Standards and Technology*, is a renowned collection of handwritten digits. It comprises 60 000 training images and 10 000 testing images, each of size 28 × 28 pixels, making it a fundamental resource for training and testing in machine learning and image recognition domains. Over the years, it has emerged as a benchmark standard for research in image classification. The dataset is organized into ten classes, each corresponding to the digits 0 through 9. Every image in the dataset is associated with a label that indicates the class of the image, i.e., the digit it represents.

In our study, we trained a $\mathbb{C}$-RSN (Complex Recurrent Spectral Network) of size $N = 1000$. The non-linearity, as defined in (1), is applied to the first $L = 800$ neurons. We set the number of fixed eigenvectors and eigenvalues at $M = 5$. The model was trained to replicate the corresponding target discrete time function within a $T = 20$ step time window, during which the loss is computed. Prior to this window, the network is allowed to evolve for $t' = 10$ time steps, starting from an initial condition shaped by the selected image[1]

Each image from the MNIST dataset is normalized to ensure the pixel values range between 0 and 1. This normalized data is then

[1] Notice that the size of the underlying network is somehow dictated by the characteristics of the supplied input. $N = 1000$ is a reasonable choice, as it allows for each pixel (784 for the images to be classified) to be associated with an individual input node, while still granting for an additional reservoir of nodes to potentiate the computational capabilities of the system. All parameters have been however tuned over a limited range that includes their reference values, yielding results that are in line with those reported in the paper for the specific selection operated.

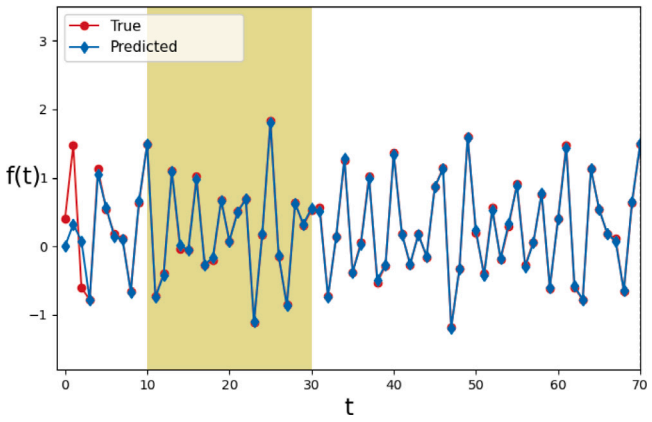

Fig. 2. Comparison of the predicted activity in the last neuron (blue line) against the target temporal function (red line) for an input corresponding to the class 1. The shaded yellow region indicates the time window where the loss is computed.

input to a selected subset of nodes in the network, particularly those incorporating non-linear processing units. During the initial ten steps, the network executes the classification task by diverging the dynamics that result from initial conditions belonging to different classes into their respective final states.

Continuing this analysis, Fig. 2 showcases the network's performance. The discrete time function $f(t)$ output by the last neuron is depicted in blue, while the red line represents the target discrete time function for the input class. Remarkably, the two curves closely align, indicating that the model has successfully learned to reproduce the target function. Notably, this alignment persists even beyond the time window where the loss is calculated, as highlighted in yellow in the figure. Furthermore, the convergence of the curves begins even before this designated time window. These observations are consistent with findings from the RSN model presented in [18], particularly the sustained proximity of the curves across successive time steps, which is a direct consequence of the stability of the final subspace.

To quantify the model's effectiveness, we assess its accuracy by determining, for each input, which of the potential target functions (based on L^2 distance) is closest to the function observed in the last neuron. Applying this method to the MNIST dataset, our model achieved an accuracy of about 0.98 on the test set, in line with what one gets with a standard Multi-Layer Perceptron (MLP) model with a ReLu activation function. The close proximity of our model's accuracy to that of the MLP underscores its efficacy.

4.1. The role of the basis in signal reconstruction

Eq. (13) elucidates that the signal observed in the last neuron is a linear combination of sinusoids, each defined by a distinct period T_m. It is important to note that each of the fixed eigenvectors in our model is sparse, with only two non-zero elements — one consistently at position N, and the other at a unique position corresponding to the eigenvector in question. The signal at these non-zero positions is, by design, a pure sinusoidal function, the period of which mirrors the T_m value specified in the definition of its associated eigenvalue. Consequently, the amplitude of these sinusoidal waves provides an indirect metric for assessing the prominence of each mode within the time-resolved patterns observed at node N.

The parameters T_m, introduced in (3), effectively filter elements from the Fourier basis, crafting a bespoke basis that the network employs to construct the signal in the last neuron. This process is displayed in Fig. 3, which presents a visual representation of the network during two distinct dynamic phases: the onset of the activity and a subsequent moment within the finite window where the loss is computed.

Panel **A** of Fig. 3 captures the network at the initial condition, showcasing activity initiation within the non-linear segment. As the dynamics progress, this activity transitions towards the linear portion of the network. By the time we observe Panel **B**, the network has iterated sufficiently for the activity to be fully contained within the linear section. Here, the real part of the activity in the last neuron is seen to closely resemble the target time function. Meanwhile, the neurons associated with the fixed eigenvectors display the anticipated sinusoidal behavior.

This visualization confirms on the one hands the theoretical underpinnings of our model. On the other, it provides evidence on the way the activity is channeled through the network, for an accurate reproduction of the desired signal.

4.2. Multiple evaluations

As anticipated, $\mathbb{C}$-RSN networks can sequentially process multiple sets of input data, while ensuring that the dynamics associated with subsequent inputs remain unaffected by the preceding ones. This independent processing is achievable, provided that each new data input is introduced after a duration sufficient to allow the system's dynamics from the initial data to converge into the stable subspace. We elucidate in this subsection that, under this precondition, the individual dynamics of each dataset evolve in isolation.

To demonstrate the capability of $\mathbb{C}$-RSN networks to handle successive inputs, we decompose the activity vector $\vec{x}_t$ into two distinct components: one that corresponds to the elements within the linear segment of the network, and another that pertains to the non-linear segment. This decomposition is represented mathematically as follows:

$$\vec{x}_t = \vec{x}_t^{\,non-linear} + \vec{x}_t^{\,linear}. \tag{16}$$

In this decomposition, the first L components of $\vec{x}_t^{\,non-linear}$ are non-zero, whereas the remaining $N-L$ components are zero. Conversely, for $\vec{x}_t^{\,linear}$, the first L components are zero, and the subsequent $N-L$ components are non-zero. This separation allows to write down the action of the activation function $\tilde{f}$ as $\tilde{f}(\vec{x}_t^{\,non-linear} + \vec{x}_t^{\,linear}) = \vec{x}_t^{\,linear} + \tanh(\vec{x}_t^{\,non-linear})$. This action demonstrates the interplay between the linear and non-linear components of the network: while the linear section remains unaffected, the other segment is transformed by the hyperbolic tangent function, which introduces the non-linearity essential for the network's complex behavior.

Let us now consider a time t much greater than $\bar{t}$, where $\bar{t}$ represents the final time step at which the loss is evaluated. At such a time, the activity vector $\vec{x}_t$, originating from the initial condition $\vec{x}_0$, will have converged to the subspace spanned by the fixed eigenvectors. Mathematically, this is expressed as $\vec{x}_t = \sum_{m=1}^{M} \alpha_m \vec{\psi}^{(m)}$, and it can be understood that $\vec{x}_t$ is equivalent to $\vec{x}_t^{\,linear}$, signifying that it resides solely within the linear sector of the network. Moreover, Eq. (8) tells us that also $\vec{x}_{t+1}$ is confined in the same subspace and in particular:

$$\vec{x}_{t+1} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^{-1}\vec{x}_t = \vec{x}_{t+1}^{\,linear}. \tag{17}$$

Consider now the introduction of an additional activity vector, $\vec{x'}_t$, which is superimposed onto the existing vector $\vec{x}_t$. The resultant neuronal activity within the network is thus captured by the composite vector $\vec{s}_t$, which is the sum of $\vec{x}_t$ and $\vec{x'}_t$, i.e, $\vec{s}_t = \vec{x}_t + \vec{x'}_t$. Performing now a new iteration on $\vec{s}_t$ one obtains:

$$\begin{aligned}\vec{s}_{t+1} &= \tilde{f}(\boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^{-1}(\vec{x}_t + \vec{x'}_t)) \\ &= \tilde{f}(\boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^{-1}\vec{x}_t + \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^{-1}\vec{x'}_t) \\ &= \vec{x}_{t+1}^{\,linear} + \tilde{f}(\boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^{-1}\vec{x'}_t) \\ &= \vec{x}_{t+1} + \vec{x'}_{t+1}.\end{aligned} \tag{18}$$

So, the evolution of the vectors $\vec{x}_t$ and $\vec{x'}_t$ occurs independently. This independence stems from the observation that the processing of

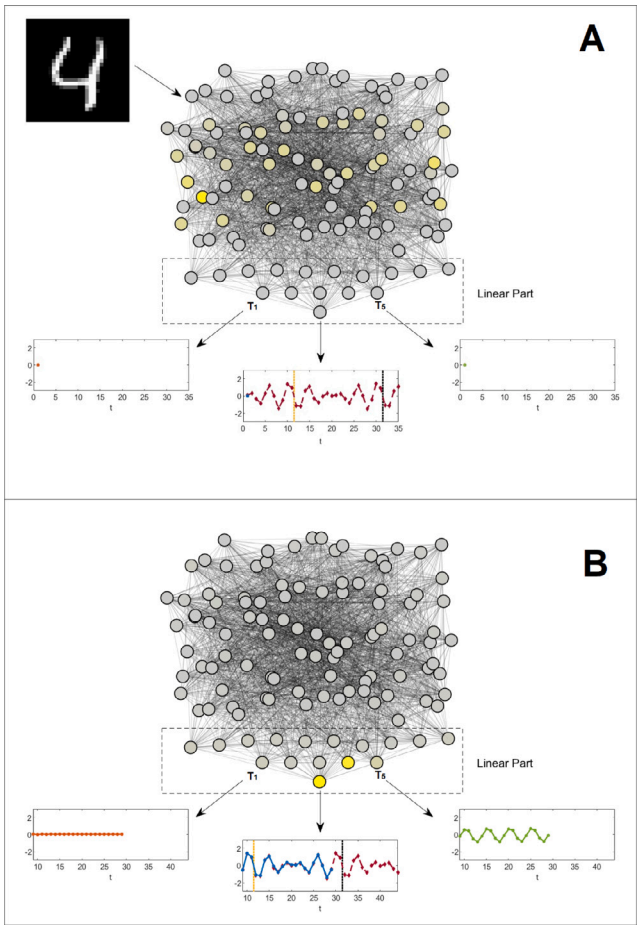

Fig. 3. Visual representation of the $\mathbb{C}$-RSN network at two distinct points in its dynamic evolution. For clarity, this schematic view only illustrates a subset of the network's nodes: individual nodes represent groups, except for the last node and those corresponding to the five fixed eigenvectors. Notice that the linear part is confined in a subportion of the network that is made of neurons that are not associated with the input to be processed (stated differently the nodes associated to pixels are all bearing a non linear filter). The activity of the last neuron is traced by a blue line in the lower sub-panel, and the target temporal function is overlaid in red within the same sub-panel. The non-zero entries of the eigenvectors that constitute the attracting manifold are depicted as a linear horizontal array positioned just above the last neuron's activity display in the illustration. The two smaller sub-panels, to the right and left, highlight the activity in two of these nodes, specifically those characterized by periods $T_1 = \infty$ and $T_5 = 5$. Panel **A** captures the initial condition where the input data stimulates activity in the non-linear portion of the network, leaving the linear segment dormant. Panel **B**, however, presents a snapshot taken during the loss computation window, where activity has transitioned entirely to the linear part of the network. At this juncture, the signal in the last neuron aligns with the target function. The selected neurons display sinusoidal activities, the amplitudes of which correspond to the coefficients of the target function when decomposed into the Fourier basis. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

input data results in a network state which develops within a subspace. Notably, this subspace comprises only those neurons that are part of the network's linear segment.

The above property, therefore, allows us to evaluate and classify different inputs sequentially by working with a trained $\mathbb{C}$-RSN on a specific dataset. Indeed, after the network has completed processing a first input and reached a stable state, it can then receive and classify a second input. The overall final state of the network is captured in the real part of the last neuron's activity over time. The time-dependent activity will be a linear superposition of the functions representing the classes of the two inputs, with a phase shift determined by the time interval between the introduction of these inputs. Mathematically, if t_1 and t_2 represent the times when the inputs are introduced, the real part of the activity in the last neuron can be described as:

$$\mathcal{R}(t) = f_{C_1}(t) + f_{C_2}(t-\gamma) \quad \text{for } t \gg t_1, t_2, \tag{19}$$

where f_{C_1} and f_{C_2} are the target functions corresponding to the classes of the inputs, and $\gamma = t_2 - t_1$ ($t_2 > t_1$) is the time difference between the two inputs, i.e., $\vec{x}_0^{\eta=1}$ and $\vec{x}_0^{\eta=2}$. For the sake of simplicity, we have omitted from $\mathcal{R}(t)$ the dependence on the initial state and the vector $\vec{\beta}$, which represents the training parameters.

An example of sequential evaluation using the MNIST dataset is reported in Fig. 4, which presents four distinct phases of the network's evolution. Panel **A**, at $t = 0$, shows the initial input being introduced into the non-linear part of the network. As per the dynamics outlined in Section 3, the network's activity evolves and, after a few steps (specifically at $t = 39$), it converges within the linear segment. This convergence is evident in Panel **B**, where the last neuron displays a temporal activity pattern resembling the target function.

At $t = 100$, illustrated in Panel **C**, a new input is fed into the network. This triggers the network dynamics to consolidate within the linear part once again, and the activity in the last neuron begins to reflect the linear superposition of the two target functions, in accordance with Eq. (19). Finally, panel **D** captures a later stage where the combined dynamics – influenced by both the residual activity from the first input and the new input – settle within the linear part. Notably, the wave pattern at the last neuron, the entry point of the eigenvectors, is shaped by a linear combination of the two target functions. This pattern intriguingly retains information about the time interval between the successive introductions of the two images.

Employing a $\mathbb{C}$-RSN model trained for individual classification, we can effectively process multiple inputs in sequence. The network's output enables the identification of both input classes and the time gap between their introductions. This ability, demonstrated by the time function in the last neuron, is a natural feature of linear systems and because of the specific design of the $\mathbb{C}$-RSN model such a result applies. As such, it does not require any additional modifications during training.

5. Conclusion

In this manuscript we have introduced an advanced version of the RSN [18], termed the Complex Recurrent Network ($\mathbb{C}$-RSN). The $\mathbb{C}$-RSN model confines non-linearity to specific nodes, rather than allowing it to fade away over time. This design ensures that the model's characteristics remain unchanged, avoiding the temporal convergence towards linearity as seen in the RSN.

Like the RSN, the $\mathbb{C}$-RSN employs a spectral decomposition to define neuron interactions, with a subset of eigenvectors and eigenvalues as trainable parameters. These eigenvalues are constrained to have magnitudes less than one, while the remaining eigenvectors and eigenvalues are fixed and unaltered during training. Notably, the fixed eigenvalues are complex values with a modulus of one, each associated to a specific frequency. The designed architecture guides the asymptotic dynamics towards a subspace defined by the set of eigenvectors associated to the fixed and complex eigenvalues. Moreover, these eigenvectors share the last neuron of the collection, thus allowing for the construction of a signal as a weighted sum of sinusoidal functions corresponding to the frequencies of the fixed eigenvalues.

In this context, we formulated a classification problem where the network learns to reproduce distinct temporal activities for different input classes on the last neuron. Tested with the MNIST dataset, the $\mathbb{C}$-RSN model demonstrated state of the art accuracy. Its classification capability remains consistent over time as the system stabilizes. Once trained, the model can sequentially classify multiple inputs. The final activity observed in the last neuron not only reveals the classes of the various inputs but also the temporal intervals between their introductions, showcasing the model's capacity for handling complex classification tasks.

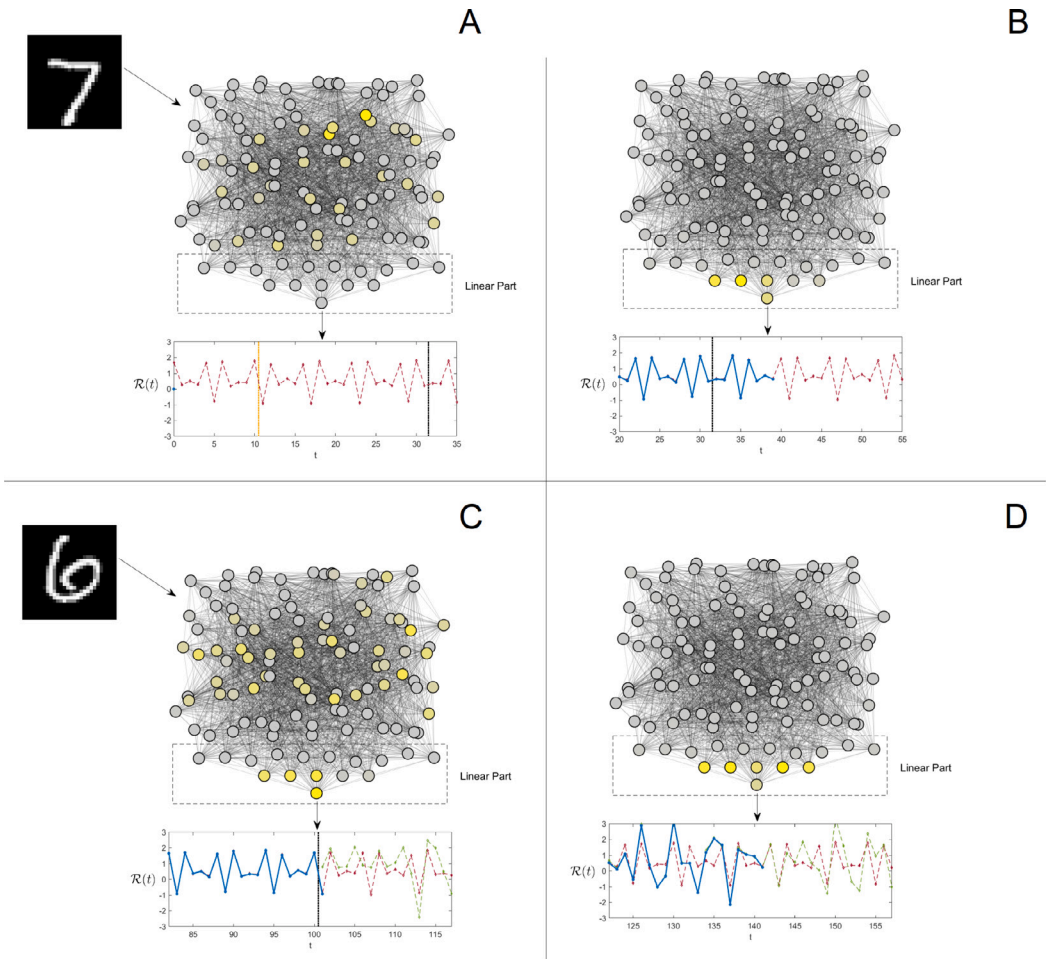

Fig. 4. Four key stages in the evolution of a $\mathbb{C}$-RSN network trained on the MNIST dataset. Panel **A** captures the initial moment of the first data input. Panel **B** depicts a later stage where the network's dynamics have stabilized, with the sub-panel illustrating the alignment of the real part of the last neuron's activity (blue line) with the target function (red line). Panel **C** shows the introduction of a second input into the network, without removing the residual activity from the first input. Finally, Panel **D** demonstrates how the combined dynamics – from both the residual and the new input – converge back towards the linear part of the network. Notably, the wave pattern at the last neuron, corresponding to the eigenvector entries, is a linear combination of the two target functions. Intriguingly, this pattern also preserves information about the time gap between the introductions of the two inputs. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Future studies will aim at exploring the full potential of the $\mathbb{C}$-RSN model. From the fundamental point of view, we will address the possibility of self-consistently shaping the time modulated signal where information is to be eventually store. Furthermore, we plan to systematically challenge the performance of $\mathbb{C}$-RSN across different fields of applications, ranging from time-series analysis to neuroscience, and beyond the basic classification paradigm here addressed as an initial proof of principle.

CRediT authorship contribution statement

Lorenzo Chicchi: Writing – original draft, Investigation, Data curation, Writing – review & editing, Conceptualization. **Lorenzo Giambagli:** Writing – review & editing, Formal analysis, Conceptualization. **Lorenzo Buffoni:** Writing – review & editing, Supervision, Conceptualization. **Raffaele Marino:** Writing – review & editing, Writing – original draft, Formal analysis. **Duccio Fanelli:** Writing – original draft, Supervision, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Duccio Fanelli reports financial support was provided by Ministry of Education and Merit. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by #NEXTGENERATIONEU (NGEU) and funded by the Ministry of University and Research (MUR), National Recovery and Resilience Plan (NRRP), project MNESYS (PE0000006) "A Multiscale integrated approach to the study of the nervous system in health and disease" (DR. 1553 11.10.2022).

References

[1] Esteva Andre, Robicquet Alexandre, Ramsundar Bharath, Kuleshov Volodymyr, DePristo Mark, Chou Katherine, Cui Claire, Corrado Greg, Thrun Sebastian, Dean Jeff. A guide to deep learning in healthcare. Nat Med 2019;25(1):24–9.

[2] Ching Travers, Himmelstein Daniel S, Beaulieu-Jones Brett K, Kalinin Alexandr A, Do Brian T, Way Gregory P, Ferrero Enrico, Agapow Paul-Michael, Zietz Michael, Hoffman Michael M, et al. Opportunities and obstacles for deep learning in biology and medicine. J R Soc Interface 2018;15(141):20170387.

[3] Sezer Omer Berat, Gudelek Mehmet Ugur, Ozbayoglu Ahmet Murat. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. Appl Soft Comput 2020;90:106181.

[4] He Yihui, Lin Ji, Liu Zhijian, Wang Hanrui, Li Li-Jia, Han Song. Amc: Automl for model compression and acceleration on mobile devices. In: Proceedings of the European conference on computer vision. ECCV, 2018, p. 784–800.

[5] Grigorescu Sorin, Trasnea Bogdan, Cocias Tiberiu, Macesanu Gigel. A survey of deep learning techniques for autonomous driving. J Field Robotics 2020;37(3):362–86.

[6] Baldi Pierre, Sadowski Peter, Whiteson Daniel. Searching for exotic particles in high-energy physics with deep learning. Nat Commun 2014;5(1):4308.

[7] Chicchi Lorenzo, Bindi Luca, Fanelli Duccio, Tommasini Simone. Frontiers of thermobarometry: GAIA, a novel deep learning-based tool for volcano plumbing systems. Earth Planet Sci Lett 2023;620:118352.

[8] Giambagli Lorenzo, Fanelli Duccio, Risaliti Guido, Signorini Matilde. Nonparametric analysis of the hubble diagram with neural networks. Astronomy & Astrophysics 2023;678:A13.

[9] Román-González Marcos, Pérez-González Juan-Carlos, Jiménez-Fernández Carmen. Which cognitive abilities underlie computational thinking? Criterion validity of the computational thinking test. Comput Human Behav 2017;72:678–91.

[10] Kriegeskorte Nikolaus, Douglas Pamela K. Cognitive computational neuroscience. Nature Neurosci 2018;21(9):1148–60.

[11] Bishop Christopher M. Pattern recognition and machine learning, In: Information science and statistics, 5th Edition. Springer; 2007.

[12] Shalev-Shwartz Shai, Ben-David Shai. Understanding machine learning: From theory to algorithms. Cambridge university press; 2014.

[13] Silver David, Huang Aja, Maddison Chris J, Guez Arthur, Sifre Laurent, Van Den Driessche George, Schrittwieser Julian, Antonoglou Ioannis, Panneershelvam Veda, Lanctot Marc, et al. Mastering the game of go with deep neural networks and tree search. nature 2016;529(7587):484–9.

[14] Prince Simon JD. Understanding deep learning. MIT press; 2023.

[15] Kriegeskorte Nikolaus. Deep neural networks: a new framework for modeling biological vision and brain information processing. Ann Rev Vis Sci 2015;1:417–46.

[16] LeCun Yann, Bengio Yoshua, Hinton Geoffrey. Deep learning. nature 2015;521(7553):436–44.

[17] Goodfellow Ian, Bengio Yoshua, Courville Aaron. Deep learning. MIT Press; 2016.

[18] Chicchi Lorenzo, Fanelli Duccio, Giambagli Lorenzo, Buffoni Lorenzo, Carletti Timoteo. Recurrent spectral network (RSN): Shaping a discrete map to reach automated classification. Chaos Solitons Fractals 2023;168:113128.

[19] Collobert Ronan, Weston Jason. A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th international conference on machine learning. 2008, p. 160–7.

[20] Selvin Sreelekshmy, Vinayakumar R, Gopalakrishnan EA, Menon Vijay Krishna, Soman KP. Stock price prediction using LSTM, RNN and CNN-sliding window model. In: 2017 international conference on advances in computing, communications and informatics (icacci). IEEE; 2017, p. 1643–7.

[21] Eck Douglas, Schmidhuber Juergen. A first look at music composition using lstm recurrent neural networks. Istituto Dalle Molle Di Studi Sull Intell Artif 2002;103(4):48–56.

[22] Agarwala Nipun, Inoue Yuki, Sly Axel. Music composition using recurrent neural networks. CS 224n: Nat Lang Process Deep Learn, Spring 2017;1:1–10.

[23] Zhang Zizhao, Xie Yuanpu, Xing Fuyong, McGough Mason, Yang Lin. Mdnet: A semantically and visually interpretable medical image diagnosis network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 6428–36.

[24] Eshete Birhanu. Making machine learning trustworthy. Science 2021;373(6556):743–4.

[25] Rudin Cynthia. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat Mach Intell 2019;1(5):206–15.

[26] Linardatos Pantelis, Papastefanopoulos Vasilis, Kotsiantis Sotiris. Explainable ai: A review of machine learning interpretability methods. Entropy 2020;23(1):18.

[27] Conmy Arthur, Mavor-Parker Augustine N, Lynch Aengus, Heimersheim Stefan, Garriga-Alonso Adrià. Towards automated circuit discovery for mechanistic interpretability. 2023, arXiv preprint arXiv:2304.14997.

[28] Bach Sebastian, Binder Alexander, Montavon Grégoire, Klauschen Frederick, Müller Klaus-Robert, Samek Wojciech. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS One 2015;10(7):e0130140.

[29] Baldassi Carlo, Lauditi Clarissa, Malatesta Enrico M, Perugini Gabriele, Zecchina Riccardo. Unveiling the structure of wide flat minima in neural networks. Phys Rev Lett 2021;127(27):278301.

[30] Baldassi Carlo, Lauditi Clarissa, Malatesta Enrico M, Pacelli Rosalba, Perugini Gabriele, Zecchina Riccardo. Learning through atypical phase transitions in overparameterized neural networks. Phys Rev E 2022;106(1):014116.

[31] Lucibello Carlo, Pittorino Fabrizio, Perugini Gabriele, Zecchina Riccardo. Deep learning via message passing algorithms based on belief propagation. Mach Learn: Sci Technol 2022;3(3):035005.

[32] Pacelli R, Ariosto S, Pastore M, Ginelli F, Gherardi M, Rotondo P, et al. A statistical mechanics framework for Bayesian deep neural networks beyond the infinite-width limit. Nat Mach Intell 2023.

[33] Agliari Elena, Alessandrelli Andrea, Barra Adriano, Ricci-Tersenghi Federico. Parallel learning by multitasking neural networks. 2023, arXiv preprint arXiv:2308.04106.

[34] Marino Raffaele, Ricci-Tersenghi Federico. Phase transitions in the mini-batch size for sparse and dense two-layer neural networks. Mach Learn: Sci Technol 2024.

[35] Angelini Maria Chiara, Cavaliere Angelo Giorgio, Marino Raffaele, Ricci-Tersenghi Federico. Stochastic gradient descent-like relaxation is equivalent to glauber dynamics in discrete optimization and inference problems. 2023, arXiv preprint arXiv:2309.05337.

[36] Marino Raffaele, Giambagli Lorenzo, Chicchi Lorenzo, Buffoni Lorenzo, Fanelli Duccio. A bridge between dynamical systems and machine learning: Engineered ordinary differential equations as classification algorithm (EODECA). 2023, arXiv preprint arXiv:2311.10387.

[37] Tsuda Ichiro. Toward an interpretation of dynamic neural activity in terms of chaotic dynamical systems. Behav Brain Sci 2001;24(5):793–810.

[38] Chalk Matthew, Gutkin Boris, Deneve Sophie. Neural oscillations as a signature of efficient coding in the presence of synaptic delays. Elife 2016;5:e13824.

[39] Huebner Bryce, Schulkin Jay. Biological cognition. In: Elements in philosophy of mind, Cambridge University Press; 2022.

[40] Giambagli Lorenzo, Buffoni Lorenzo, Carletti Timoteo, Nocentini Walter, Fanelli Duccio. Machine learning in spectral domain. Nat Commun 2021;12(1):1330.

[41] Buffoni Lorenzo, Civitelli Enrico, Giambagli Lorenzo, Chicchi Lorenzo, Fanelli Duccio. Spectral pruning of fully connected layers. Sci Rep 2022;12(1):11201.

[42] Chicchi Lorenzo, Giambagli Lorenzo, Buffoni Lorenzo, Carletti Timoteo, Ciavarella Marco, Fanelli Duccio. Training of sparse and dense deep neural networks: Fewer parameters, same performance. Phys Rev E 2021;104(5):054312.

[43] Giambagli Lorenzo, Buffoni Lorenzo, Chicchi Lorenzo, Fanelli Duccio. How a student becomes a teacher: learning and forgetting through spectral methods. Adv Neural Inf Process Syst 2024;36.

[44] Aizenberg Igor. Complex-valued neural networks with multi-valued neurons, vol. 353, Springer; 2011.

[45] Kingma Diederik, Ba Jimmy. Adam: A method for stochastic optimization. 2014, arXiv preprint arXiv:1412.6980.

[46] Werbos Paul J. Backpropagation through time: what it does and how to do it. Proc IEEE 1990;78(10):1550–60.

[47] LeCun Yann. The MNIST database of handwritten digits. 1998, http://yann.lecun.com/exdb/mnist/.