University of Sheffield

Efficient Representations over Multiple Timescales

Luca Manneschi

Supervisors: Eleni Vasilaki, Andrew C. Lin

A report submitted in partial fulfilment of the requirements
for the degree of PhD in Computer Science

in the

Department of Computer Science

August 30, 2022

Declaration

All sentences or passages quoted in this document from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure.

Name: LUCA MANNESCHI

Signature: Luca Manneschi

Date: 08/12/2021

Acknowledgements

To all the people that helped and accompanied me during this path.

To the people I miss, to my friends, to Marta, to my family. In particular, to my mother and dad, who supported me all these years and made me grow in an incredibly stimulating environment, from which I am still trying to learn. I will always be thankful.

To Paolo and Guido, for being the firsts to show me the wonders of learning. Paolo, in your loving memory.

To my supervisor, Eleni, who believed in me and helped me to grow as a person and a young researcher. For the freedom and the support she gave me, as a mentor and a friend.

Thank you.

Contents

Chapter 1

Introduction

Despite the recent success of artificial neural networks in solving complex tasks and achieving superhuman performance in a variety of tasks, the ability of biological brains to generalise the learnt knowledge and to quickly adapt to novel situations remains unmatched. This gap between specialised models and behavioural flexibility demonstrated by biological systems encourages researchers to weigh biological plausibility in the process of formulating machine learning models. In this sense, works in the broad field of optimisation and learning lie on a spectrum whose extremes have two diverse research approaches: the first can be described as a virtuous race toward higher performance and more challenging applications, often met at the cost of interpretability of the model and biological plausibility; the second tries to replicate how biological systems operate at a functional scale of modelling detail and gives higher priority to the system understanding rather than performance measures. We believe that the present work lies in between these complementary approaches, also definable as machine learning and computational neuroscience respectively. In this thesis, we will take inspiration from biology to develop machine learning models, but without the certainty that the formulated systems fall within the limits dictated by biological plausibility. Of course, the novel bio-inspired models are required to have practical advantages, measured in terms of performance, interpretability or computational cost, in comparison to pre-existing models. While an example of this line of

thought can be found in the first two papers reported in the thesis, the third publication reported arises from a different, but complementary reasoning process. In the latter, we first formulated theoretically the model from abstract and desirable principles and then demonstrated its ability to explain neuroscientific, experimental findings. In summary, the connection, or flow of information, between the neuroscientific and machine learning fields is bi-directional across the thesis.

A fascinating and inspiring ability of the brain is the capacity to retain and accumulate information over a wide range of timescales. In other words, we can integrate information over different temporal lengths, from the order of seconds, of days, to years, and plan a succession of action to achieve a desirable outcome. A typical example of activity where the concept of multiple timescales plays a fundamental role is the process of reading a book. In this act, the brain can recognise words thanks to the short-term correlations among letters and understand the context of the situation described because of correlations among sentences and chapters. The necessity of neural representations that carry information over different temporal lengths is consequently evident. Nevertheless, the concept of multiple timescales is not often considered or studied in machine learning models that deal with tasks characterised by short and long temporal dependencies. Such models are often, despite their success, computationally expensive, biologically unrealistic and hardly interpretable, in particular from the point of view of the temporal dynamics exhibited by the associated dynamical system. In contrast, the present study will focus on the concept of timescales to gain insights in the process of defining useful representations and developing machine learning algorithms. Thus, multiple timescales constitute an underlying feature of all the research works exposed here, where they are expressed directly by the rate at which information is integrated and/or by the recurrent connectivity of a neural network called echo state network (ESN).

Echo state networks are the theoretical prototype of the reservoir computing paradigm, where the properties of a dynamical system, called the reservoir, are exploited for computation (see Section 1.1). Learning is traditionally characterised by simple optimisation

techniques on the read-out from the representation of the reservoir, which can be viewed as a system with a rich repertoire of dynamical behaviours and intrinsic computation capabilities [1]. The application of simple learning algorithms is possible because the system acts as a non-linear, high dimensional kernel that transforms inputs into representations where data are linearly separable. In this regard, the first publication reported in the thesis aims to formulate a methodology to define efficient reservoirs that exploit a rich repertoire of timescales and that can spontaneously adapt to the temporal features of the task considered (see Section 2.1). The idea behind echo state networks, a multitude of nodes connected through random topological graph structures where learning can locally occur between the reservoir and a set of output neurons, resonates with the current understanding of the working process of various biological networks.

An example of biological circuit that shares similar features to echo state networks is the mushroom body of the Drosophila fruit fly. Dimensionality expansion, non-linearity and synaptic plasticity that seems to exclusively occur on the output of the mushroom body [2], are indeed the distinctive features of such a circuit [3]. In contrast to echo state networks, neurons in the MB respond selectively to different stimuli [4]. Specialised neuronal activities can be achieved through sparsity, which do not only favour savings of neural resources but facilitates learned discrimination between similar stimuli [4]. Thus, taking inspiration from the mushroom body, we developed a normalisation technique that adds learnable thresholds on the read-out from the reservoir representation to introduce sparsity and improve the performance of the system in the context of classification tasks.

The second publication of the following is dedicated to the formulation of such a model and its consequences in terms of learning facilitation (see Section 2.2). A hierarchical model of reservoirs with a rich variety of timescales is adopted in conjunction to learnable thresholds to achieve leading performance on the psMNIST task, a popular benchmark to measure the ability of recurrent neural networks to understand temporal dependencies.

After having focused on echo state networks on supervised learning tasks, we desired to study the concept of multiple timescales in reinforcement learning environments. In

particular, the aim of the third publication (see Section 2.3) reported is to develop a model over multiple timescales for the decision-making process, possibly showing how the adoption of a variety of integration times can lead to robust decision strategies. The simple, yet rich, model developed will demonstrate the fundamental role of multiple timescales to achieve robust performance and reproduce important, experimentally observed, features, to which we will refer with the names of scalar property, signal neutrality and collapsing boundaries. From our knowledge, this is the first model that can spontaneously reproduce such experimental results through learning and reward maximisation, suggesting novel and different interpretations to established neuroscientific observations.

Fig.1.1 highlights the common concepts of the publications through a Venn diagram and aims to facilitate the understanding and reading of the thesis.

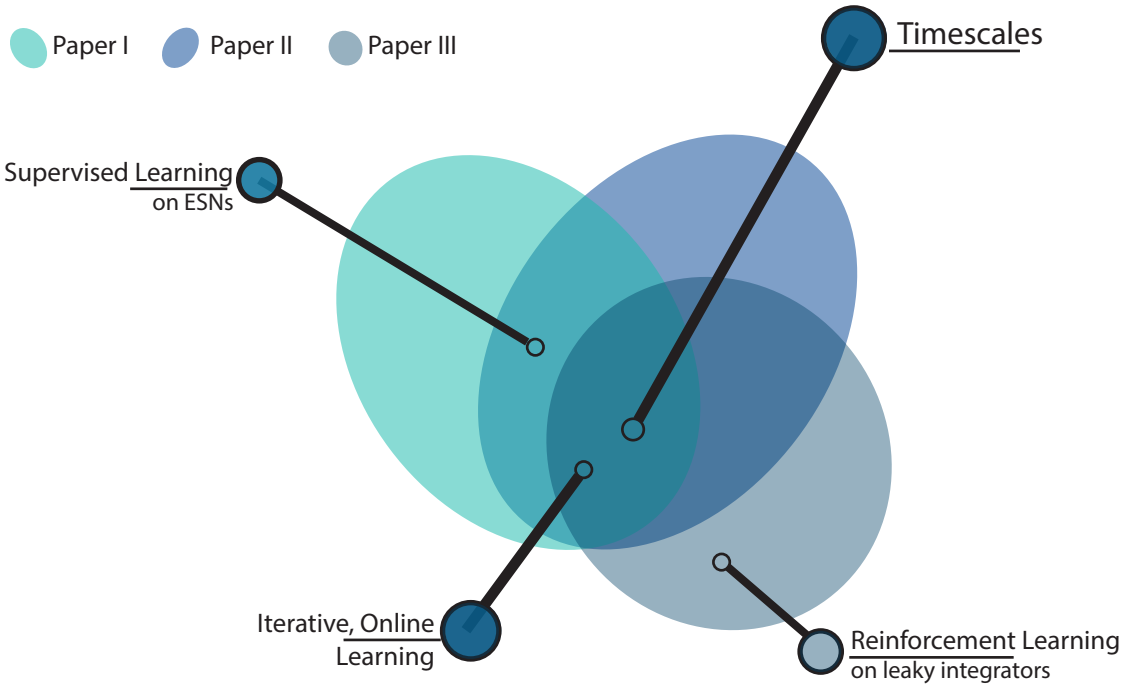

Figure 1.1: Conceptual scheme of the work, highlighting the common ideas of the three publications reported.

1.1 Echo states

Reservoir computing is a computational paradigm where an input signal is processed by a higher dimensional, non-linear dynamical system, called the reservoir. The representation of the reservoir should be dynamically rich in order to capture temporal features of the driving signal and to be exploited for temporally driven machine learning tasks that can not be solved through a linear layer. Traditionally, learning occurs on the read-out of the reservoir exclusively. Now, we introduce reservoir computing by describing its theoretical genesis, which goes by the names of echo state networks (ESN) [5] and liquid state machines [6]. In particular, we will focus on echo state networks in the following. An ESN is an ensemble of recurrently connected nodes, described by the evolution of an N-dimensional variable $\mathbf{x}(t)$ that respond to a M-dimensional external signal $\mathbf{s}(t)$ and that produce a K-dimensional output $\mathbf{y}(t)$. The equations describing this system in continuous time are

$$\tau \frac{d\mathbf{x}(t)}{dt} = -\mathbf{x}(t) + f\Big[\mathbf{W}_{\text{in}}\mathbf{s}(t) + \mathbf{W}\mathbf{x}(t) + \mathbf{W}_{\text{back}}\mathbf{y}(t)\Big] \tag{1.1}$$

$$\tau_y \frac{d\mathbf{y}(t)}{dt} = -\mathbf{y}(t) + f_y\Big[\mathbf{W}_{\text{out,s}}\mathbf{s}(t) + \mathbf{W}_{\text{out,x}}\mathbf{x}(t) + \mathbf{W}_{\text{out,y}}\mathbf{y}(t)\Big] \tag{1.2}$$

where $\mathbf{W}_{\text{in}}$, $\mathbf{W}$, $\mathbf{W}_{\text{back}}$ and $\mathbf{W}_{\text{out,x}}$ ($\mathbf{W}_{\text{out,s}}$, $\mathbf{W}_{\text{out,y}}$) represent the input, recurrent, feedback and output connectivity matrices, whose dimensions are $\text{N}\times\text{M}$, $\text{N}\times\text{N}$, $\text{N}\times\text{K}$ and $\text{K}\times\text{N}$ ($\text{K}\times\text{M}$, $\text{K}\times\text{K}$) respectively. While f and f_y are non-linear activation functions, τ and τ_y define the timescales of the reservoir and the output nodes respectively. Eq. 1.1 and 1.2 are derived considering a general neural architecture (Fig.1.2), which unifies the vast majority of previous research works on echo state networks. An alternative definition of Eq. 1.1 (and similarly of 1.2) is possible by applying the non linear function f exclusively on the term $\mathbf{W}\mathbf{x}(t)$ coming from the recurrency of the system [7]. The

discretised approximate versions of the above equations are

$$\mathbf{x}(t) = (1-\alpha)\mathbf{x}(t-\delta t) + \alpha f\Big[\mathbf{W}_{\text{in}}\mathbf{s}(t) + \mathbf{W}\mathbf{x}(t-\delta t) + \mathbf{W}_{\text{back}}\mathbf{y}(t-\delta t)\Big] \tag{1.3}$$

$$\mathbf{y}(t) = (1-\alpha_y)\mathbf{y}(t-\delta t) + \alpha_y f_y\Big[\mathbf{W}_{\text{out,s}}\mathbf{s}(t) + \mathbf{W}_{\text{out,x}}\mathbf{x}(t) + \mathbf{W}_{\text{out,y}}\mathbf{y}(t-\delta t)\Big] \tag{1.4}$$

where δt is the discretisation step and $\alpha = \delta t/\tau$. Drawing the recurrent connectivity $\mathbf{W}$ of the system from any random distribution would probably lead the system to exhibit a chaotic behaviour, where a small difference in the initial conditions of the network is exponentially amplified across time. In this scenario, the representation of the network would be unreliable and extremely difficult to use for computation purposes. To avoid such chaotic behaviours, we can guarantee the echo state property [5].

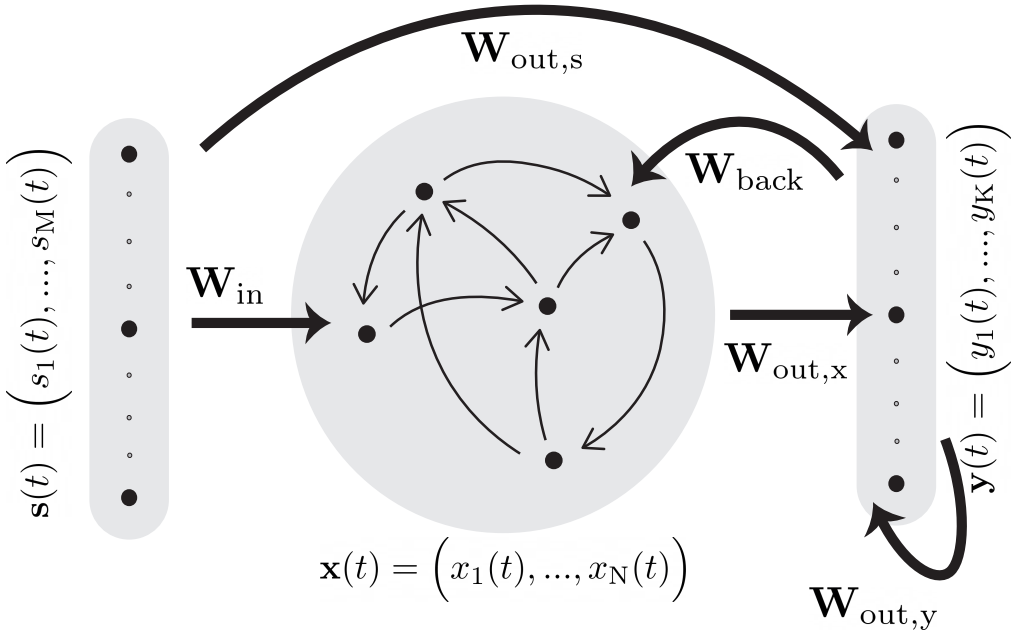

Figure 1.2: General architecture of an Echo State Network.

Definition. *Assume that network activities lie in a compact* [1] *space* $\mathcal{X}$*, that inputs are drawn from a compact space* $\mathcal{S}$*, and that the output feedback is absent (*$\mathbf{W}_{\text{back}} = \mathbf{0}$*). The network has echo states if for every infinite input sequence* $\mathbf{s}(1)$*, ...,* $\mathbf{s}(t-1)$*,* $\mathbf{s}(t)$ *and for*

[1] Explain...

all sequences of activities $\mathbf{x}(1)$, ..., $\mathbf{x}(t-1)$, $\mathbf{x}(t)$ *and* $\tilde{\mathbf{x}}(1)$, ..., $\tilde{\mathbf{x}}(t-1)$, $\tilde{\mathbf{x}}(t)$, *where* $\tilde{\mathbf{x}}(t) = \mathcal{F}\Big(\tilde{\mathbf{x}}(t-1), \mathbf{u}(t)\Big)$ *and* $\mathbf{x}(t) = \mathcal{F}\Big(\mathbf{x}(t-1), \mathbf{u}(t)\Big)$ $\forall t$, *it holds that* $\lim_{\mathrm{t}\to\infty} ||\tilde{\mathbf{x}}(\mathrm{t}) - \mathbf{x}(\mathrm{t})|| = 0$ *regardless to the initial conditions* $\tilde{\mathbf{x}}(0)$, $\mathbf{x}(0)$.

In other words, the network becomes independent of its initial conditions, that is the starting value of $\mathbf{x}$, and is univocally characterised by the considered infinitely long input sequence. In the paper of 2001, Jaeger introduced a practical recipe to achieve the echo state property which is reported below and that will be implicitly used in the rest of the works proposed here

Proposition. *Assume a network with sigmoid-like activation functions, as* $f = \tanh$. *Then:*

i) The echo state property is guaranteed if the maximum singular value Λ *of the connectivity matrix* $\mathbf{W}$ *satisfies* $\Lambda < 1$.

ii) The echo state property is violated for any input containing the zero state $\mathbf{0}$ *if the spectral radius* $\sigma = max(|\operatorname{eig}(W)|) > 1$.

Proof. *Appendix I1 contains a reworked version of the original proof.*

While condition i) guarantees the echo state network property, condition ii) is only necessary (in the case of zero inputs) for the achievement of the latter. From a practical point of view, constraining the spectral radius is often enough to achieve echo states in a large variety of situations. Such a constraint provides a softer bound for the definition of ESNs in comparison to condition i), which can lead to networks with limited dynamical properties. For this reason, the second constraint has been widely adopted by the research community as a practical recipe to rescale the connectivity matrix of the reservoir. We will follow the same approach in the following. In the paper [8], the authors provided a new sufficient condition for the ESN property that is less restrictive than the one considered above, where the maximum singular value of the connectivity matrix needs to be

lower than one. In the new condition, the property is guaranteed if $\mathbf{W}$ is diagonally Schur schedule, that means if a diagonal matrix $\mathbf{D}$ exists such that $\mathbf{W}^{\mathrm{T}}\mathbf{D}\mathbf{W} - \mathbf{D}$ is negative definite.

We note how the above demonstration holds for the case without output feedback, which can lead to delayed effects in the reservoir dynamics that are difficult to control when adaptation occurs on the output weights. In this work, we will focus on echo state networks without output feedback, where the evolution of the network can be described by the following

$$\mathbf{x}(t) = (1-\alpha)\mathbf{x}(t) + \alpha f\Big[\mathbf{W}_{\mathrm{in}}\mathbf{s}(t) + \mathbf{W}\mathbf{x}(t-\delta t)\Big] \tag{1.5}$$

1.2 Achieving meaningful representations

For simplicity and clarity of exposition, we will consider learning in a reservoir described by Eq. 1.5 and refer to this specific case unless stated otherwise. So far, we have not discussed how to set the different terms in Eq.1.5 in order to define a working reservoir and control its dynamical properties. Each parameter of Eq.1.5 impacts differently the behaviour of the network and needs to be tuned to the temporal features of the specific task considered. Thus, we will now focus on the meaning of such parameters and share our experience to guide the application of echo state networks.

- The leakage term α. By definition, α defines the rate of integration of the network and must be set in the interval $[0, 1]$. A value of α close to one corresponds to a quickly reacting reservoir, desirable when the driving signal carries relevant information over high frequencies. However, this implies that the system is also forgetful and that its memory is supported by the timescales introduced only by the recurrency. On the contrary, a value of the leakage term close to zero corresponds to a slowly reacting reservoir, capable of capturing 'smooth', continuous, and long temporal dynamics of the input signal at the cost of losing information that lies on the

fast timescales. Echo state networks were first introduced with $\alpha = 1$ [5] [9], but it is evident that different tasks require the adoption of diverse leakage terms' values [10]. This trade-off among reaction speed and memory that lies in the interval $[0, 1]$ of possible values of α is one of the reasons that led some researchers to introduce systems composed by multiple, and differently tuned echo state networks.

- The connectivity matrix $\mathbf{W}$. The connectivity adopted dictates what is the topological graph structure of the network. We describe here the most common methodology to define the connectivity matrix and we will introduce the diverse alternatives that have been explored by the reservoir community only later. Traditionally, $\mathbf{W}$ [9] has been a sparse matrix reflecting an Erdos-Renyi structure where each node is connected to $\mathrm{N_c}$ other random nodes on average. A pair of nodes is connected, or in other words an element of the matrix is non-zero, with some probability $p_{ER} = \mathrm{N_c}/\mathrm{N}$. Once connections have been defined, each connection strength is drawn from a desirable random distribution, which is usually uniform in the interval $[-1, 1]$. If we call the random matrix defined with the above procedure $\tilde{\mathbf{W}}$, the connectivity $\mathbf{W}$ adopted in Eq.1.5 can be written as

$$\mathbf{W} = \frac{\rho \tilde{\mathbf{W}}}{\max |eig(\tilde{\mathbf{W}})|} \tag{1.6}$$

  where the denominator normalises the matrix (see the previous paragraph for more details regarding conditions for the Echo State property) and the additional parameter $\rho \in [0, 1]$ serves to further control the value of the spectral radius. A higher value of ρ gives more importance to the connectivity matrix and introduces longer timescales in the system. For this reason, values of ρ close to one are optimal when the task considered has long temporal dependencies and requires a larger range of timescales [2]. Regarding the choice of ρ, previous research works have also adopted values greater than one, demonstrating that a spectral radius above unity is optimal

[2] More details on this will be given in Paper I.

for specific tasks. To understand this choice, one needs to consider that the necessary condition introduced in the previous paragraph is derived by considering all possible sequence of inputs, and in particular the null sequence. For specific cases, the echo state network can be well defined even for $\rho > 1$ [8]. We emphasise once more how this procedure described is a practical recipe and does not guarantee the echo state property. An alternative, more restrictive approach, is to apply the sufficient condition in [8]. Following the latter, one should define the random matrix $\mathbf{W}$ with all positive values, rescale the spectral radius inside the unit circle, and finally switch the signs of some desired connections.

We want to notice how the sparsity level of the connectivity p_{ER} can be another important parameter of the reservoir. Practically speaking, it has been noticed that $N_c \approx 10$ [11] is a good starting value for many applications, but often sub-optimal. To understand this, one can think that if $\mathrm{N_c}$ is excessively high, nodes' activities become susceptible to perturbations. A perturbation of the signal or node activity at a given time could indeed propagate rapidly in the network, leading to correlated states and reducing the global amount of information carried by the system. On the contrary, a low intra-connectivity would correspond to a silent network, where perturbations would not propagate and thus remain practically undetected. As often, an effective operational regime lies in the middle. Unfortunately, it is difficult to guess a priori an optimal sparsity level, since the latter leaves unaltered the eigenvalues of the system and consequently the timescales exhibited by the reservoir.

- The input matrix $\mathbf{W}_{\mathrm{in}}$. From the first introduction of echo state networks, the input connectivity has been written as

$$\mathbf{W}_{\mathrm{in}} = \gamma \tilde{\mathbf{W}}_{\mathrm{in}} \tag{1.7}$$

to give emphasis to the rescaling factor γ and to the variance of the random distribution from which the connectivity is defined. In this case, $\tilde{\mathbf{W}}_{\mathrm{in}}$ is typically fully

connected and drawn from a random Gaussian distribution with unitary variance. However, there are many considerations that needs to be done. First, γ should be chosen to confine the network activities well outside the saturation regime of the activation function f considered. This can be achieved by computing the average input signal to a node for a given task or by simple visual inspection of the evolution of the system across time. For the example case of $f = \tanh$, activities should not frequently populate intervals of values close to ± 1.

Second, the optimal level of sparsity of the input connectivity matrix can depend on the input dimensionality. For instance, we have observed that a binary diagonal matrix of randomly placed ± 1 values can lead to better performance than a full Gaussian connectivity matrix when the driving signal is uni-dimensional, since different nodes are not erroneously weighted differently beforehand. However, as the dimensionality of the signal increases, it is important to consider different linear combination of the input features and to reduce the sparsity level of $\mathbf{W}_{\text{in}}$.

Having analysed the different roles of the factors that appear in Eq. 1.5, the following hyperparameters appear to be critical for the definition of an efficient ESN: α, ρ, $\mathrm{N_c}$ and γ. To explore all the dynamic behaviours that the hyperparameters' space has to offer and to find the best operational regime for the considered task is often prohibitive. This expensive tuning procedure and the lack of understanding of the relation between hyperparameters and single task performance constitute an unsolved research challenge in the application of ESNs and reservoirs. A list of possible strategies explored by different research works to mitigate this difficult exploration and tuning problem is reported in appendix I2.

1.3 Learning Algorithms

Finally, we want to focus on the final stage of the reservoir computing paradigm: the learning process. The most standard application of ESN is a regression task, where the

output of the system and the error function E to be minimised are

$$\mathbf{y}(t) = \mathbf{W}_{\text{out}}\mathbf{x}(t)$$
$$\mathrm{E} = \sum_{\mathrm{t}} \left[\tilde{\mathbf{y}}(\mathrm{t}) - \mathbf{y}(\mathrm{t})\right]^2$$

$\tilde{\mathbf{y}}(t)$ is the desired output value and $\mathbf{y}$ is the output of the network. In the case without output feedback of Eq. 1.5 and a linear output $\mathbf{y}$, a closed-form solution to this minimisation problem exists and can be found by setting the gradient of E equal to zero. Following this procedure from a squared error function with an additional L2 penalty term on the output weights leads to ridge regression, which is the most common learning algorithm to train an ESN. Ridge regression, described in more details in Appendix I3, is an offline algorithm that offers a one step solution to the optimisation problem. In the case where the output feedback is present, $\mathbf{W}_{\text{feed}} \neq 0$ in Eq.1.4, a change in the output weights has consequences on the dynamic of the reservoir, and a straightforward application of the closed-form solution is not possible.

Despite the success of ridge regression across the reservoir computing community, it can be challenging to apply Eq.4.3 when the matrix to be inverted is rank deficient. Moreover, it could be necessary to adapt the output weights in an online fashion for some tasks, as for instance in robotics or reinforcement learning applications, and it is extremely difficult to consider the offline ridge regression solution as biologically plausible.
For this reason, previous research have also adapted iterative learning algorithms to minimise a generic error function, that in this case is not constrained to the mean-squared error. In other words, given an arbitrary cost function E, the output weights are optimised iteratively through gradient descent

$$\mathbf{W}_{\text{out}}(n+1) = \mathbf{W}_{\text{out}}(n) - \eta\nabla_{\mathbf{W}_{\text{out}}}\mathrm{E} \tag{1.8}$$

where n refers to the iteration number. Of course, it is possible to use more complex

gradient descent methods as RMSProp or Adam [12] [13], which exploits the first and second order momentum of the derivatives. Finally, learning algorithms that are formulated to deal with the presence of output feedbacks without the utilisation of teacher forcing, and by learning the output weights online as the feedback affects the reservoir dynamics, exist [14]. Important examples of such algorithms are FORCE [15] and BackPropagation-DeCorrelation [14].

We want to notice how ridge regression is the most utilised method because it can be difficult to apply online learning algorithms to ESNs representations and find an optimal solution. In other words, the closed form solution usually outperforms, when applicable, an iterative gradient approach. From our experience, the reason for this lies in the complex reservoir dynamics, which leads gradient descent methods to get trapped in local minima in many tasks.

Chapter 2

Publications

2.1 Paper I

One of the main challenge of reservoir computing and, in particular, of echo state networks is to understand how the hyperparameters of the model are related to a specific task.
The first contribution of the paper is to provide a survey of previous theoretical results regarding dynamical systems and reinterpret the latter in the context of echo state networks. In more details, we quantify pseudo-analytically the repertoire of timescales of the network and their link to the hyperparameters' values in order to guide the fine-tuning procedure of the network dynamical properties to a given task.
The second contribution of the paper is to interpret the performance of models composed by multiple echo state networks. Interestingly, the most efficient structure is composed by multiple networks connected hierarchically, where deeper reservoirs have lower timescales than the ones closer to the external signal. This hierarchy of timescales is also reflected in the modularity of various biological networks, where sensory information needs to be rapidly represented and then fed to circuits where integration occurs over a variety, but more importantly slow, timescales.
Finally, we developed a learning rule to adapt the hyperparameters of the system and change the timescales of the model in a data-driven fashion. The learning rule is derived through gradient descent and from recently proposed approximations of backpropagation through time. The derived optimisation method can be computed online, but the intrinsic

nature of the hyperparameters of the system leads inevitably to a non-local learning rule. Simulations are performed in standard reservoir computing benchmarks, but also in more complex tasks that were not previously faced with echo state networks.

The paper has been published under Frontiers in Applied Mathematics and Statistics (Front. Appl. Math. Stat., 17 February 2021 Sec.Dynamical Systems).

My contributions to the work were: model development, coding, testing, writing of the paper.

EXPLOITING MULTIPLE TIMESCALES IN HIERARCHICAL ECHO STATE NETWORKS

Luca Manneschi [1], Matthew O. A. Ellis [1], Guido Gigante [2], Andrew C. Lin [3,4], Paolo Del Giudice [2,+], Eleni Vasilaki [1,+]

[1] Department of Computer Science, The University of Sheffield, Sheffield, United Kingdom,
[2] National Center for Radiation Protection and Computational Physics, Italian Institute of Health, Rome, Italy,
[3] Department of Biomedical Science, The University of Sheffield, Sheffield, United Kingdom,
[4] Neuroscience Institute, The University of Sheffield, Sheffield, United Kingdom
[+] Joint senior authorship

Abstract

Echo state networks (ESNs) are a powerful form of reservoir computing that only require training of linear output weights whilst the internal reservoir is formed of fixed randomly connected neurons. With a correctly scaled connectivity matrix, the neurons' activity exhibits the echo-state property and responds to the input dynamics with certain timescales. Tuning the timescales of the network can be necessary for treating certain tasks, and some environments require multiple timescales for an efficient representation. Here we explore the timescales in hierarchical ESNs, where the reservoir is partitioned into two smaller linked reservoirs with distinct properties. Over three different tasks (NARMA10, a reconstruction task in a volatile environment, and psMNIST), we show that by selecting the hyper-parameters of each partition such that they focus on different timescales, we achieve a significant performance improvement over a single ESN. Through a linear analysis, and under the assumption that the timescales of the first partition are much shorter than the second's (typically corresponding to optimal operating conditions), we interpret the feedforward coupling of the partitions in terms of an effective representation of the input signal, provided by the first partition to the second, whereby the instantaneous input signal is expanded into a weighted combination of its time derivatives. Furthermore, we propose a data-driven approach to optimise the hyper-parameters through a gradient descent optimisation method that is an online approximation of backpropagation through time. We demonstrate the application of the online learning rule across all the tasks considered.

1 Introduction

The high inter-connectivity and asynchronous loop structure of Recurrent Neural Networks (RNNs) make them powerful techniques for processing temporal signals[1]. However, the complex inter-connectivity of RNNs means that they cannot be trained using the conventional back-propagation (BP) algorithm[2] used in feed-forward networks, since

each neuron's state depends on other neuronal activities at previous times. A method known as Back-Propagation-Through-Time (BPTT) [3], which relies on an unrolling of neurons' connectivity through time to propagate the error signal to earlier time states, can be prohibitively complex for large networks or time series. Moreover, BPTT is not considered biologically plausible as neurons must retain memory of their activation over the length of the input and the error signal must be propagated backwards with symmetric synaptic weights [4].

Many of these problems can be avoided using an alternative approach: reservoir computing (RC). In the subset of RC networks known as Echo State networks, a fixed 'reservoir' transforms a temporal input signal in such a way that only a single layer output perceptron needs to be trained to solve a learning task. The advantage of RC is that the reservoir is a fixed system that can be either computationally or physically defined. Since it is fixed it is not necessary to train the reservoir parameters through BPTT, making RC networks much simpler to train than RNNs. Furthermore, the random structure of a RC network renders the input history over widely different time-scales, offering a representation that can be used for a wide variety of tasks without optimising the recurrent connectivity between nodes.

Reservoirs have biological analogues in cerebellum-like networks (such as the cerebellum, the insect mushroom body and the electrosensory lobe of electric fish), in which input signals encoded by relatively few neurons are transformed via 'expansion re-coding' into a higher-dimensional space in the next layer of the network, which has many more neurons than the input layer [5, 6, 7, 8]. This large population of neurons (granule cells in the cerebellum; Kenyon cells in the mushroom body) acts as a reservoir because their input connectivity is fixed and learning occurs only at their output synapses. The principal neurons of the 'reservoir' can form chemical and electrical synapses on each other (e.g. Kenyon cells: [9, 10, 11]), analogous to the recurrent connectivity in reservoir computing that allows the network to track and transform temporal sequences of input signals. In some cases, one neuronal layer with recurrent connectivity might in turn connect to

another neuronal layer with recurrent connectivity; for example, Kenyon cells of the mushroom body receive input from olfactory projection neurons of the antennal lobe, which are connected to each other by inhibitory and excitatory interneurons [12, 13]. Such cases can be analogised to hierarchically connected reservoirs. In biological systems, it is thought that transforming inputs into a higher-dimensional neural code in the 'reservoir' increases the associative memory capacity of the network [5]. Moreover, it is known that for the efficient processing of information unfolding in time, which requires networks to dynamically keep track of past stimuli, the brain can implement ladders of neural populations with hierarchically organised 'temporal receptive fields' [14].

The same principles of dimensional expansion in space and/or time apply to artificial RC networks, depending on the non-linear transformation of the inputs into a representation useful for learning the task at the single linear output layer. We focus here on a popular form of RC called Echo State Networks [15], where the reservoir is implemented as a RNN with a fixed, random synaptic connection matrix. This connection matrix is set so the input 'echoes' within the network with decaying amplitude. The performance of an Echo State Network depends on certain network hyper-parameters that need to be optimised through grid search or explicit gradient descent. Given that the dependence of the network's performance on such hyper-parameters is both non-linear and task-dependent, such optimisation can be tedious.

Previous works have studied the dependence of the reservoir properties on the structure of the random connectivity adopted, studying the dependence of the reservoir performance on the parameters defining the random connectivity distribution, and formulating alternatives to the typical Erdos-Renyi graph structure of the network [16, 17, 18]. In this sense, in [17] a model with a regular graph structure has been proposed, where the nodes are connected forming a circular path with constant shortest path lengths equal to the size of the network, introducing long temporal memory capacity by construction. The memory capacity has been studied previously for network parameters such as the spectral radius (ρ) and sparsity; in general memory capacity is higher for ρ close to 1 and low sparsity,

but high memory capacity does not guarantee high prediction [19, 20]. ESNs are known to perform optimally when at the "edge of criticality"[21], where low prediction error and high memory can be achieved through network tuning.

More recently, models composed of multiple reservoirs have gathered the attention of the community. From the two ESNs with lateral inhibition proposed in [22], to the hierarchical structure of reservoirs first analysed by Jaeger in [23], these complex architectures of multiple, multilayered reservoirs have shown improved generalisation abilities over a variety of tasks [24, 23, 25]. In particular, the works [26] [27] have studied different dynamical properties of such hierarchical structures of ESNs, while [28] have proposed hierarchical (or deep) ESNs with projection encoders between layers to enhance the connectivity of the ESN layers. The partitioning (or modularity) of ESNs was studied by [29], where the ratio of external to internal connections was varied. By tuning this partitioning performance can be increased on memory or recall tasks. Here we demonstrate that one of the main reasons to adopt a network composed by multiple, pipelined sub-networks, is the ability to introduce multiple timescales in the network's dynamics, which can be important in finding optimal solutions for complex tasks. Examples of tasks that require such properties are in the fields of speech, natural language processing, and reward driven learning in partially observable Markov decision processes [30]. A hierarchical structure of temporal kernels [31], as multiple connected ESNs, can discover higher level features of the input temporal dynamics. Furthermore, while a single ESN can be tuned to incorporate a distribution of timescales with a prefixed mode, optimising the system hyper-parameters to cover a wide range of timescales can be problematic.

Here, we show that optimisation of hyper-parameters can be guided by analysing how these hyper-parameters are related to the timescales of the network, and by optimising them according to the temporal dynamics of the input signal and the memory required to solve the considered task. This analysis improves performance and reduces the search space required in hyper-parameter optimisation. In particular, we consider the case where an ESN is split into two sections with different hyper-parameters resulting in separate

temporal properties. In the following, we will first provide a survey of timescales in ESNs before presenting the comparative success of these hierarchical ESNs on three different tasks. The first is the non-linear auto-regressive moving average 10 (NARMA10) task which requires both memory and fast non-linear transformation of the input. Second, we explore the performance of the network in a reconstruction and state "perception" task with different levels of external white noise applied on the input signal. Finally, we apply the hierarchical ESN to a permuted sequential MNIST classification task, where the usual MNIST hand written digit database is serialised and permuted as a 1d time-series.

2 Survey of timescales in Echo State networks

We begin by describing the operations of an ESN and present a didactic survey of the inherent timescales in ESNs, which will be drawn upon in later sections to analyse the results.

As introduced in the previous section, an ESN is a recurrent neural network and the activity, $\mathbf{x}(t)$, of the neurons due to a temporal input signal $\mathbf{s}(t)$ is given by

$$\mathbf{x}(t+\delta t) = (1-\alpha)\mathbf{x}(t) + \alpha f\left(\mathbf{h}(t)\right), \tag{1}$$

$$\mathbf{h}(t) = \gamma \mathbf{W}_{\text{in}}\mathbf{s}(t) + \rho \mathbf{W}\mathbf{x}(t), \tag{2}$$

where $\mathbf{W}$ is a possibly sparse random matrix defining the connectivity of the network, $\mathbf{W}_{\text{in}}$ defines the input adjacency matrix, and γ is a rescaling factor of the input weights. $\alpha = \delta t/\tau$ is the leakage term of the node, and ρ is a scaling factor for the spectral radius of the connectivity matrix and will be discussed in more detail in the following. $f()$ is a non-linear function, which in this work we define as the hyperbolic tangent. A good practical condition for the Echo-State property [15] is to rescale the initial random connectivity matrix $\mathbf{W}$ by its maximum eigenvalue magnitude (spectral radius), $|\lambda_{\mathbf{W}}^{\max}| = \max|\text{eig}(\mathbf{W})|$,

thus ensuring a unitary spectral radius which can be tuned using ρ as a hyper-parameter. In practice, $\mathbf{W}$ is constructed from a matrix of Normally distributed random numbers and the sparseness is enforced by randomly setting to zero a fixed proportion of these elements. Typically 10 non-zero connections per node are retained in $\mathbf{W}$.

The timescales of this dynamical system are closely linked to the specific structure of $\mathbf{W}$ and to the two hyper-parameters; α and ρ. Since α is the leakage rate, it directly controls the retention of information from previous time steps, while ρ specifies the maximum absolute magnitude of the eigenvalues and as such tunes the decay time of internal activity of the network. Thus, the basic hyper-parameters that need to be set are γ, α and ρ. Considering the nonlinear dependence of the network performance on these values and the task-dependent nature of an efficient parameterisation, this process can be challenging. Such hyper-parameters are commonly optimised through a grid search or through explicit gradient descent methods in online learning paradigms [32]. However, the fine tuning procedure can be guided, and the searchable space reduced, using a simple analysis of the hyper-parameters' relation to the timescales of the network, the external signal's temporal dynamics, and the memory required to solve the considered task.

Considering that the eigenvalues $\lambda_{\mathbf{W}}$ of the connectivity matrix are inside the imaginary unit circle due to the normalisation procedure described previously, and that α is a constant common to all neurons, the eigenvalues of the linearised system given by Eq. 1 are

$$\lambda = 1 - \alpha(1 - \rho\lambda_{\mathbf{W}}). \tag{3}$$

This corresponds to a rescaling of value $\alpha\rho$ and to a translation of value $1 - \alpha$ across the real axis of the original $\lambda_{\mathbf{W}}$. This operation on the eigenvalues of $\mathbf{W}$ is depicted in Fig. 1A. Thus, considering that each eigenvalue λ_i can be decomposed in its corresponding exponential decaying part $\exp(-\delta t/\tau_i)$ and its oscillatory imaginary component, the timescales of the linearised system are

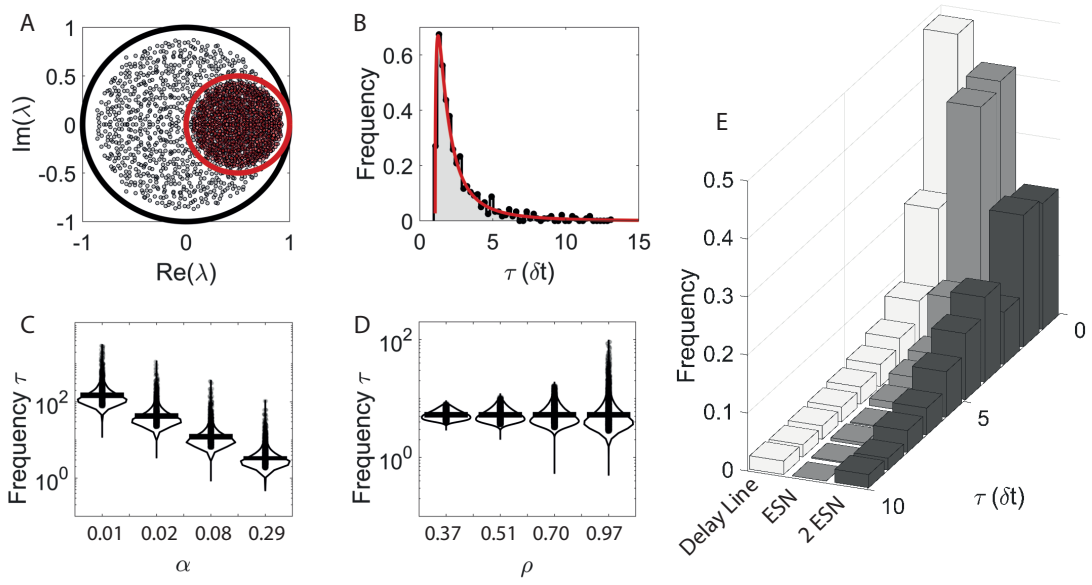

Figure 1: The analysis of the timescales of the system in the linear regime can guide the search for the optimal values of the hyper-parameters α and ρ. **A**: Translation and scaling of the eigenvalues of the system due to the presence of the leakage factor. **B**: Example of distribution of timescales, computed analytically (red line) and computationally (black points) estimated from the eigenvalues of $\mathbf{W}$. **C**: Pirate plot of the distributions of timescales as α increases. Both axes are logarithmic. Higher α values correspond to longer timescales and to a more compressed range of timescales (logarithmic y-axis). **D**: Pirate plot of the distributions of timescales: as ρ increases, the range of timescales expands. Again, both axes are logarithmic. **E**: Example distributions of timescales for reservoirs with different connectivity structure. From left to right, a delay line, single ESN, 2 ESNs (connected and unconnected, see text for the reason why the timescales for these two structures are the same in the linear regime). The higher complexity of the models reported is reflected in a richer distribution of timescales.

$$\tau = \frac{\delta t}{1 - \mathrm{Re}(\lambda)} \tag{4}$$

$$= \frac{\delta t}{\alpha(1 - \rho \mathrm{Re}(\lambda_{\mathbf{W}}))} \tag{5}$$

When the connectivity matrix, $\mathbf{W}$, is given by a sparse matrix with non-zero elements drawn randomly from a uniform distribution with the range $[-1, 1]$, then the corresponding eigenvalues will be uniformly distributed within a circle with a radius of $\max(|\lambda_{\mathbf{W}}|)$ in the complex plane [33]. These eigenvalues are then re-scaled by $\max(|\lambda_{\mathbf{W}}|)$ to en-

sure they are within the unit circle. The distribution of the eigenvalues then reveals the distribution of timescales of the linearised system. Indeed, given $p\left(\mathrm{Re}(\lambda), \mathrm{Im}(\lambda)\right)$, the distribution of timescales can be found through computation of the marginal $p\big(\mathrm{Re}(\lambda)\big) = \int p\big(\mathrm{Re}(\lambda), \mathrm{Im}(\lambda)\big) d\mathrm{Im}(\lambda)$ and the change of variable defined in equation 5, giving

$$p(\tau) = \frac{2\delta t^2}{\pi \alpha^2 \rho^2 \tau^2} \sqrt{\alpha^2 \rho^2 - \left(\alpha - \delta t/\tau\right)^2} \tag{6}$$

Importantly we note that whilst the eigenvalues are uniformly distributed over the unit circle, the timescales are not due to the inverse relationship between them. The resulting distribution of the linearised system, shown in Fig. 1B (red line), is in excellent agreement with the numerically computed distribution for a single ESN (black points + shaded area).

The analytical form of the distribution, together with Eq. 5, allows us to explicitly derive how changes in α and ρ affect the network timescales. Notably we can obtain analytical expression for the minimum, maximum and most probable (peak of the distribution) timescale:

$$\tau_{\min} = \frac{\delta t}{\alpha(1+\rho)}, \tag{7}$$

$$\tau_{\max} = \frac{\delta t}{\alpha(1-\rho)}, \tag{8}$$

$$\tau_{\mathrm{peak}} = \frac{5\delta t}{4\alpha(1-\rho^2)} \left(1 - \sqrt{1 - \frac{24}{25}(1-\rho^2)}\right) \tag{9}$$

where Eq. 8 and 7 can be derived directly from Eq. 5, while Eq. 9 follows from maximisation of Eq. 6. As expected, α strongly affects all these three quantities; interestingly, though, α does not influence the relative range of the distribution, $\tau_{\max}/\tau_{\min} = (1+\rho)/(1-\rho)$. Indeed α plays the role of a unit of measure for the τs, and can then be used to scale the distribution in order to match the relevant timescales for the specific task. On the other hand, ρ does not strongly affect the shape of the distribution, but determines how dispersed the τs are. Given the finite number of τs expressed by a finite ESN, the hyper-

parameter ρ can be used to balance the raw representation power of the network (how wide the range of timescales is) with the capacity to approximate any given timescale in that range. Fig. 1C and D give a more detailed view of how the distribution of timescales changes as α and ρ, respectively, vary; note the logarithmic scale on the y-axis, that makes the dependence on α linear. The link between the eigenvalues and the reservoir dynamics can be shown through the analysis of the network response to an impulsive signal, shown in Section 5.2, where the experimental activities are compared with the theoretical ones expected from the linearised system.

2.1 Hierarchical Echo-State Networks

Different studies have proposed alternatives to the random structure of the connectivity matrix of ESNs, formulating models of reservoirs with regular graph structures. Examples include a delay line [17], where each node receives and provides information only from the previous node and the following one respectively, and the concentric reservoir proposed in [18], where multiple delay lines are connected to form a concentric structure. Furthermore, the idea of a hierarchical architecture of ESNs, where each ESN is connected to the preceding and following one, has attracted the reservoir computing community for its capability of discovering higher level features of the external signal [34]. Fig. 2 schematically shows the architecture for (**A**) a single ESN, (**B**) 2 sub-reservoir hierarchical ESN for which the input is fed into only the first sub-reservoir which in turn feeds into the second and (**C**) a parallel ESN, where two unconnected sub-reservoirs receive the same input. These heirarchical ESNs are identical to the 2 layer DeepESN given by [27]. A general ensemble of interacting ESNs can be described by

$$\mathbf{x}^{(k)}(t+\delta t) = (1-\alpha^{(k)})\mathbf{x}^{(k)} + \alpha^{(k)} f\left(\mathbf{h}^{(k)}(t)\right), \tag{10}$$

$$\mathbf{h}^{(k)}(t) = \gamma^{(k)}\mathbf{W}_{\text{in}}^{(k)}\mathbf{s}^{(k)}(t) + \sum_{l}^{N_{\text{ESN}}} \rho^{(kl)}\mathbf{W}^{(kl)}\mathbf{x}^{(l)}(t), \tag{11}$$

where the parameters have the similar definitions as in the case of a single ESN in Eq. 1. The index k indicates the network number and N_{ESN} is the total number of networks under consideration. In a hierarchical structure of ESNs $\mathbf{W}^{(kl)} \neq 0$ for $k = l$ or $k = l + 1$ only, and $\mathbf{W}^{(kl)}$ can be drawn from any desirable distribution thanks to the absence of feedback connections to higher-order reservoirs. Indeed, in this case, a good practical condition for the Echo-State network property is that all the inner connectivity matrices $\mathbf{W}^{(kk)}$ have eigenvalues with an absolute value less or equal than one. Furthermore, in the typical hierarchical structure proposed in previous works [23, 24, 27, 25, 35], the input is fed to the first network only, and $\mathbf{W}_{\mathrm{in}}^{(k)} \neq 0$ if $k = 1$ only. We emphasise that the values of $\alpha^{(k)}$ and $\rho^{(k)}$, which are closely related to the timescales and repertoire of dynamics of network number k (and, in the case of hierarchical reservoirs, also to all subsequent networks), do not have to be equal for each ESN, but can be chosen differently to fit the necessity of the task. In particular, some tasks could require memory over a wide range of timescales that could not effectively be covered by a single ESN.

In Fig. 1E we show examples of the timescale distributions of the corresponding linearised dynamical systems for different ESN structures, from the simple delay line model to the higher complexity exhibited from two hierarchical ESNs. In order from left to right, the histograms of timescales are for a delay line, a single ESN, and two ESNs (whether hierarchically connected or unconnected; see below for clarification). All the models share an ESN with $\rho = 0.9$ and $\alpha = 0.9$; where present, the second reservoir has $\alpha = 0.2$. By construction, the richness and range of timescales distributions reported increases with the complexity of the models. However, we note how a simple delay line could exhibit longer temporal scales than the other structures analysed thanks to its constant and high value of minimum path length between any pairs of nodes. Nevertheless, its limited dynamics restricts its application to simple tasks. The cases with two ESNs show a bimodal distribution corresponding to the two values of α.

Yet, the spectrum of the eigenvalues of the linearised system is only partially informative of the functioning and capabilities of an ESN. This is clearly demonstrated by the

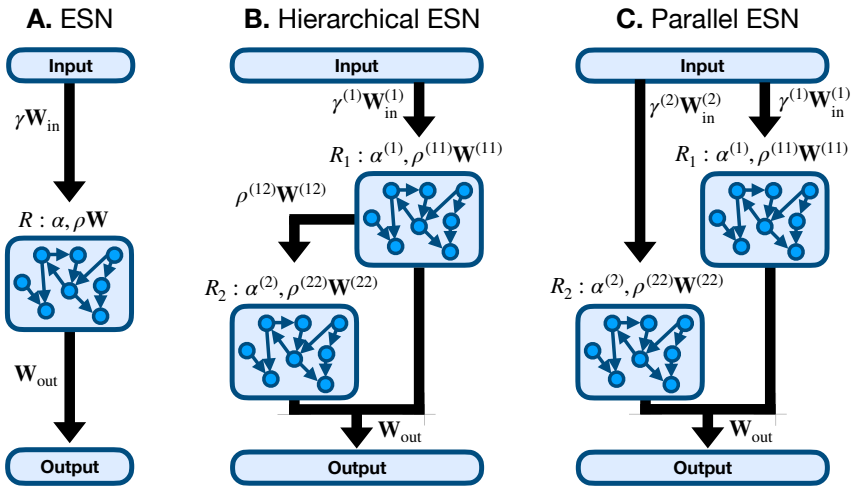

Figure 2: Single and hierarchical echo-state network (ESN) architectures. **A**: A single ESN with internally connected nodes with a single set of hyper-parameters α and ρ. **B**: A hierarchical ESN composed of 2 connected reservoirs where the input is fed into reservoir 1 only and the connection is unidirectional from R1 to R2, which is identical to the 2 layer DeepESN of [27]. **C**: A parallel (or unconnected hierarchical) ESN where the network is partitioned into 2 reservoirs, R1 and R2, which each receive the input and provide output but have distinct hyper-parameters.

fact that a hierarchical and a parallel ESN share the same spectrum in the linear regime. Indeed, for a hierarchical ESN, whose connectivity matrix of the linearised dynamics is given by:

$$\tilde{\mathbf{W}} = \begin{bmatrix} \mathbf{W}^{(11)} & \mathbf{0} \\ \mathbf{W}^{(12)} & \mathbf{W}^{(22)} \end{bmatrix}, \tag{12}$$

it is easy to demonstrate that every eigenvalue of $\mathbf{W}^{(11)}$ and $\mathbf{W}^{(22)}$ is also an eigenvalue of $\tilde{\mathbf{W}}$, irrespective of $\mathbf{W}^{(12)}$, not unlike what happens for a parallel ESN (where $\mathbf{W}^{(12)} = \mathbf{0}$, and hence the demonstration follows immediately). Nonetheless, as we will see in the next sections, the hierarchical ESN has better performance on different tasks compared to the other structures considered, including the parallel ESN.

It is interesting to note, in this respect, that the success of the hierarchical ESN is generally achieved when the leakage term of the first reservoir is higher than the leakage

term of the second (or, in other words, when the first network has much shorter timescales). Such observation opens the way to an alternative route to understand the functioning of the hierarchical structure, as the first reservoir expanding the dimensionality of the input and then feeding the enriched signal into the second network. Indeed, in the following, we will show how, in a crude approximation and under the above condition of a wide separation of timescales, the first ESN extracts information on the short term behaviour of the input signal, notably its derivatives, and the second ESN integrates such information over longer times.

We begin with the (continuous time) linearized dynamics of a Hierarchical ESN is given by

$$\dot{\mathbf{x}}^{(1)}(t) = -\mathbf{M}^{(1)}\mathbf{x}^{(1)}(t) + \mathbf{W}_{\text{in}}^{(1)}\mathbf{s}(t), \tag{13}$$

$$\dot{\mathbf{x}}^{(2)}(t) = -\mathbf{M}^{(2)}\mathbf{x}^{(2)}(t) + \mathbf{W}^{(12)}\mathbf{x}^{(1)}(t), \tag{14}$$

where, for simplicity, we have reabsorbed the $\rho^{(kl)}$ and $\gamma^{(k)}$ factors into the definitions of $\mathbf{W}^{(kl)}$ and $\mathbf{W}_{\text{in}}^{(k)}$ respectively, and the new constants can be derived with reference to Eq. 1 and 2; for example:

$$\mathbf{M}^{(k)} = \frac{\alpha^{(k)}}{\delta t}\left[1 - f'(0)\,\rho^{(k)}\,\mathbf{W}^{(kk)}\right]. \tag{15}$$

The neuron activity can be projected on to the left eigenvector of each of the $\mathbf{M}^{(i)}$ matrices. As such we define the eigenvector matrices, $\mathbf{V}^{(i)}$, where each row is a left eigenvector and so satisfies the equation $\mathbf{V}^{(i)}\mathbf{M}^{(i)} = \mathbf{\Lambda}^{(i)}\mathbf{V}^{(i)}$. $\mathbf{\Lambda}^{(1)}$ and $\mathbf{\Lambda}^{(22)}$ are the diagonal matrices of the eigenvalues of the two $\mathbf{M}$ matrices. Using these we can define $\mathbf{y}^{(k)} \equiv \mathbf{V}^{(k)}\,\mathbf{x}^{(\mathbf{k})}$, and so the dynamical equations can be expressed as

$$\dot{\mathbf{y}}^{(1)}(t) = -\mathbf{\Lambda}^{(1)}\,\mathbf{y}^{(1)}(t) + \tilde{\mathbf{W}}_{\text{in}}^{(1)}\mathbf{s}(t), \tag{16}$$

$$\dot{\mathbf{y}}^{(2)}(t) = -\mathbf{\Lambda}(2)\,\mathbf{y}^{(2)}(t) + \tilde{\mathbf{W}}^{(12)}\,\mathbf{y}^{(1)}(t), \tag{17}$$

where $\tilde{\mathbf{W}}_{\text{in}}^{(1)} = \mathbf{V}^{(1)}\mathbf{W}_{\text{in}}^{(1)}$ and $\tilde{\mathbf{W}}^{(12)} = \mathbf{V}^{(2)}\,\mathbf{W}^{(12)}\,\left(\mathbf{V}^{(1)}\right)^{-1}$ are the input and connection

matrices expanded in this basis. Taking the Fourier transform on both sides of Eq. 16, such that $FT\left[\mathbf{y}^{(1)}(t)\right] = \tilde{\mathbf{y}}^{(1)}(\omega)$ and $FT\left[\dot{\mathbf{y}}^{(1)}(t)\right] = -i\omega\tilde{\mathbf{y}}^{(1)}(\omega)$, where i is the imaginary unit. The transform $\tilde{\mathbf{y}}^{(2)}(\omega)$ of $\mathbf{y}^{(2)}(t)$ can now be expressed as a function of the transform of the signal $\tilde{\mathbf{s}}(\omega)$ giving

$$(\mathbf{\Lambda}^{(1)} - i\omega\mathbf{I})\tilde{\mathbf{y}}^{(1)}(\omega) = \tilde{\mathbf{W}}_{\text{in}}^{(1)}\tilde{\mathbf{s}}(\omega) \tag{18}$$

where $\mathbf{I}$ is the identity matrix of the same size as $\mathbf{\Lambda}^{(1)}$. If the second ESN's timescale are much longer than that of the first one (*i.e.*, $\mathbf{\Lambda}^{(1)} \gg \mathbf{\Lambda}^{(2)}$), then we can expand the inverse of the $\tilde{\mathbf{y}}^{(1)}$ coefficient on the LHS of Eq. 18 when $\mathbf{\Lambda}^{(1)} \to \infty$ as

$$(\mathbf{\Lambda}^{(1)} - i\omega\mathbf{I})^{-1} = (\mathbf{\Lambda}^{(1)})^{-1}\left(1 - i\omega(\mathbf{\Lambda}^{(1)})^{-1}\right)^{-1} \tag{19}$$

$$\approx (\mathbf{\Lambda}^{(1)})^{-1}\sum_{n=0}^{\infty}(i\omega(\mathbf{\Lambda}^{(1)})^{-1})^n \tag{20}$$

By applying this approximation to Eq. 18, and by defining the diagonal matrix of characteristic times $\mathbf{T}^{(1)} \equiv -(\mathbf{\Lambda}^{(1)})^{-1}$, the relation between the activity of reservoir 1 and the input in Fourier space is given by

$$\tilde{\mathbf{y}}^{(1)}(\omega) = -\mathbf{T}^{(1)}\sum_{n=0}^{\infty}(-i\omega\mathbf{T}^{(1)})^n\tilde{\mathbf{W}}_{\text{in}}^{(1)}\tilde{\mathbf{s}}(\omega). \tag{21}$$

The coefficients of this series are equivalent to taking successive time derivatives in Fourier space, such that $(-i\omega)^n\tilde{s} = d^{(n)}\tilde{s}/dt^{(n)}$. So by taking the inverse Fourier transform we find the following differential equation for $y^{(1)}$

$$\mathbf{y}^{(1)}(t) = -\mathbf{T}^{(1)}\sum_{n=0}^{\infty}(\mathbf{T}^{(1)})^n\tilde{\mathbf{W}}_{\text{in}}^{(1)}\frac{d^{(n)}\mathbf{s}(t)}{dt^{(n)}}, \tag{22}$$

which can be inserted into Eq. 17 to give

$$\dot{\mathbf{y}}^{(2)} = \mathbf{\Lambda}^{(2)}\,\mathbf{y}^{(2)} - \tilde{\mathbf{W}}^{(12)}\,T^{(1)}\left[\tilde{\mathbf{W}}_{\text{in}}^{(1)}\mathbf{s}(t) + \sum_{n=1}^{\infty}(\mathbf{T}^{(1)})^n\tilde{\mathbf{W}}_{\text{in}}^{(1)}\frac{d^{(n)}\mathbf{s}(t)}{dt^{(n)}}\right]. \tag{23}$$

Thus the second ESN integrates the signal with a linear combination of its derivatives. In other words, the first reservoir expands the dimensionality of the signal to include information regarding the signal's derivatives (or, equivalently in discretised time, the previous values assumed by the signal). In this respect, Eq. 23 is key to understanding how the hierarchical connectivity between the two reservoirs enhances the representational capabilities of the system. The finite-difference approximation of the time derivatives appearing in Eq. 23 implies that a combination of past values of the signal appears, going back in time as much as the retained derivative order dictates.

2.2 Online learning of hyper-parameter

Selecting the hyper-parameters of such systems can be challenging. Such selection process can be informed by the knowledge of the natural timescales of the task/signal at hand. Alternatively one can resort to a learning method to optimise the parameters directly. The inherent limitation of these methods is the same as learning the network weights with BPTT: the whole history of network activations is required at once. One way to by-pass this issue is to approximate the error signal by considering only past and same-time contributions, as suggested by Bellec *et al.* [4] in their framework known as *e-prop* (see also [36]), and derive from this approximation an online learning rule for the ESN hyper-parameters. Following their approach, we end up with a novel learning rule for the leakage terms of connected ESNs that is similar to the rule proposed by Jaeger *et al.* [32] but extended to two hierarchical reservoirs. The main learning rule is given by:

$$\frac{dE}{d\alpha^{(i)}}(t) = \sum_{k=1}^{N_{\text{ESN}}} \frac{\partial E}{\partial \mathbf{x}^{(k)}(t)} \mathbf{e}^{(ki)}(t) \tag{24}$$

where $\mathbf{e}^{(ki)}(t) = d\mathbf{x}^{(k)}(t)/d\alpha^{(i)}$ is known as the eligibility trace which tracks the gradient of neuron activities in the reservoir number k with respect to the i-th leakage rate. Given the closed form for the hierarchical ESNs in Eqs. 10 and 11 these terms can be readily calculated. For our N_{ESN} sub-reservoirs in the hierarchical structure there will be N_{ESN}^2

eligibility traces to track how each sub-reservoir depends on the other leakage rates. In the hierarchical case of a fixed feed-forward structure some of these traces will be zero, and the number of non-zero eligibility traces would be $N(N+1)/2$. Since the update of the neuron's activity depends on its previous values, so do the eligibility traces; therefore, they can be calculated recursively through

$$\begin{aligned}\mathbf{e}^{(ki)}(t+\delta t) = (1-\alpha^{(k)})\mathbf{e}^{(ki)}(t) + \delta_{ki}(f(\mathbf{h}^{(k)}(t)) - \mathbf{x}^{(k)}(t)) \\ + \alpha^{(k)} f'(\mathbf{h}^{(k)}(t)) \sum_{l\neq k} \rho^{(kl)} \mathbf{W}^{(kl)} \mathbf{e}^{(li)}(t),\end{aligned} \tag{25}$$

where $\delta_{ki} = 1$ if $k = i$ and 0 otherwise, i.e the Kronecker delta. The update of equations 25 for each k-i pair needs to follow the order of dependencies given by the structure of connected reservoirs considered. The eligibility trace is an approximation that only includes same-time contributions to the gradient but has the advantage that is can be easily computed online. A complete description of our method is given in the Supplementary Material. For an example where the mean squared error function $E(t) = \frac{1}{2}\big[\tilde{y}(t) - y(t)\big]^2$ is used in a regression task and a structure composed by two reservoirs, the updating equations on the leakage terms are

$$\begin{aligned}\alpha^{(1)} &\leftarrow \alpha^{(1)} - \eta_\alpha \big[\tilde{y}(t) - y(t)\big] \mathbf{W}_{\text{out}} \begin{pmatrix} \mathbf{e}^{(11)}(t) \\ \mathbf{e}^{(12)}(t) \end{pmatrix} \\ \alpha^{(2)} &\leftarrow \alpha^{(2)} - \eta_\alpha \big[\tilde{y}(t) - y(t)\big] \mathbf{W}_{\text{out}} \begin{pmatrix} \mathbf{e}^{(21)}(t) \\ \mathbf{e}^{(22)}(t) \end{pmatrix}\end{aligned} \tag{26}$$

where η_α is the learning rate on the leakage terms and $\big(\mathbf{e}^{(k1)}(t), \mathbf{e}^{(k2)}(t)\big)$ ($k = 1, 2$ in this case with two reservoirs) is a vector composed by the juxtaposition of the eligibility traces, which can be computed through Eq. 25. Of course, the gradient can be combined with existing gradient learning techniques, among which we adopt the Adam optimiser,

described in the Supplementary Material. In all online learning simulations, training is accomplished through minibatches with updates at each time step. Training is stopped after convergence. When learning αs and the output weights simultaneously, the learning rates corresponding to these hyper-parameters need to be carefully set, since the weights need to adapt quickly to the changing dynamic of the network, but a fast convergence of $\mathbf{W}_{\text{out}}$ can trap the optimisation process around sub-optimal values of the leakage terms. For a reservoir with trained and converged output weights, a further variation of α's, even in the right direction, could correspond to an undesirable increase in the error function. We found that this problem of local minimum can be avoided by applying a high momentum in the optimisation process of α and randomly re-initialising the output weights when the α's are close to convergence. The random re-initialisation functions to keep the output weights from being too close to convergence. Thus, we defined the convergence of the algorithm for α's as when the α's do not change considerably after re-initialisation. When this happens, it is possible to turn off the learning on the leakage terms and to optimise the read-out only. More details about online training can be found in the discussions related to each task.

3 Results

The following sections are dedicated to the study of the role of timescales and the particular choices of α and ρ in various tasks, with attention on networks composed by a single ESN, 2 unconnected ESNs and 2 hierarchical ESNs. The number of trainable parameters in each task for the different models will be preserved by using the same total number of neurons in each model. The results analysed will be consequently interpreted through the analysis of timescales of the linearised systems.

3.1 NARMA10

A common test signal for reservoir computing systems is the non-linear auto-regressive moving average sequence computed with a 10 step time delay (NARMA10) [37, 38]. Here we adopt a discrete time formalism where $n = t/\delta t$ and the internal state of the reservoir is denoted as $\mathbf{x}_n = \mathbf{x}(n\delta t)$. The input, s_n, is a uniformly distributed random number in the range $[0, 0.5]$ and the output time-series is computed using

$$y_n = y_{n-1}\left(a + b\sum_{k=1}^{D} y_{n-k}\right) + cs_{n-1}s_{n-D} + d, \tag{27}$$

where $D = 10$ is the memory length, $a = 0.3$, $b = 0.05$, $c = 1.5$, and $d = 0.1$. The task for the network is to predict the NARMA10 output y_n given the input s_n. We have adapted this to also generate a NARMA5 task where $D = 5$ but the other parameters are unchanged. This provides an almost identical task but with different timescales for comparison.

The task of reconstructing the output of the NARMA10 sequence can be challenging for a reservoir as it requires both a memory (and average) over the previous 10 steps and fast variation with the current input values to produce the desired output. A typical input and output signal is shown in Fig. 3A and the corresponding auto-correlation function of the input and output in B. Since the input is a random sequence it does not exhibit any interesting features but for the output the auto-correlation shows a clear peak at a delay of 9 δt in accordance with the governing equation. For a reservoir to handle this task well it is necessary to include not only highly non-linear dynamics on a short timescale but also slower dynamics to handle the memory aspect of the task.

This regression task is solved by training a set of linear output weights to minimise the mean squared error (MSE) of the network output and true output. The predicted output is computed using linear output weights on the concatenated network activity ($\mathbf{x}_n = \left(\mathbf{x}_n^{(1)}, \mathbf{x}_n^{(2)}\right)^T$), such that

$$\tilde{y}_n = \mathbf{x}_n^T \mathbf{W}_{\text{out}} \tag{28}$$

where $\mathbf{W}$ is the weight vector of length N+1 when an additional bias unit is included. The MSE is minimised by using the ridge regression method [39] such that the weights are computed using

$$\mathbf{W}_{\text{out}} = \left(\mathbf{x}^T\mathbf{x} - \lambda\mathbf{I}\right)^{-1}\mathbf{x}^T\mathbf{y} \tag{29}$$

where $\mathbf{x}$ is a matrix formed from the activation of the internal states with a shape of number of samples by number of neurons, $\mathbf{y}$ is the desired output vector, λ is the regularisation parameter that is selected using a validation data set and $\mathbf{I}$ the identity matrix. To analyse the performance of the ESNs on the NARMA10 task we use the normalised root mean squared error as

$$\text{NRMSE} = \sqrt{\frac{1}{N_s}\frac{\sum_n^{N_s}(\tilde{y}_n - y_n)^2}{\text{Var}(\mathbf{y})}}, \tag{30}$$

where $\tilde{y}_n$ is the predicted output of the network and y_n is the true output as defined by Eq. 27.

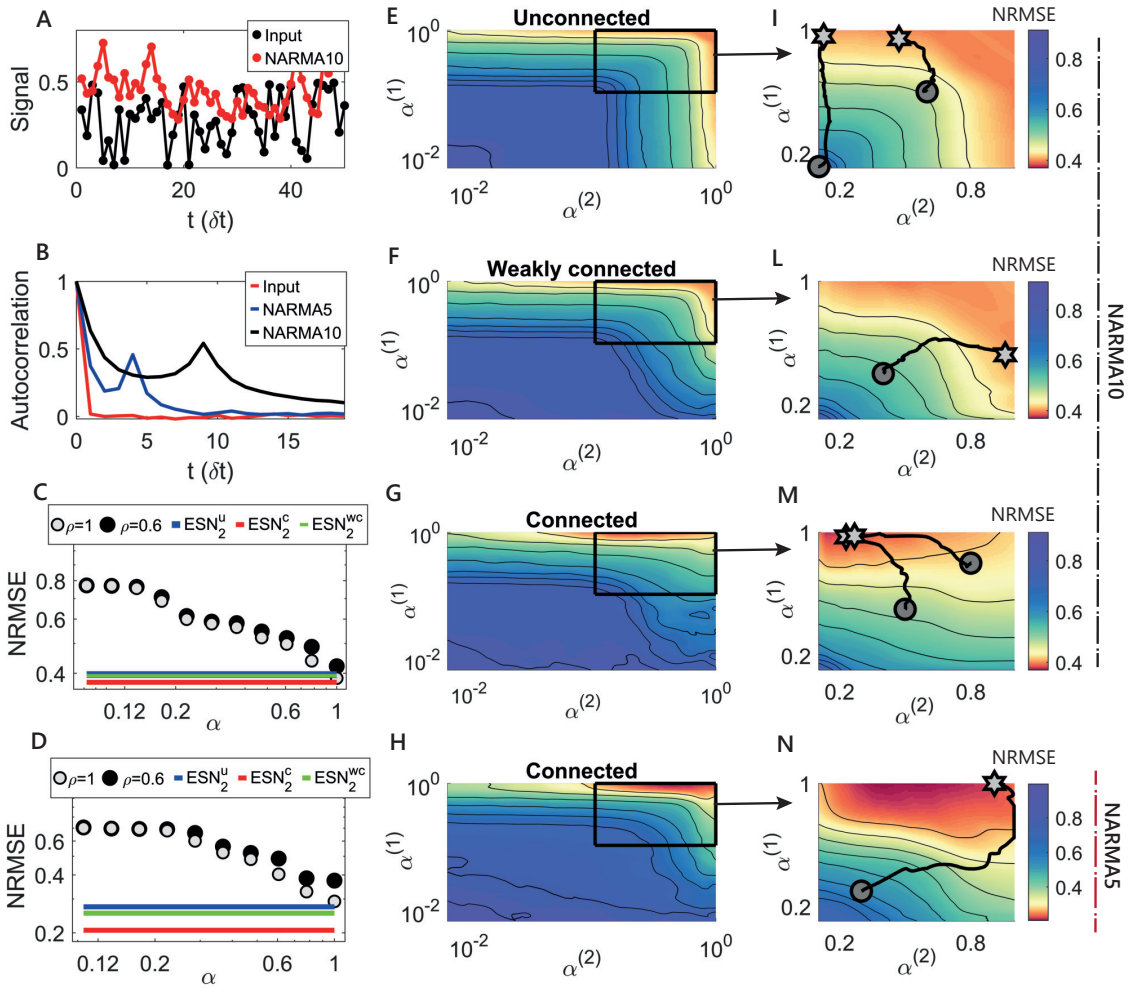

Figure 3: Performance of single or hierarchical ESNs on the NARMA10 and NARMA5 task. **A**: Example input signal (black) and desired output (red) for the NARMA10 task. **B**: The auto-correlation function of the (black) input, (red) NARMA10 and (blue) NARMA5 desired output signals, showing a second peak at about 9 delay steps for the NARMA10 and 4 for the NARMA5. **C**: The NRMSE for a single ESN for with $\rho = 1.0$ and 0.63 over a range of α. The NRMSE is lower for $\rho \approx 1$ and $\alpha = 1$. The solid lines show the minimum NRMSE for the unconnected (blue line) and connected (red line); for the unconnected case the minimum NRMSE is similar to the single ESN while the connected case has a smaller NRMSE by about 10%. **D**: Average NRMSE of a single ESN for various α compared to the hierarchical ESNs for the NARMA5 task. **E-N:** The average NRMSE surface using a hierarchical ESN computed for varying the leakage rates $\alpha^{(k)}$ of both the reservoir components for **E** and **I** (no coupling, $\rho^{(12)} = 0$), **F** and **L** (weak coupling, $\rho^{(12)} = 0.1$), and **G** and **M** (strong coupling, $\rho^{(12)} = 1$). Panels **I-N** show a close up in region for the range $\alpha^{(k)} = [0.1, 1]$ to highlight the changing behaviours. The lines on these panels show the trajectory of the $\alpha^{(k)}$ values trained directly using the online method. For each case of the coupling the online learning trends towards the approximate error minimum. **H** shows the NRMSE surface for the NARMA5 task using a strongly connected hierarchical ESN, with **N** again showing a zoom of the $\alpha = [0.1, 1]$ region. The region of best performance is with $\alpha^{(2)} \approx 0.5$ which matches the shorter timescale demonstrated in the auto-correlation in **B**.

To test the effectiveness of including multiple time-scales in ESNs, we simulate first a single ESN with $N = 100$ neurons and vary both α and ρ to alter the time-scale distribution. Secondly, we simulate a hierarchical ESN split into 2 reservoirs each with $N = 50$ neurons, where we vary $\alpha^{(1)}$ and $\alpha^{(2)}$ with $\rho^{(1)} = \rho^{(2)} = 0.95$. The input factor was set as $\gamma^{(1)} = 0.2$ and $\gamma^{(2)} = 0$ for the connected hierarchical ESN but when they are unconnected the input is fed into both, such that $\gamma^{(1)} = \gamma^{(2)} = 0.2$. In all cases the NRMSE is computed on an unseen test set and averaged over 20 initialisations of the ESN with a running median convolution is applied to the error surfaces to reduce outliers. In parallel to this we have also applied the online training method for the α hyper-parameters. The hyper-parameters used for the gradient descent learning are summarised in Table 1.

Figure 3E-G and I-M show the NRMSE depending on $\alpha^{(1)}$ and $\alpha^{(2)}$ for 3 variations of the hierarchical ESN connection strength on the NARMA10 task. In the unconnected case ($\rho^{(21)} = 0$, panels E and I), we find that the NRMSE drops by increasing both leakage rates but the minimum is when one of the leakage rates is ≈ 0.5. This is in agreement with the online learning method for the αs in shown in I but the error minimum is shallow and prone to noise in the signal or ESN structure. For the weakly connected hierarchical ESN ($\rho^{(21)} = 0.1$, panels F and L) we find again that when the sub-reservoirs have different timescales the NRMSE is reduced. In comparison to the unconnected case the error surface is asymmetric with a minimum at approximately $\alpha^{(1)} = 1.0$ and $\alpha^{(2)} \approx 0.5$. As the strength of the connection is increased ($\rho^{(21)} = 1.0$, Panel G and M), the minimum error moves to a lower leakage rate in the second reservoir ($\alpha^{(2)} \approx 0.2$) which reflects a better separation of the timescale distributions. This is a gradual effect with respect to the connection strength since stronger connection allows for a relative increase of the expanded input from the first reservoir compared to the base input signal. Since the input feeds into reservoir 1, a high α provides a transformation on the input over short time-scales, expanding the dimensionality of the signal, offering a representation that preserves much of the dynamic of the driving input and that is fed to the second reservoir. Then, since the latter does not have a direct connection to the input it performs a longer timescale

transformation of the internal states of reservoir 1. In this way the reservoirs naturally act on different parts of the task, i.e. reservoir 1 provides a fast non-linear transformation of the input while reservoir 2 follows the slower varying 10-step average of the signal, and thus returning a lower NRMSE. As a side note, we can demonstrate the validity of the theoretical analysis in Section 2.1 by replacing the first reservoir by Eq. 23 on the NARMA task (see Section 3 Supplementary Material), resulting in a similar landscape as in Fig. 3G and a similar optimal value for $\alpha^{(2)}$.

Figure 3C shows the relative performance of the single ESN to the minimum values for the unconnected (ESN_2^u) and connected (ESN_2^c) hierarchical reservoirs. The single ESN shows the similar decrease in NRMSE with increasing α and reaches a similar minimum NRMSE as the unconnected case. In comparison with the connected cases the multiple timescales provides a more optimised result. If we consider the analysis of the timescales discussed in the previous section the choice of these hyper-parameters becomes more evident. With $\alpha = 1$ the timescale distribution of the network is sharply peaked close to the minimum timescale of 1 discrete step while when $\alpha = 0.1$ this peak is broader and the peak of the distribution is closer to the second peak present in the auto-correlation function shown in Panel B. We note that whilst the most likely timescale is $\tau_{\text{peak}} \approx 6$ for $\alpha = 0.1, \rho = 0.95$ which is lower than the natural timescale of the problem, the increased width of the distribution increases the number of timescales at $\tau = 10$ dramatically which maybe why a lower α is not necessary.

To further investigate the effect of the inherent timescale of the task on the timescales we performed a similar analysis on the NARMA5 task. Figure 3H and N show the NRMSE surface for the strongly connected case. The minimum error occurs at $\alpha^{(1)} \approx 1.0$ (similar to the NARMA10 results in G and M) but $\alpha^{(2)} \approx 0.5$ (as opposed to ≈ 0.2 for NARMA10). This is due to the shorter timescales required by the NARMA5 task and the peak timescale for these values is much closer to the peak in the auto-correlation shown in B. Panel D shows the performance of the single ESN where again the optimal leakage rate is $\alpha = 1$ and similar to the unconnected cases but the NRMSE is higher than the connected cases.

In this theoretical task where the desired output is designed *a priori*, the memory required and the consequent range of timescales necessary to solve the task are known. Consequently, considering the mathematical analysis in section 2.1, and that for hierarchical ESNs the timescales of the first ESN should be faster than those of the second Fig. 3), the best-performing values of the leakage terms can be set *a priori* without the computationally expensive grid search reported in Fig. 3E-I. However, it can be difficult to guess the leakage terms in the more complex cases where the autocorrelation structure of the signal is only partially informative of the timescales required.

This problem can be solved using the online learning approach defined through Eq. 24. In this case, learning is accomplished through minibatches and the error function can be written explicitly as

$$E(t) = \frac{1}{2\mathrm{N}_{\mathrm{batch}}} \sum_{m=1}^{\mathrm{N}_{\mathrm{batch}}} \left[\tilde{y}(t,m) - y(t,m)\right]^2 \tag{31}$$

where $\mathrm{N}_{\mathrm{batch}}$ is the minibatch size and m is its corresponding index. A minibatch is introduced artificially by dividing the input sequence into $\mathrm{N}_{\mathrm{batch}}$ signals or by generating different NARMA signals. Of course, the two methods lead to equivalent results if we assure that the $\mathrm{N}_{\mathrm{batch}}$ sequences are temporally long enough. A learning rate $\eta_\alpha/\eta_W \approx 10^{-2} - 10^{-3}$ was adopted. The optimiser used for this purpose is Adam, with the suggested value of $\beta_1 = 0.9$ adopted for the output weights and a higher first momentum $\beta_1 = 0.99$ adopted for the leakage terms. Instead, we set $\beta_2 = 0.999$ of the second momentum for both types of parameters (See section 5.1 for a description of the updating rules). Panels I-N show a zoomed in region of the error surface with the lines showing the online training trajectory of the α hyper-parameters. In each case the trajectory is moving towards the minimum NRMSE of the α phase space.

Learning hyper-parameters		
	NARMA/Telegraph	**psMNIST**
Network size, N	100	1200
Minibatch size, N_{batch}	10	50
	Learning $\mathbf{W}_{out}$	
η_W	10^{-3}	$5e-2\,[5e-3]^{\dagger}$
β_1	0.9	0.9
β_2	0.999	0.999
ϵ	10^{-8}	10^{-8}
	Learning α	
η_α	5×10^{-6}	10^{-4}
β_1	0.99	0.999
β_2	0.999	0.999
ϵ	10^{-8}	10^{-8}

Table 1: Table of the hyper-parameters adopted in the online learning process. η is the learning rate in each case, while β_1, β_2 and ϵ are parameters for the Adam optimiser (further details are given in the Supplementary Material). The † symbol indicates that the learning rate $5e-2$ is for the case with 4 hidden states, while the learning rate $5e-3$ is for the case with 28 hidden states. This decrease of η is due to the increase in the dimensionality of the representation for the latter case in comparison to the situation where the read-out is composed by four concatenated values of activity. Furthermore, such learning rates are 10 times higher than the case in which only the read-out is trained (only in the psMNIST task). Thus, the high learning rate adopted has the purpose to introduce noise in the learning process and to avoid local minima in the complex case where α and $\mathbf{W}_{out}$ are optimised simultaneously.

3.2 A volatile environment

We now turn to study the reservoir performance on a task of a telegraph process in a simulated noisy environment. The telegraph process $s^{(1)}(t)$ has two states that we will call *up* (1) and *down* (0), where the probability of going from a *down* state to an *up* state $p(s=1|s=0)$ (or the opposite $p(s=0|s=1)$) is fixed for any time step. The environment is also characterised by a telegraph process $s^{(2)}(t)$, but the transition probability is much lower and controls the transition probability of the first signal. To simplify the notation in the following we denote the probability of the signal i transitioning from state a to state b as $P(s^{(i)}(t)=a|s^{(i)}(t-\delta t)=b) = p^{(i)}_{ab}(t)$. The signal taken under consideration is then composed by a fast telegraph process with probabilities $p^{(1)}_{01}(t)$ and $p^{(1)}_{10}(t)$, whose values are interchanged by following the dynamic of a slower telegraph process $s^{(2)}(t)$. Every

time the slower environment signal changes its state, the probabilities of the first signal are changed, i.e $p_{01}^{(1)}(t) \leftrightarrow p_{10}^{(1)}(t)$. The resulting signal is then characterised by

$$p_{10}^{(1)}(t) = \begin{cases} p_1, & \text{if } s^{(2)}(t) = 0 \\ p_2, & \text{if } s^{(2)}(t) = 1 \end{cases} \tag{32}$$

$$p_{01}^{(1)}(t) = \begin{cases} p_2, & \text{if } s^{(2)}(t) = 0 \\ p_1, & \text{if } s^{(2)}(t) = 1 \end{cases} \tag{33}$$

The transition probabilities of the second signal are fixed and symmetric such that

$$p_{01}^{(2)}(t) = p_{10}^{(2)}(t) = p_3, \tag{34}$$

The probabilities p_1, p_2 and p_3 are fixed parameters of the signal that define the process. Given that the second signal controls the probabilities of the first telegraph process, we say that it defines the regime of the input, while we refer to the *up* and *down* values of the first process simply as states. Thus, the reconstruction of $s^{(1)}(t)$ from the input will be called state reconstruction, while reconstruction of $s^{(2)}(t)$ will be called regime reconstruction. These reconstructions can be considered separately or as a joint task requiring the system to be modeled on long and short timescales simultaneously. Due to the probability transition caused by $s^{(2)}(t)$, both states and regime will be equally present over a infinitely long signal. The values adopted for the simulation are $p_1 = 0.05$, $p_2 = 0.1$ and $p_3 = 0.0005$.

The input signal corresponds to $s^{(1)}(t) + \sigma\mathcal{N}(0, 1)$, that is the faster telegraph process with additional white noise. The input signal constructed is a metaphor of a highly stochastic environment with two states and two possible regimes that define the probability of switching between the two states. The reservoir will be asked to understand in which state ($s^{(1)}(t) = 1$ or 0) and/or regime ($s^{(2)}(t) = 1$ or 0) it is for each time t, measuring

the understanding of the model to estimate the state of the input signal. The input signal and telegraph processes is shown in Fig. 4A, while the B shows the corresponding auto-correlation structure of the processes. The auto-correlation shows that the input has a temporal structure of around 10 δt while the slow 'environment' process has a structure close to 1000 δt. This corresponds directly to the timescales defined by the probabilities of the signals.

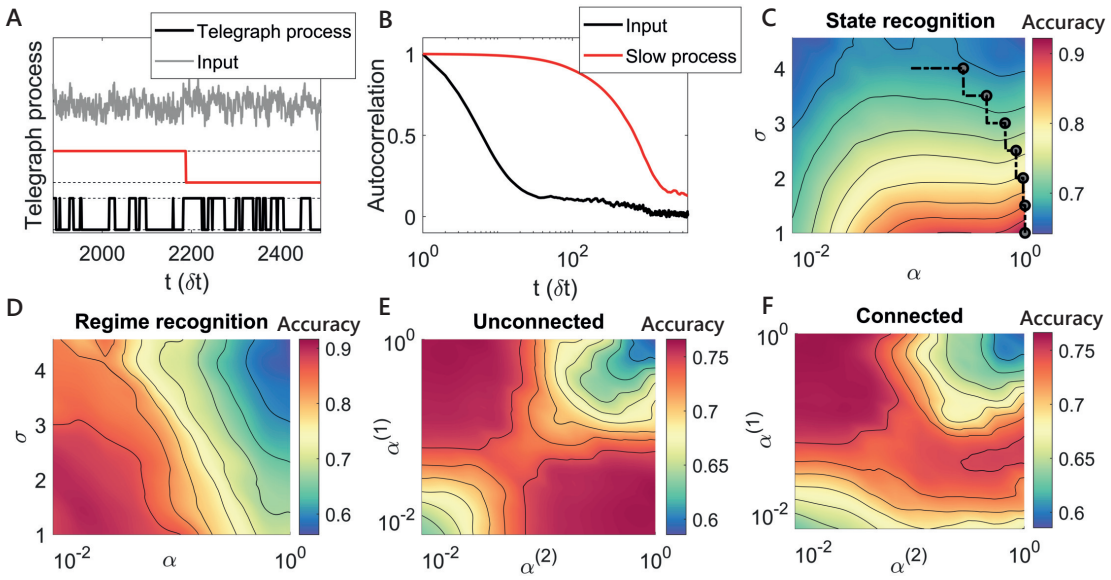

Figure 4: The best structure and parameters of the model depend on the specific environment considered, that is different values of the additive noise in the input signal, and on the specific desired output. **A**: Example of input signal and of its generative processes, which have a faster and a slower dynamic respectively. When the slower process (red line) is *up* (*down*), the other signal is in a regime where the average time in the zero (one) state is greater than the average time spent in the other state. The input signal (grey line) corresponds to the faster process (black line) with additional white noise. **B**: Auto-correlation structure of the two generative processes. **C**: The accuracy surface for a single ESN on the state recognition sub-task for varying level of noise (σ) and leakage rate of the network showing that for increasing levels of noise a lower leakage rate is needed to determine the state. The line shows the trajectory of α using the online learning method when the strength of the noise is changed. **D**: The accuracy for a single ESN on the regime recognition sub-task for varying noise and leakage rate. In this case the low leakage rate is preferred for all values of noise. **E**: Accuracy surface for the state recognition sub-task for an unconnected hierarchical ESN showing how either of the leakage rates must be low while the other is high. **F**: Accuracy surface for the regime recognition sub-task for a hierarchical ESN showing the first reservoir must have a high leakage rate and the second a low leakage rate.

Panels C and D of Fig. 4 show the performance of a single ESN when it is tasked to reconstruct the processes $s^{(1)}(t)$ (state recognition) and $s^{(2)}(t)$ (regime recognition)

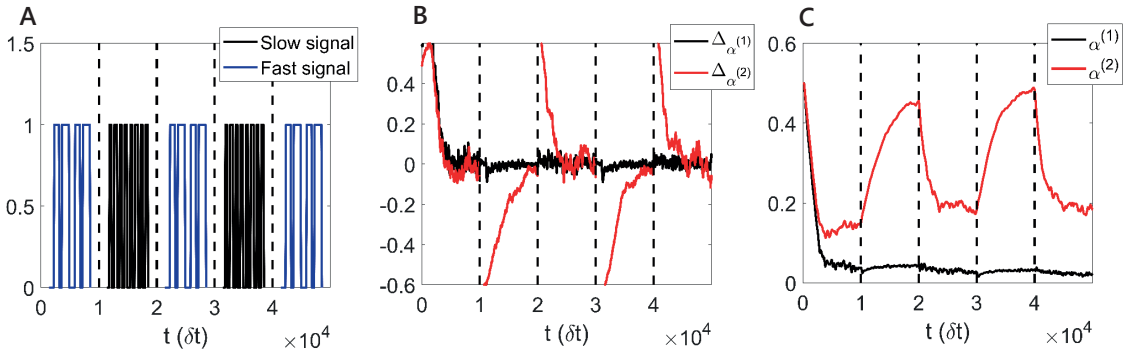

Figure 5: The online training of the leakage terms can adapt to the changing environment, that is the signal probabilities are increased or decreased periodically. **A**: Scheme of the change of the values of probabilities, where high probabilities of switching are referred to as fast phase of the telegraph process, while low probabilities as slow phase. **B**: Running average of the gradients of $\alpha^{(1)}$ and $\alpha^{(2)}$ as time varies. **C**: Online adaptation of the leakage terms.

respectively. In this simulation, learning is always accomplished online and the error function is the same as Eq. 31. First, panel C demonstrates how the leakage term, α, must be tuned to the level of noise of the environment, and how lower values of α are desirable for noisier signals, in order to solve the state recognition problem. Indeed, the need to smooth the fluctuations of the input signal increases with σ, while for low values of noise the network should simply mimic the driving input. Second, panel D shows how the desirable values of α must be lower in the case where the network is asked to reproduce the slower dynamic of $s^{(2)}(t)$ independently of having to output the fast signal, in order to solve the regime recognition problem. This result exemplifies how the timescales of the network must be tuned depending on the desired output. It demonstrates that, even in this relatively simple environment, it is crucial to adopt multiple timescales in the network to obtain results that are robust with respect to a variation of the additional white noise σ.

Finally, panels E and F of Fig. 4 show the accuracy of two unconnected (E) and connected (F) reservoirs when the network has to classify the state and the regime of the input signal at the same time. In this case, the desired output corresponds to a four dimensional signal that encodes all the possible combinations of states and regimes; for

instance, when the signal is in the state one and in the regime one, we would require the first dimension of the output to be equal to one and all other dimensions to be equal to zero, and so on. The best performance occurs when one leakage term is high and the other one is low and in the range of significant delays of the auto-correlation function. This corresponds to one network solving the regime recognition and the other network solving the state recognition. For the unconnected reservoirs, it does not matter which reservoir has high vs. low leakage terms, reflected by the symmetry of Fig. 4E, while for the connected reservoirs, the best performance occurs when the first reservoir has the high leakage term and the second the low leakage terms, see Fig. 4F, similar to Fig. 3. Both two-reservoir networks can achieve accuracy 0.75, but the single ESN can not solve the task efficiently, since it cannot simultaneously satisfy the need for high and low αs, reporting a maximum performance of about 0.64.

The path reported in panel C of Fig. 4 and all panels in Fig. 5 show the application of the online training algorithm in this environment. The values of the hyper-parameters adopted in the optimisation process through the Adam optimiser are the same as in section 3.1, where we used a slower learning rate and a higher first momentum on the leakage terms in comparison to the values adopted for the output weights. The line of panel C (Fig. 4) shows the online adaptation of α for a simulation where the external noise increases from one to four with six constant steps of 0.5 equally spaced across the computational time of the simulation. The result shows how the timescales of the network decrease for each increase in σ, depicted with a circle along the black line. The path of online adaptation reports a decrease of the α value for noisier external signals. This result occurs because as the signal becomes noisier (σ rises), it becomes more important to dampen signal fluctuations. This result also shows that the online algorithm can adapt in environments with varying signal to noise ratio. Panels A, B, C of Fig. 5 show the online training of $\alpha^{(1)}$ and $\alpha^{(2)}$ for an environment composed by a faster and a slower composition of telegraph processes. This specific simulation is characterised by the alternation of two signals defined by Eq. 32, 33 and 34, each with different values of p_1 and p_2. In

particular, while $p_1 = 0.5$ and $p_2 = 0.1$ for the 'fast' phase of the external signal, $p_1 = 0.1$ and $p_2 = 0.05$ for the 'slow' phase. In contrast, the slower timescale of the task defined by $p_3 = 0.0005$ remains invariant across the experiment. Panel C shows the adaptation of the leakage terms for this task in the case of a hierarchical structure of ESNs. While $\alpha^{(2)}$ adapts to the change of p_1 and p_2 following the transition between the two phases of the external signals, the relatively constant value of $\alpha^{(1)}$ indicates how the first network sets its timescales to follow the slower dynamic of the signal, characterised by the constant value of p_3. Thus, the composed network exploits the two reservoirs separately, and the first (second) reservoir is used to represent the information necessary to recognise the regime (state) of the external signal.

3.3 Permuted Sequential MNIST

The Permuted Sequential MNIST (psMNIST) task is considered a standard benchmark for studying the ability of recurrent neural networks to understand long temporal dependencies. The task is based on the MNIST dataset, which is composed of $60,000$ handwritten digits digitised to 28x28 pixel images. In the standard MNIST protocol every pixel is presented at the same temporal step so a machine has all the information of the image available at once and needs to classify the input into one out of ten classes. In contrast, in the psMNIST task, the model receives each pixel sequentially once at a time, so that the length of the one dimensional input sequence is 784. Thus, the machine has to rely on its intrinsic temporal dynamic and consequent memory ability to classify the image correctly. Furthermore, each image in the dataset is transformed through a random permutation of its pixels in order to include temporal dependencies over a wide range of input timescales and to destroy the original images' structure. Of course, the same permutation is applied on the entire dataset. The performance of ESNs on the MNIST dataset, where each columns of pixels in a image is fed to the network sequentially (each image corresponds to a 28 dimensional signal of length 28 time steps), has been analysed in [40] and in [41].

In [40] the original dataset was preprocessed through reshaping and rotating the original image to enhance the network's ability to understand high level features of the data. In this case, the original dataset is used. In [41], the addition of thresholds and the introduction of sparse representation in the read-out of the reservoir was used to improve the performance of the network in the online learning of the standard MNIST task through reservoir computing. This section is focused on the analysis of the performance of ESNs on the psMNIST task and on their dependence on the range of timescales available in the network, i.e. the values of α and ρ chosen. In contrast to the previous sections where ESNs are trained through ridge regression, we have applied an online gradient descent optimisation method. The cost function chosen to be minimised is the cross entropy loss

$$E = -\frac{1}{\mathrm{N_{batch}}} \sum_{m=1}^{\mathrm{N_{batch}}} \sum_{j=1}^{\mathrm{N_{class}}} \Big[y_j(m) \log\left(\tilde{y}_j(m)\right) + \left(1 - y_j(m)\right) \log\left(1 - \tilde{y}_j(m)\right) \Big], \tag{35}$$

where m is the minibatch index, $\mathrm{N_{batch}}$ corresponds to the minibatch size and $\mathrm{N_{class}}$ is the number of classes. For this task the desired output, y_j, is a one-hot encoded vector of the correct classification while the desired output is a sigmoid function of the readout of the reservoir nodes. Furthermore, instead of reading out the activity of the reservoir from the final temporal step of each sequence only, we have expanded the reservoir representation by using previous temporal activities of the network. In practice, given the sequence of activities $\mathbf{x}(0), \mathbf{x}(\delta t), ..., \mathbf{x}(\delta t T)$ ($T = 784$) that defines the whole temporal dynamic of the network subjected to an example input sequence, we trained the network by reading out from the expanded vector $\mathbf{X} = \left[\mathbf{x}(M\delta t), \mathbf{x}(2M\delta t), ..., \mathbf{x}(T\delta t)\right]$, where M defines the 'time frame' used to sample the activities of the evolution of the system across time.

$$\tilde{\mathbf{y}} = \mathrm{sigm}\left(\sum_{n=1}^{T/M} \mathbf{W}_{\mathrm{out}}^{(n)} \mathbf{x}(nM\delta t) \right) \tag{36}$$

, where sigm stands for sigmoid activation function. We then repeat the simulation for two different time frames of sampling for each different model, that is a single ESN and a pair of unconnected or connected ESNs, as in the previous sections.

The two values of M used are 28 and 196, corresponding to a sampling of 28 and 4 previous representations of the network respectively. Of course, a higher value of M corresponds to a more challenging task, since the network has to exploit more its dynamic to infer temporal dependencies. We note, however, that none of the representation expansions used can guarantee a good understanding of the temporal dependencies of the task, or in other words, can guarantee that the system would be able to discover higher order features of the image, considering that these features depend on events that could be distant in time.

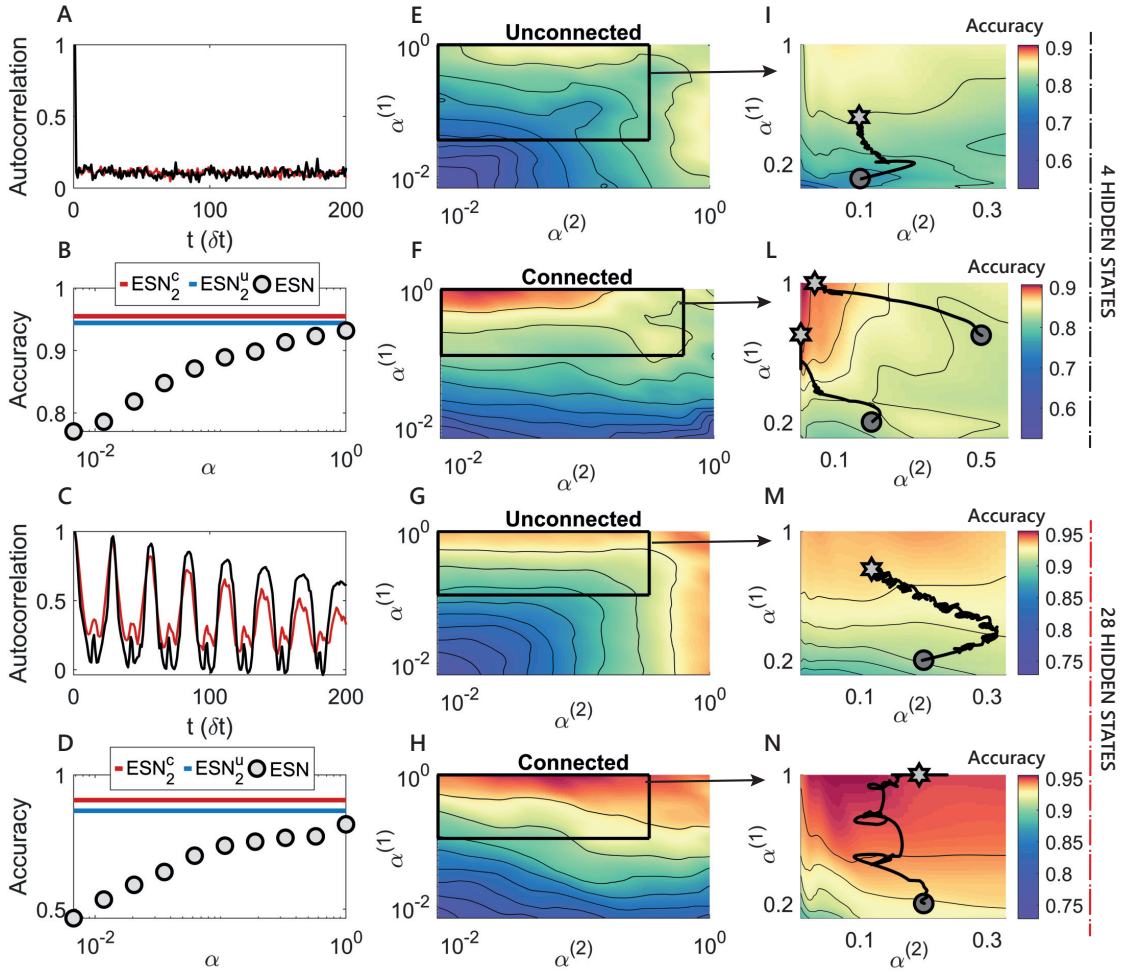

Figure 6: The additional non linearity added by the hierarchical reservoir structure is responsible for a relevant modification and increase of the performance surface. **A,C**: Auto-correlation structure of the MNIST dataset for two examples of digits, where each pixel is presented one after the other (**C**), and auto-correlation structure of the data after the random permutation(**A**). The oscillatory trend in **C** reflects the form of the written digits, when this is seen one pixel after the other. The auto-correlation function of the permuted data is low, but not negligible, for all the temporal steps, showing the necessity to have a wide repertoire of timescales in the interval corresponding to the image size. **B,D**: Accuracy of a single ESN for various α values compared the maximum accuracy of the hierarchical ESNs with 4 hidden states (**B**) or 28 hidden states (**D**). **E-F**: Case with low sampling frequency of the ESNs which corresponds to a higher demand of internal memory in the reservoir. While the best region of accuracy for the unconnected reservoirs is characterised by intermediate values of the leakage factors, the hierarchically connected network structure reports the best performance when the second network has slower dynamics. **G-H**: The utilisation of a high sampling frequency alleviates the need for long term memory, and the reservoirs prefer the regions with fast timescales. In both cases analysed, the additional complexity of the hierarchical model leads to a considerable boost in performance. **I-N**: Paths (black line, starting from the circle and ending in the star) that describe the online changes of the leakage terms achieved through the online training algorithm in a zoomed region of the performance surface of $\alpha^{(1)}$ and $\alpha^{(2)}$. The paths are smoothed through a running average.

In Fig. 6 we again analyse the performance of two connected or unconnected ESNs varying $\alpha^{(1)}$ and $\alpha^{(2)}$ for both $M = 28$ and 196. In contrast to the previous sections, we now use gradient descent learning on the output weights instead of ridge regression and increase the total number of neurons in each model to $N = 1200$ due to the complexity of the task. The Adam optimiser is used; its parameters, for both the output weights and α learning, are in Table 1. As previously, we have trained the output weights over a range of fixed αs and report the performance on an unseen test data set. In parallel to this we have trained both the output weights and α values which, as shown by the lines on the contour plots, converge towards the minimum computed using the fixed α's.

As in the other simulations, we found that the values of ρ corresponding to the best performance was approximately one, which maximises the range of timescales and the memory available in the network. Fig. 6E-F shows the case with $M = 28$, while Fig. 6G-H reports the accuracy for the simulation with $M = 196$ where E and G are unconnected and F and H connected reservoirs. The accuracy surface demonstrates how, in the case of the unconnected ESNs with a fast sampling rate in panel G, the best performance is achieved when at least one of the two values of α is close to one. The result is due to the fast changing dynamic of the temporal sequence that is introduced through the random permutation of the pixels. On the contrary, in the case of the unconnected ESNs with a slow sampling rate in panel E the best accuracy is in a range of intermediate timescales since both partitions must respond to both fast and slow timescales.

This relatively simple behaviour of the dependence of the accuracy on the setting of the hyper-parameters changes in the cases of two connected ESNs, whose additional complexity corresponds to a considerable increase in the performance. Fig. 6H reports how the network prefers a regime with a fast timescale in the first reservoir and a intermediate timescale in the second, which acts as an additional non-linear temporal filter of the input provided by the first network. The need of memory of events distant in time is emphasised in 6F, where the best performing network is composed by reservoirs with fast and slow dynamics respectively. The performance boost from the panels E-G to the ones

F-H has only two possible explanations: first, the timescales of the second network are increased naturally thanks to the input from the first reservoir; second, the connections between the two reservoirs provide an additional non-linear filter of the input that can be exploited to discover higher level features of the signal. Thus, we can conclude once again that achieving high performance in applying reservoir models requires (1) additional non-linearity introduced through the interconnections among the reservoirs and (2) an appropriate choice of timescales, reflecting the task requirements in terms of external signal and memory.

Panels I, L, M and N show the application of the online training of αs for the various cases analysed. In the psMNIST task we found that the major difficulties in the application of an iterative learning rule on the leakage terms are: the possibility to get trapped in local minima, whose abundance can be caused by the intrinsic complexity of the task, the intrinsic noise of the dataset, the randomness of the reservoir and of the applied permutation; the high computational time of a simulation that exploits an iterative optimisation process on αs arising from a practical constraint in the implementation. Indeed, while the activities of the reservoir can be computed once across the whole dataset and then saved in the case of untrained values of αs, the activities of the nodes need to be computed every time the leakage terms change in the online learning paradigm. However, we found that using a higher learning rate η_W on the output weights, compared to the value adopted in the paradigm where the leakage terms are not optimised (as in Panels E, F, G and H), can introduce beneficial noise in the learning process and help to avoid local minima. Furthermore, a higher value of the learning rate on the output weights corresponds to an increased learning rate on the thresholds, as shown from Eq. 43 and from the dependence of the updating equations on $\mathbf{W}_{\text{out}}$. As in the previous simulations of Sections 3.1 and 3.2, the output weights are randomly reinitialised after the convergence of αs, helping the algorithm to avoid an undesirable quick convergence of weights. The online process is then ended when the leakage terms remain approximately constant even after the re-initialisation. Following this computational recipe, it possible to avoid the

difficulties found and train the leakage terms efficiently.

Finally, we note how the best accuracy of 0.96 reached throughout all the experiments on the psMNIST is comparable to the results obtained by recurrent neural networks trained with BPTT, whose performance on this task are analysed in [42] and can vary from 0.88 to 0.95. In comparison to recurrent structures trained through BPTT, a network with two interacting ESNs provide a cheap and easily trainable model. However, this comparison is limited by the necessity of recurrent neural networks to carry the information from the beginning to the end of the sequence, and to use the last temporal state only or to adopt attention mechanisms.

4 Conclusion

In summary, ESNs are a powerful tool for processing temporal data, since they contain internal memory and time-scales that can be adjusted via network hyper-parameters. Here we have highlighted that multiple internal time-scales can be accessed by adopting a split network architecture with differing hyper-parameters. We have explored the performance of this architecture on three different tasks: NARMA10, a benchmark composed by a fast-slow telegraph process and PSMNIST. In each task, since multiple timescales are present the hierarchical ESN performs better than a single ESN when the two reservoirs have separate slow and fast timescales. We have demonstrated how choosing the optimal leakage terms of a reservoir can be aided by the theoretical analysis in the linear regime of the network, and by studying the auto-correlation structure of the input and/or desired output and the memory required to solve the task. The theoretical analysis developed needs to be considered as a guide for the tuning of the reservoir hyper-parameters, and in some specific applications it could be insufficient because of the lack of information about the nature of the task. In this regard, we showed how to apply a data-driven online learning method to optimise the timescales of reservoirs with different structures, demonstrating its ability to find the operating regimes of the network that correspond to

high performance and to the best, task-dependent, choice of timescales. The necessity of adopting different leakage factors is emphasised in the case of interactive reservoirs, whose additional complexity leads to better performance in all cases analysed. Indeed, the second reservoir, which acts as an additional non linear filter with respect to the input, is the perfect candidate to discover higher temporal features of the signal, and it consequently prefers to adopt longer timescales in comparison to the first reservoir, which has instead the role of efficiently representing the input. We believe such hierarchical architectures will be useful for addressing complex temporal problems and there is also potential to further optimise the connectivity between the component reservoirs by appropriate adaptation of the online learning framework presented here.

5 Appendix

5.1 Online Learning

The online learning method formulated is similar to the approach followed in e-prop by [4] (see also [36]), a local learning rule for recurrent neural networks that exploits the concept of an eligibility trace, and in [32]. As in these previous works, we approximated the error function to neglect the impact that the instantaneous and online changes of the network's parameters have on future errors. In particular, considering a recurrent neural network as the one depicted in the computational graph in Fig. 7A and the dependencies of the error function E on the activities $\mathbf{x}(t)$

$$\begin{aligned}
\frac{dE}{d\alpha} &= \sum_t \frac{dE}{d\mathbf{x}(t)} \frac{d\mathbf{x}(t)}{d\alpha} \\
&= \sum_t \left\{ \frac{\partial E(t)}{\partial \mathbf{x}(t)} + \frac{\partial E(t+1)}{\partial \mathbf{x}(t+1)} \frac{\partial \mathbf{x}(t+1)}{\partial \mathbf{x}(t)} + \frac{\partial E(t+2)}{\partial \mathbf{x}(t+2)} \frac{\partial \mathbf{x}(t+2)}{\partial \mathbf{x}(t+1)} \frac{\partial \mathbf{x}(t+1)}{\partial \mathbf{x}(t)} + ... \right\} \frac{d\mathbf{x}(t)}{d\alpha} \\
&= \sum_t \left\{ \sum_{t' \geq t} \frac{\partial E(t')}{\partial \mathbf{x}(t')} \mathcal{J}_{t't} \right\} \frac{d\mathbf{x}(t)}{d\alpha}, \qquad (37) \\
\mathcal{J}_{t't} &= \frac{\partial \mathbf{x}(t')}{\partial \mathbf{x}(t'-1)} \cdots \frac{\partial \mathbf{x}(t+1)}{\partial \mathbf{x}(t)} \qquad (38)
\end{aligned}$$

Eq. 37 and 38 define the algorithm back-propagation through time, where the dependencies of $\frac{dE}{d\mathbf{x}(t)}$ on activities at future time t' do not permit the definition of an online learning rule. As in the works of [32] and [4] we decided to adopt the following approximation

$$\begin{aligned}
\frac{dE}{d\alpha} &= \sum_t \left\{ \sum_{t' \geq t} \frac{\partial E(t')}{\partial \mathbf{x}(t')} \mathcal{J}_{t't} \right\} \frac{d\mathbf{x}(t)}{d\alpha} \\
&\approx \sum_t \frac{\partial E(t)}{\partial \mathbf{x}(t)} \frac{d\mathbf{x}(t)}{d\alpha} \qquad (39)
\end{aligned}$$

We will now derive the equations defining the iterative learning approach for the example cost function

$$E(t) = \frac{1}{2}\big[\tilde{y}(t) - y(t)\big]^2 \tag{40}$$

where $\tilde{y}$ is the desired output and $y = \mathbf{W}_{\text{out}}\mathbf{x}(t)$ is the output of the ESN. Then, we desire to compute $\partial E/\partial \alpha^{(k)}$, which describes the leakage term k for a network compose by multiple reservoirs. In particular, the case of two connected ESNs in considered and analysed here, while the more general case with N interacting ESNs can be easily derived following the same approach. In this case, the vector of activities $\mathbf{x}(t) = \big(\mathbf{x}_1(t), \mathbf{x}_2(t)\big)$ is composed by the juxtaposition of the vectors of activities of the two reservoirs.

$$\mathbf{x}^{(1)}(t+1) = (1-\alpha^{(1)})\mathbf{x}^{(1)}(t) + \alpha^{(1)} f\big[\mathbf{W}_{\text{in}}\mathbf{s}(t) + \mathbf{W}^{(11)}\mathbf{x}^{(1)}(t)\big] \tag{41}$$

$$\mathbf{x}^{(2)}(t+1) = (1-\alpha^{(2)})\mathbf{x}^{(2)}(t) + \alpha^{(2)} f\big[\mathbf{W}^{(21)}\mathbf{x}^{(1)}(t) + \mathbf{W}^{(22)}\mathbf{x}^{(2)}(t)\big] \tag{42}$$

$$\begin{aligned}
\frac{dE(t)}{d\alpha^{(1)}} &= -\big[\tilde{y}(t) - y(t)\big]\mathbf{W}_{\text{out}} \begin{pmatrix} \dfrac{d\mathbf{x}^{(1)}(t)}{d\alpha^{(1)}} \\ \dfrac{d\mathbf{x}^{(2)}(t)}{d\alpha^{(1)}} \end{pmatrix} \\
\frac{dE(t)}{d\alpha^{(2)}} &= -\big[\tilde{y}(t) - y(t)\big]\mathbf{W}_{\text{out}} \begin{pmatrix} \dfrac{d\mathbf{x}^{(1)}(t)}{d\alpha^{(2)}} \\ \dfrac{d\mathbf{x}^{(2)}(t)}{d\alpha^{(2)}} \end{pmatrix}
\end{aligned} \tag{43}$$

$$\frac{d\mathbf{x}^{(1)}(t)}{d\alpha^{(1)}} = (1-\alpha^{(1)})\frac{d\mathbf{x}^{(1)}(t-1)}{d\alpha^{(1)}} - \mathbf{x}^{(1)}(t-1) + \\ + \alpha^{(1)} f'\big[\mathbf{W}_{\text{in}}\mathbf{s}(t) + \mathbf{W}^{(11)}\mathbf{x}^{(1)}(t-1)\big]\mathbf{W}^{(11)}\frac{d\mathbf{x}^{(1)}(t-1)}{d\alpha^{(1)}} + \\ + f\big[\mathbf{W}_{\text{in}}\mathbf{s}(t) + \mathbf{W}^{(11)}\mathbf{x}^{(1)}(t-1)\big] \tag{44}$$

$$\frac{d\mathbf{x}^{(2)}(t)}{d\alpha^{(2)}} = (1-\alpha^{(2)})\frac{d\mathbf{x}^{(2)}(t-1)}{d\alpha^{(2)}} - \mathbf{x}^{(2)}(t-1) + \\ + \alpha^{(2)} f'\big[\mathbf{W}^{(21)}\mathbf{x}^{(1)}(t-1) + \mathbf{W}^{(22)}\mathbf{x}^{(2)}(t-1)\big]\mathbf{W}^{(22)}\frac{d\mathbf{x}^{(2)}(t-1)}{d\alpha^{(2)}} + \\ + f\big[\mathbf{W}^{(21)}\mathbf{x}^{(1)}(t-1) + \mathbf{W}^{(22)}\mathbf{x}^{2)}(t-1)\big] \tag{45}$$

$$\frac{d\mathbf{x}^{(1)}(t)}{d\alpha^{(2)}} = 0 \tag{46}$$

$$\frac{d\mathbf{x}^{(2)}(t)}{d\alpha^{(1)}} = (1-\alpha^{(2)})\frac{d\mathbf{x}^{(2)}(t-1)}{d\alpha^{(1)}} + \\ + \alpha^{(2)} f'\big[\mathbf{W}^{(21)}\mathbf{x}^{(1)}(t-1) + \mathbf{W}^{(22)}\mathbf{x}^{(2)}(t-1)\big]\big[\mathbf{W}^{(22)}\frac{d\mathbf{x}^{(2)}(t-1)}{d\alpha^{(1)}} + \\ + \mathbf{W}^{(21)}\frac{d\mathbf{x}^{(1)}(t-1)}{d\alpha^{(1)}}\big] \tag{47}$$

That can be computed online tracking the eligibility traces $\frac{d\mathbf{x}^{(1)}(t)}{d\alpha^{(1)}} = \mathbf{e}^{(11)}(t)$, $\frac{d\mathbf{x}^{(2)}(t)}{d\alpha^{(1)}} = \mathbf{e}^{(21)}(t)$, $\frac{d\mathbf{x}^{(2)}(t)}{d\alpha^{(2)}} = \mathbf{e}^{(22)}(t)$ and updating them in an iterative way. Of course, for the more general case of N connected reservoirs, the number of eligibility traces to be computed would be N^2. We note how the differences between the connected and unconnected reservoirs are: $\mathbf{e}^{(21)}(t) = 0$ in the latter case, since the activity of the second reservoir does not depend on the activities of the first; $\mathbf{e}^{(22)}(t)$ would have an analogous expression to $\mathbf{e}^{(11)}(t)$ in the case of unconnected reservoirs.

In order to understand the meaning of the approximation in Eq. 39, we can consider

the psMNIST task defined in section 3.3, in which two different numbers of previous hidden states are used for classification. In this example, the future terms t' from which $\frac{dE}{d\mathbf{x}(t)}$ depends correspond to the concatenated temporal steps $\{t_l\}_{l=1,\dots,N_{conc}}$ used for the readout. Following the computational graph in panel B of Fig. 7 , the approximation of BPTT is

$$\begin{aligned}\frac{dE}{d\alpha} &= \sum_l \left\{ \sum_{q \geq l} \frac{\partial E(t_q)}{\partial \mathbf{x}(t_q)} \mathcal{J}_{t_q t_l} \right\} \frac{d\mathbf{x}(t_l)}{d\alpha} \\ &\approx \sum_l \frac{\partial E(t_l)}{\partial \mathbf{x}(t_l)} \frac{d\mathbf{x}(t_l)}{d\alpha} \end{aligned} \tag{48}$$

where the contribution of the terms corresponding to $\sum_{q>l} \frac{\partial E(t_q)}{\partial \mathbf{x}(t_q)} \mathcal{J}_{t_q t_l}$ are neglected. The number of these terms increases as the number of hidden states used to define the read-out rises, and the contribution of the matrices $\mathcal{J}_{t_q t_l}$ becomes more important when the hidden states utilised are in closer proximity. Thus, the approximation used to define the online training algorithm is less precise for an increasing number of hidden states used. This consideration can be observed in Panels C and D of Fig. 7, in which the values of the gradients are compared to those given by BPTT for the two different numbers of concatenated values adopted in Section 3.3.

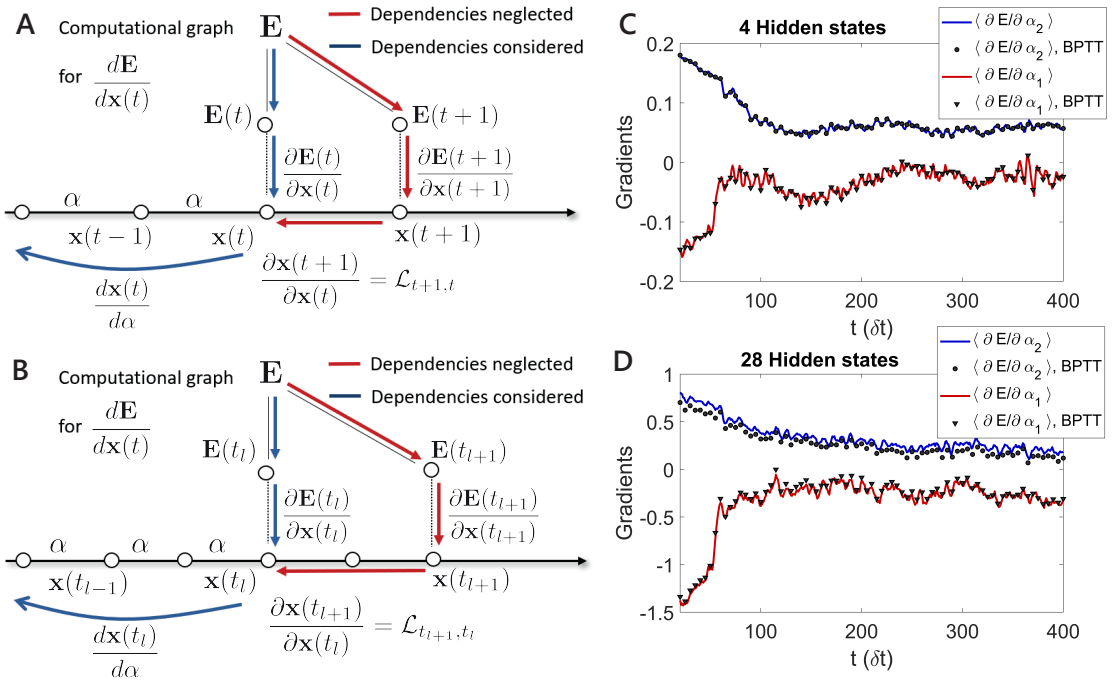

Figure 7: The online training algorithm used [4] maintains the majority of the temporal information of BPTT, while the approximation becomes less precise as the number of concatenated steps increases. **A-B**: Scheme of the computational graph for the contribution of $\frac{dE}{d\mathbf{x}(t)}$ for the case where the error function is 'continuous' across time **A**, and the case where the error function is sparse across time **B**. The blue arrows represent the factors considered, while the red arrows correspond to the factors that are neglected in the approximation. Each mathematical term adjacent to an arrow is a multiplicative factor in the contribution of a path of dependencies in the computation of $\frac{dE}{d\mathbf{x}(t)}$. **C-D**: Comparison of a running average of the derivative estimated through the online training algorithm used (red and blue lines) and BPTT (dots and triangles). The approximation is less precise when the number of hidden states used for the read-out increases, as it is evident from the greater distance between the blue trend and the dots in panel **D**.

Given the gradients with respect of the parameters of the network $\frac{dE}{d\alpha^{(k)}}$ and $\frac{dE}{dW_{ij}}$ (**W** are the output weights here) in our simulations, we used the Adam optimisation algorithm, described below for completeness for a general parameter α (that could be one of the leakage terms or W_{ij}).

$$m_t \leftarrow (1-\beta_1)m_{t-1} + \beta_1 \frac{dE}{d\alpha} \tag{49}$$

$$v_t \leftarrow (1-\beta_2)v_{t-1} + \beta_2 (\frac{dE}{d\alpha})^2 \tag{50}$$

$$\tilde{m}_t \leftarrow m_t/(1-\beta_1^t) \tag{51}$$

$$\tilde{v}_t \leftarrow v_t/(1-\beta_2^t) \tag{52}$$

$$\alpha_t = \alpha_{t-1} - \eta_\alpha(\tilde{m}_t/(\sqrt{\tilde{v}_t}+\epsilon)) \tag{53}$$

where t is the index corresponding to the number of changes made and $m_0 = 0$, $v_0 = 0$.

5.2 Timescales, oscillations and eigenvalues

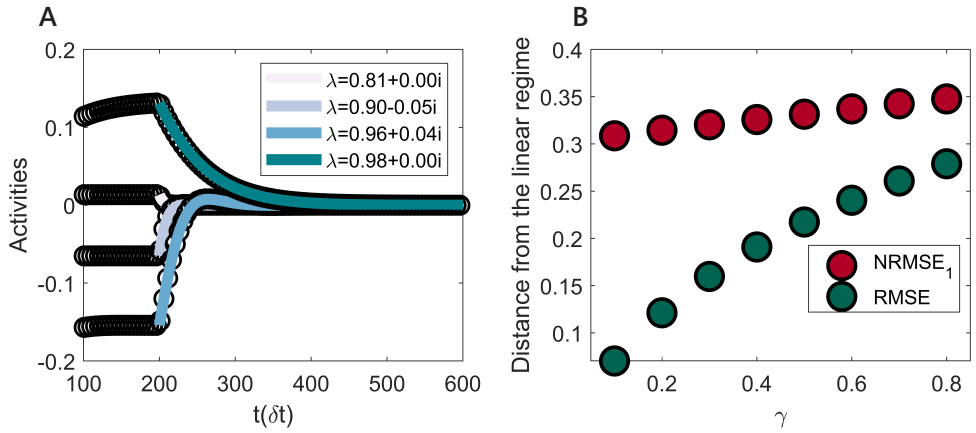

Figure 8: The analysis of the eigenvalues in the linear regime can offer insights in the understanding of the dynamical behaviour of the network. **A**: Experimental (black dots) and theoretical (coloured lines) response computed though Eq. (56). The timescales estimated are reflected in the exponentially decaying trends shown, while the oscillations are consequent to the imaginary parts of the eigenvalues. **B**: RMSE (blue) and NRMSE (red) between the activities of the nodes and the dynamic estimated in the linear regime as γ, the input strength, varies.

We stimulated the reservoir with a square wave of duration $200\delta t$ (the time frame of the considered simulation) and analysed the system activity after the impulse to study its relaxation dynamics. Thus, we exploited the fact that, given a system described by $\frac{d\mathbf{x}}{dt} = \mathbf{M}\mathbf{x}(t)$ and where $\mathbf{V}$ are the left eigenvectors of $\mathbf{M}$, i.e

$$\mathbf{V}^T \frac{d\mathbf{x}}{dt} = \mathbf{V}^T \mathbf{M}\mathbf{x}(t) = \mathbf{\Lambda}\mathbf{x}(t), \tag{54}$$

Thus the dynamics of the eigenvectors will be given by

$$\mathbf{V}^T \mathbf{x}(t) = e^{\mathbf{\Lambda} t} \mathbf{V}^T \mathbf{x}(0), \tag{55}$$

where $\mathbf{\Lambda}$ is the diagonal matrix composed by the eigenvalues of the matrix $\mathbf{M}$. Of course, in the case considered $\mathbf{M} = (1-\alpha)\mathbb{I} + \alpha\mathbf{W}$ and $\mathrm{Re}(\lambda) = 1 - \alpha + \alpha\lambda_{\mathbf{W}}$, $\mathrm{Im}(\lambda) = \alpha\lambda_{\mathbf{W}}$. Thus, considering a column $\mathbf{v}$ of $\mathbf{V}$ and the corresponding eigenvalue λ

$$\mathbf{v}^T \mathbf{x}(t) = e^{\mathrm{Re}(\lambda)t} \Big[\mathrm{Re}\big(\mathbf{v}^T \mathbf{x}(0)\big) \cos\big(\mathrm{Im}(\lambda)t\big) - \mathrm{Im}\big(\mathbf{v}^T \mathbf{x}(0)\big) \sin\big(\mathrm{Im}(\lambda)t\big) \Big], \tag{56}$$

can be used to compare the true dynamic $\mathbf{V}^T\mathbf{x}(t)$ with the linearised one. Fig. 8 shows the result of this procedure for each dimension of $\mathbf{V}^T\mathbf{x}(t)$. Panel A reports example activities and their corresponding theoretical trend for the case of small input values ($\gamma = 0.05$, see 2), case in which the system can be well approximated through a linear behaviour. Panel B shows the RMSE and NRMSE_1 between the experimental activities and the theoretical one as γ increases. In this case, with $y_i(t) = \mathbf{v}^T\mathbf{x}(t)$ experimentally observed, while $\tilde{y}_i(t)$ estimated through the right side of Eq. 56

$$\mathrm{NRMSE}_1 = \sum_{\mathrm{i}} \frac{1}{\mathrm{N}|\max(\mathrm{y_i}) - \min(\mathrm{y_i})|} \sum_{\mathrm{t}} \sqrt{\frac{\big(\tilde{\mathrm{y}}_\mathrm{i}(\mathrm{t}) - \mathrm{y}_\mathrm{i}(\mathrm{t})\big)^2}{\mathrm{T}-1}} \tag{57}$$

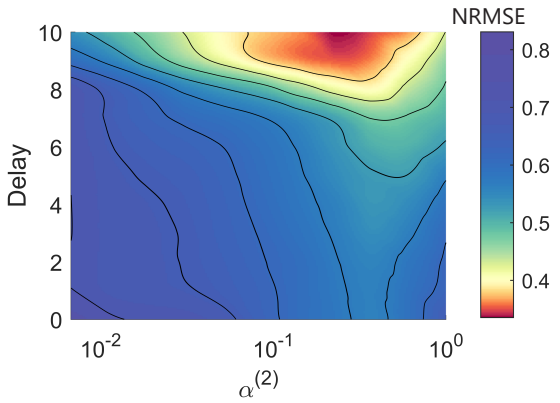

Figure 9: Performance of an ESN when the signal is expanded by addition of its previous temporal steps in the NARMA10 task. The lowest error corresponds to a leakage term $\alpha^{(2)}$ that is in agreement with the optimal value of $\alpha^{(2)}$ of the connected ESN structure reported Section 3.1 (Main Text).

5.3 Delayed Signal to ESN

We computationally validate the equation 58 (below), derived in Section 2.1, on the NARMA10 task. The NARMA10 task is described in full in Section 3.1 (Main Text).

$$\dot{\mathbf{y}}^{(2)} = \mathbf{\Lambda}^{(2)}\,\mathbf{y}^{(2)} - \tilde{\mathbf{W}}^{(12)}\,\mathbf{T}^{(1)}\left[\tilde{\mathbf{W}}^{(1)}_{\text{in}}\mathbf{s}(t) + \sum_{n=1}^{\infty}(\mathbf{T}^{(1)})^n\tilde{\mathbf{W}}^{(1)}_{\text{in}}\frac{d^{(n)}\mathbf{s}(t)}{dt^{(n)}}\right]. \tag{58}$$

In order to approximate the scaling of the coefficients of the derivatives in Eq. 58, we incorporate a delay into the input signal such that the activity of the first reservoir is replaced by

$$x_i^{(1)}(t) = \sum_{j=0}^{\text{Delay}} \xi_{ij}\,0.8^j\,s(t-j) \tag{59}$$

where ξ_{ij} are independent Gaussian variables of variance σ_ξ^2 chosen such that $\text{Var}[x_i^{(1)}] = 1$ for every i and every value of Delay. In practice, we adopted the following approximation;

$$x_i(t) \cong \mathbf{T}^{(1)}\left[\tilde{\mathbf{W}}^{(1)}_{\text{in}}\mathbf{s}(t) + \sum_{n=1}^{\infty}(\mathbf{T}^{(1)})^n\tilde{\mathbf{W}}^{(1)}_{\text{in}}\frac{d^{(n)}\mathbf{s}(t)}{dt^{(n)}}\right]. \tag{60}$$

The stochastic elements ξ_{ij} emulate the random mixing matrix that, in Eq. 58, projects

the expanded input onto the second reservoir network.

We compare the result obtained for the hierarchical network, reported in Fig. 3G, with the one illustrated in Fig. 9, where the first network has been replaced by Eq. 58, for different delays (equivalent to different orders of retained derivatives). Figure 9 shows that as the delay increases, thus higher derivatives are included, the performance appears to converge to an optimal value of $\alpha^{(2)}$ very close to the one in Fig. 3G. We also notice that the analysis illustrated earlier suggests that optimal performances are obtained for small $\alpha^{(1)}$. The agreement of results confirms the validity of the approximation used in deriving Eq. 58.

References

[1] J. Ludik, W. Prins, K. Meert, and T. Catfolis. A comparative study of fully and partially recurrent networks. In Proceedings of International Conference on Neural Networks (ICNN'97), volume 1, pages 292–297 vol.1, 1997.

[2] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[3] P. J. Werbos. Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10):1550–1560, 1990.

[4] Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. Nature Communications, 11(1):3625, Jul 2020.

[5] D Marr. A theory of cerebellar cortex. The Journal of physiology, 202(2):437–470, June 1969.

[6] Sarah M Farris. Are mushroom bodies cerebellum-like structures? Arthropod Structure and Development, 40(4):368–379, July 2011.

[7] Gilles Laurent. Olfactory network dynamics and the coding of multidimensional signals. Nature Reviews Neuroscience, 3(11):884–895, November 2002.

[8] Richard Warren and Nathaniel B Sawtell. A comparative approach to cerebellar function: insights from electrosensory systems. Current Opinion in Neurobiology, 41:31–37, December 2016.

[9] Shin-Ya Takemura, Yoshinori Aso, Toshihide Hige, Allan Wong, Zhiyuan Lu, C Shan Xu, Patricia K Rivlin, Harald Hess, Ting Zhao, Toufiq Parag, Stuart Berg, Gary Huang, William Katz, Donald J Olbris, Stephen Plaza, Lowell Umayam, Roxanne Aniceto, Lei-Ann Chang, Shirley Lauchie, Omotara Ogundeyi, Christopher Ordish, Aya Shinomiya, Christopher Sigmund, Satoko Takemura, Julie Tran, Glenn C Turner, Gerald M Rubin, and Louis K Scheffer. A connectome of a learning and memory center in the adult Drosophila brain. eLife, 6:5643, July 2017.

[10] Zhihao Zheng, J Scott Lauritzen, Eric Perlman, Camenzind G Robinson, Matthew Nichols, Daniel Milkie, Omar Torrens, John Price, Corey B Fisher, Nadiya Sharifi, Steven A Calle-Schuler, Lucia Kmecova, Iqbal J Ali, Bill Karsh, Eric T Trautman, John A Bogovic, Philipp Hanslovsky, Gregory S X E Jefferis, Michael Kazhdan, Khaled Khairy, Stephan Saalfeld, Richard D Fetter, and Davi D Bock. A Complete Electron Microscopy Volume of the Brain of Adult Drosophila melanogaster. Cell, 174(3):730–743.e22, July 2018.

[11] Q Liu, X Yang, J Tian, Z Gao, M Wang, Y Li, and A Guo. Gap junction networks in mushroom bodies participate in visual learning and memory in Drosophila. eLife, 2016.

[12] Yuhua Shang, Adam Claridge-Chang, Lucas Sjulson, Marc Pypaert, and Gero Miesenböck. Excitatory local circuits and their implications for olfactory processing in the fly antennal lobe. Cell, 128(3):601–612, February 2007.

[13] Shawn R Olsen and Rachel I Wilson. Lateral presynaptic inhibition mediates gain control in an olfactory circuit. Nature, 452(7190):956–960, March 2008.

[14] Yaara Yeshurun, Mai Nguyen, and Uri Hasson. Amplification of local changes along the timescale processing hierarchy. Proceedings of the National Academy of Sciences, 114(35):9475–9480, 2017.

[15] Herbert Jaeger. The âecho stateâ approach to analysing and training recurrent neural networks-with an erratum note. Bonn, Germany: German National Research Center for Information Technology GMD Technical Report, 148(34):13, 2001.

[16] Zhidong Deng and Yi Zhang. Collective behavior of a small-world recurrent neural system with scale-free distribution. IEEE Transactions on Neural Networks, 18(5):1364–1375, 2007.

[17] Ali Rodan and Peter Tino. Minimum complexity echo state network. IEEE transactions on neural networks, 22(1):131–144, 2010.

[18] Davide Bacciu and Andrea Bongiorno. Concentric esn: assessing the effect of modularity in cycle reservoirs. In 2018 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE, 2018.

[19] Igor Farkaš, Radomír Bosák, and Peter Gergel'. Computational analysis of memory capacity in echo state networks. Neural Networks, 83:109–120, 2016.

[20] Sarah Marzen. Difference between memory and prediction in linear recurrent networks. Phys. Rev. E, 96:032308, Sep 2017.

[21] L. Livi, F. M. Bianchi, and C. Alippi. Determination of the edge of criticality in echo state networks through fisher information maximization. IEEE Transactions on Neural Networks and Learning Systems, 29(3):706–717, 2018.

[22] Yanbo Xue, Le Yang, and Simon Haykin. Decoupled echo state networks with lateral inhibition. Neural Networks, 20(3):365–376, 2007.

[23] Herbert Jaeger. Discovering multiscale dynamical features with hierarchical echo state networks. Technical report, Jacobs University Bremen, 2007.

[24] Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli. Deep echo state networks for diagnosis of parkinson's disease. arXiv preprint arXiv:1802.06708, 2018.

[25] Zeeshan Khawar Malik, Amir Hussain, and Qingming Jonathan Wu. Multilayered echo state machine: a novel architecture and algorithm. IEEE Transactions on cybernetics, 47(4):946–959, 2016.

[26] Claudio Gallicchio and Alessio Micheli. Echo state property of deep reservoir computing networks. Cognitive Computation, 9(3):337–350, 2017.

[27] Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli. Design of deep echo state networks. Neural Networks, 108:33–47, 2018.

[28] Qianli Ma, Lifeng Shen, and Garrison W. Cottrell. Deepr-esn: A deep projection-encoding echo-state network. Information Sciences, 511:152 – 171, 2020.

[29] Nathaniel Rodriguez, Eduardo Izquierdo, and Yong-Yeol Ahn. Optimal modularity and memory capacity of neural reservoirs. Network Neuroscience, 3(2):551–566, 2019.

[30] István Szita, Viktor Gyenes, and András Lőrincz. Reinforcement learning with echo state networks. In International Conference on Artificial Neural Networks, pages 830–839. Springer, 2006.

[31] Michiel Hermans and Benjamin Schrauwen. Recurrent kernel machines: Computing with infinite echo state networks. Neural Computation, 24(1):104–133, 2012.

[32] Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. Neural networks, 20(3):335–352, 2007.

[33] Vyacheslav L Girko. Circular law. Theory of Probability & Its Applications, 29(4):694–706, 1985.

[34] Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli. Deep reservoir computing: A critical experimental analysis. Neurocomputing, 268:87 – 99, 2017. Advances in artificial neural networks, machine learning and computational intelligence.

[35] Xiaochuan Sun, Tao Li, Qun Li, Yue Huang, and Yingqi Li. Deep belief echo-state network and its application to time series prediction. Knowledge-Based Systems, 130:17–29, 2017.

[36] Luca Manneschi and Eleni Vasilaki. An alternative to backpropagation through time. Nature Machine Intelligence, 2(3):155–156, 2020.

[37] A. F. Atiya and A. G. Parlos. New results on recurrent network training: unifying the algorithms and accelerating convergence. IEEE Transactions on Neural Networks, 11(3):697–709, 2000.

[38] Alireza Goudarzi, Peter Banda, Matthew R Lakin, Christof Teuscher, and Darko Stefanovic. A comparative study of reservoir computing for temporal signal processing. arXiv preprint arXiv:1401.2224, 2014.

[39] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. Computer Science Review, 3(3):127–149, 2009.

[40] Nils Schaetti, Michel Salomon, and Raphaël Couturier. Echo state networks-based reservoir computing for mnist handwritten digits recognition. In 2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES), pages 484–491. IEEE, 2016.

[41] Luca Manneschi, Andrew C Lin, and Eleni Vasilaki. Sparce: Sparse reservoir computing. arXiv preprint arXiv:1912.08124, 2019.

[42] Sarath Chandar, Chinnadhurai Sankar, Eugene Vorontsov, Samira Ebrahimi Kahou, and Yoshua Bengio. Towards non-saturating recurrent units for modelling long-term dependencies. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 3280–3287, 2019.

2.2 Paper II

The context of this second paper is again reservoir computing. The aim is to develop a learning algorithm that can improve the ability of ESNs, more commonly adopted for regression problems, to deal with classification tasks. In order to achieve this, we added thresholds on the read-out level of the reservoir to introduce specialised activities. Thanks to the proposed technique, representations of data belonging to different classes, or tasks, do not completely overlap, thus facilitating learning of associations, or the ability of the system to learn multiple tasks sequentially. The idea to introduce a thresholding mechanism that can control the sparsity level of a representation is inspired by the sparse coding observed in biological neural networks, where neurons are specialised and react to specific stimuli. In particular, an example of biological network that exhibits sparse coding and that acts similarly to a reservoir is the mushroom body (MB) of the *Drosophila*. In the MB, dimensionality expansion occurs from the input, encoded as neuronal activity of projection neurons, to Kenyon cells (KCs), which can be thought as a biological analogous of the nodes in a reservoir. Adaptation is driven by dopaminergic signals and occurs on the synapses among KCs and output neurons. There are two interesting features, experimentally observed in the MB and across a multitude of biological networks, that inspired the following work. First, the high intrinsic neuronal thresholds and the presence of inhibitory feedback loops can control the level of specialisation of a population of neurons [4]. Second, neurons can be recurrently connected and affect each other through sub-threshold activities [16]. Similarly to this biological findings, we formulated a model (SpaRCe) where thresholds are introduced on the read-out level without affecting the reservoir dynamics. Thresholds will be optimised to discover the optimal sparsity level for a given task. Similarly to the learning process of the leakage terms in Paper I, the hierarchical dependence of the output weights on the thresholds' has consequences on the optimal learning regime of the model. Indeed, we will find that output connections need to adapt at a much faster timescale in comparison to thresholds in order to exploit the proposed algorithm best. Consequently, the concept of multiple timescales can be found in the different rates of adaptation of the trainable parameters. In the future, the model proposed could be also applied to physical reservoir computing devices.

The paper has been published under IEEE: Transcations on Neural Networks and Learning Systems (doi: 10.1109/TNNLS.2021.3102378).

My contributions to the work were: model development, coding, testing, writing of the paper.

SpaRCe: Improved Learning of Reservoir Computing Systems through Sparse Representations

Luca Manneschi [1], Andrew C. Lin [2], Eleni Vasilaki [1,+]

[1] Department of Computer Science, The University of Sheffield, Sheffield, United Kingdom,
[2] Department of Biomedical Science, The University of Sheffield, Sheffield, United Kingdom,

Abstract

"Sparse" neural networks, in which relatively few neurons or connections are active, are common in both machine learning and neuroscience. Whereas in machine learning, "sparsity" is related to a penalty term that leads to some connecting weights becoming small or zero, in biological brains, sparsity is often created when high spiking thresholds prevent neuronal activity. Here we introduce sparsity into a reservoir computing network via neuron-specific learnable thresholds of activity, allowing neurons with low thresholds to contribute to decision-making but suppressing information from neurons with high thresholds. This approach, which we term "SpaRCe", optimises the sparsity level of the reservoir without affecting the reservoir dynamics. The read-out weights and the thresholds are learned by an on-line gradient rule that minimises an error function on the outputs of the network. Threshold learning occurs by the balance of two opposing forces: reducing inter-neuronal correlations in the reservoir by deactivating redundant neurons, while increasing the activity of neurons participating in correct decisions. We test SpaRCe on classification problems and find that threshold learning improves performance compared to standard reservoir computing. SpaRCe alleviates the problem of catastrophic forgetting, a problem most evident in standard echo state networks and recurrent neural networks in general, due to increasing the number of task-specialised neurons that are included in the network decisions.

1 Introduction

The performance of artificial neural networks is often improved by adopting "sparse" representations, in which relatively few neurons or connections are active. Previous research has studied the role of sparse connectivity, in terms of memory, in Hopfield networks, demonstrating how sparse connectivity increases storage capacity [1] [2] [3] [4]. Memory retrieval and associative learning have been studied in the context of neural network attractors, and the work in [5] has provided an abstract mathematical analysis of retrieval capacity. From the machine learning perspective, adopting sparse connectivity can lead to more interpretable models [6], a reduced computational cost [7], and can help solve

overfitting problems [8]. Sparsity in machine learning is typically introduced to artificial networks through regularisation, in which a penalty term leads to the reduction of the connection weights. In this regard, the work in [7] demonstrated how structured sparsity can improve computational speed and accuracy in a convolutional neural network. Rasmussen et al. [9] showed how the choice of regularization parameters of the model can impact the interpretability and the reproducibility of a classifier of neuroimaging data, and showed the existence of a trade-off between pure classification accuracy and reproducibility.

Sparsity is also a well-known concept in neuroscience: biological neurons are highly selective in systems ranging from mammalian sensory cortex [10] to the insect mushroom body [11] [12]. However, unlike in typical machine learning approaches, biological sparsity is introduced not only by reducing connection weights between neurons, but also by the fact that neurons have spiking thresholds: they only fire when their summed inputs exceed a certain threshold. High spiking thresholds relative to the size of synaptic inputs can often contribute to high selectivity of neurons, as with Kenyon cells (KCs), the principal neurons of the insect mushroom body, which fire sparsely in response to odor stimuli [13] [14] [15] [16]. In the fruit fly *Drosophila*, this sparse odor coding enhances learned discrimination of similar odors [12]. Moreover, spiking thresholds vary across neurons [13] and over time for the same neuron [17] [18], and spiking thresholds for different neurons are adapted to neurons' particular input statistics [17] and past activity [19].

Previous efforts that introduce sparsity inspired by neuroscience findings are based on the BCM learning rule [20] and on intrinsic plasticity [21]. Both these concepts introduce a "soft" regulation of network activity and have been applied to reservoir computing [22] [23] [24]. Here we are taking a different approach to regulate sparsity in reservoir computing: we introduce hard, adaptable activity thresholds to create SpaRCe.

Reservoir computing takes inspiration from the complex non-linear behavior of recurrent neural networks and their abilities to process temporal information. Such a computing paradigm exploits the inherent complexity of a dynamical system, which is called the reservoir and that can be virtually or physically defined, to represent a stream of data

into a high dimensional space useful for learning. In the classical learning framework of such systems, training occurs on the read-out connections only, which connect the activities of the non-linear reservoir components to the output neurons. The advantages of the reservoir computing framework are: first, the system can be physically defined as in [25, 26, 27, 28, 29], an aspect that can dramatically reduce the computational and energetic cost [30]; second, the response of the reservoir can exhibit memory over a range of timescales without training the internal dynamic of the system. In the latter aspect, the reservoir computing framework contrasts with that of standard recurrent neural networks, whose connectivity is trained through the computationally expensive backpropagation through time (BPTT) algorithm.

In this work, we model the reservoir as a recurrent network of leaky integrators [31] known as an echo state network (ESN). The connectivity between the nodes is represented through a random sparse fixed adjacency matrix, whose spectrum of eigenvalues allows the system to exhibit a wide range of timescales and to efficiently represent temporal information. Previous works have explored alternatives to the typical Erdos-Renyi connectivity of the reservoir, imposing a regular structure as a delay line [32], where the nodes are connected unidirectionally composing a circular graph, or defining the graph connectivity through more complex topologies [33] as scale-free or small world. Furthermore, complex structures composed by multiple, interconnected ESNs have been recently studied, and the works [34] [35] demonstrate how the use of different hyperparameters for different ESNs can expand the range of timescales that is accessible by the system and consequently increase the quality of the reservoir representation for temporal tasks with complex temporal dynamic. In this regard, the work [35] gives insights into the understanding of hierarchical systems from the point of view of accessible timescales, offering a theoretical and a practical analysis on how to tune the hyperparameters of the system to a considered task.

While ESNs are most traditionally adopted by the machine learning community for time-series prediction and forecasting [31], more recent works have studied their behav-

ior for time-series classification [36]. In particular, [36] studies and compares different methods to exploit the representation of the reservoir as input to more complex machine learning algorithms as multilayer perceptrons or SVM. In this work, we formulate a different approach to improve the classification ability of reservoir systems in general while maintaining the low cost of computation that is intrinsic to the concept of reservoir computing. Analogously to the concept of firing thresholds, SpaRCe exploits learnable thresholds to optimize the level of sparsity inside the network, without disturbing the underlining network dynamics. Both the learnable thresholds and the read-out weights (but not the recurrent connections within the reservoir) are optimised by minimising an error function that does not include any normalization term. We analysed the learning rule derived from this error minimisation and found that learning occurs by two antagonist factors: the first raises the thresholds proportionally to the correlated activity of the nodes (thus silencing nodes that are correlated and therefore redundant), while the second lowers the thresholds of nodes that contribute to the correct classification (Fig. 2). The novelty of our work lies in the fact that a sparsity level is reached due to the presence of *on-line* learnable firing thresholds, rather than to penalty terms [37] [6] [38], as well as in the detailed analysis on how such thresholds enable ESNs to perform competitively in tasks where their performance was lacking.

2 Methods

2.1 Standard Echo State Network

An echo state network (ESN) is composed of randomly connected neurons. The activity of such a system is commonly defined through the following equation [31]:

$$\mathbf{V}(t+\delta t) = (1-\alpha)\mathbf{V}(t) + \alpha f\left[\gamma \mathbf{W}_{in}\mathbf{s}(t) + \rho \mathbf{W}\mathbf{V}(t)\right] \tag{1}$$

where $\alpha = \frac{\delta t}{\tau}$ is the leakage term, τ defines the temporal scale of integration of the nodes, $\mathbf{V}(t)$ is the activity vector of the integrators and $\mathbf{s}(t)$ is the input signal. $\mathbf{W}$ is the fixed sparse random matrix that describes the recurrency of the reservoir. The matrix $\mathbf{W}$ is random and sparse, corresponding to an Erdos-Renyi graph where the probability of a connection is p_{ER}. The eigenvalues of $\mathbf{W}$ are rescaled to be confined inside the unit circle of the imaginary plane, necessary condition for the echo state property. Considering the parameterisation of the network in Eq. 1, this condition is equivalent to the one proposed in [34] and guarantees that the eigenvalues of the associated, linearised dynamic system are also confined in the unit circle of the imaginary plane. The hyperparameter ρ (between 0 and 1) controls further the radius of the spectrum of the system's eigenvalues. Finally, γ is a gain factor of the input signal. The specific form of the input matrix $\mathbf{W}_{in}$ and the activation function f is task-dependent and will be specified in sections 3.1 and 3.2.

It is possible to control *a priori* the range of timescales that the reservoir exhibits by appropriately choosing α and ρ as described in the methodology reported in Appendix A.1 and our earlier work [35]. In the standard ESN learning protocol, learning occurs exclusively on the read-out defined from a vector $\tilde{\mathbf{V}}$ that represents an ensemble of the reservoir activities $\mathbf{V}$. The output of the system is then computed through

$$\mathbf{y} = \mathbf{W}^o \tilde{\mathbf{V}} \tag{2}$$

$$E = \frac{1}{2}\Big[\tilde{\mathbf{y}} - \mathbf{y}\Big]^2 \tag{3}$$

where $\tilde{\mathbf{y}}$ denotes the desired output values, and the output connectivity $\mathbf{W}^o$ is optimised to minimise the cost function E through ridge regression [31] or through gradient descent methods [39]. We note that, depending on the task and on the learning framework, the $\tilde{\mathbf{V}}$ vector can be defined from different ensembles of activities of the reservoir across time. The idea to exploit the dynamic of the ESNs, instead of using its activities at one step only, is particularly useful for classification tasks. Previous works have introduced different techniques to define a $\tilde{\mathbf{V}}$ vector from the ESN's dynamic: adopting PCA on

the multidimensional response of the reservoir across time [36], introducing a learnable temporal kernel to define the read-out [40], or concatenating the past activities of the reservoir to define a higher dimensional vector $\tilde{\mathbf{V}}$ [39] [41]. In this work, we will exploit the latter approach, while other techniques such as PCA, which reduces the dimensionality of the ESN's representation, are more desirable in the case of overfitting. In this regard, while the most spontaneous choice for time-series prediction would be $\tilde{\mathbf{V}} = \mathbf{V}(t)$, corresponding to the activity of the reservoir at the current time, it is also possible to expand the dimensionality of the system by including previous reservoir representations at times t_l and define a vector $\tilde{\mathbf{V}} = \mathcal{C}\Big(\{\mathbf{V}(t_l)\}_{t_l \varepsilon \mathcal{T}}\Big)$, where $\mathcal{C}$ denotes the concatenation operator and $\mathcal{T}$ the ensemble of time steps t_l considered. This latter approach has been adopted also in physical reservoir models through the use of virtual nodes [42] [43]. Of course, such dimensionality expansion leads to an artificial increase of the memory of the system. However, the concatenation of previous activities does not guarantee the understanding of the dependencies among events that are distant in time, because the memory of a past input signal could have faded away when a new stimulus $\mathbf{s}(t)$ comes. A scheme of this procedure for time-series classification can be found in Fig. 1 A.

2.2 Sparse Reservoir Computing

In contrast to previous models that define the output of the neural network through a read-out of the $\tilde{\mathbf{V}}$ vector, i.e. the activities of the reservoir considered for the learning process, we introduced another variable x_i for each dimension of $\tilde{\mathbf{V}}$, defined as follows:

$$x_i = \text{sign}\Big(\tilde{V}_i\Big)\text{ReLU}\Big\{|\tilde{V}_i| - \theta_i\Big\} \tag{4}$$

$$\theta_i = P_n\Big(|\tilde{\mathbf{V}}_i|\Big) + \tilde{\theta}_i \tag{5}$$

where ReLU stands for rectified linear unit, sign is the sign function (1 if $\tilde{V}_i > 0$, -1 if $\tilde{V}_i < 0$, 0 if $\tilde{V}_i = 0$), and θ_i is a threshold that enables x_i to be sparse. Thus, the variable

x_i is zero if the absolute value of the variable V_i is lower than the corresponding threshold. We term this variant SpaRCe for sparse reservoir computing. Of course, $\mathbf{x} = \tilde{\mathbf{V}}$ if $\theta = 0$, which is the case where the formulated learning procedure coincides with the standard ESN paradigm. Considering Eq. 5, each threshold is composed by two factors:

- $P_n(|\tilde{\mathbf{V}}_i|)$, defined as the percentile n of the distribution of activity of the i-th component of $|\tilde{\mathbf{V}}|$ across a dataset, where the same value of n is applied to all dimensions of $\tilde{\mathbf{V}}$. Given that the response $\tilde{\mathbf{V}}$ of the reservoir to a specific input remains unchanged across learning, this term can be computed over the training dataset and maintained constant throughout the simulation. Thus, the role of $P_n(|\tilde{\mathbf{V}}_i|)$ is to initialise the thresholds from an initial condition where the sparsity level of all dimensions of $\mathbf{x}$ is $n/100$. The thresholding mechanism introduced is symmetric with respect to zero, as is visible from the examples in Fig. 1D-E. While panel D highlights the portions of the distributions that are positive after normalisation, panel E shows the resulting distributions after application of Eq. 5, but before learning, that is when $\tilde{\theta}_i \approx 0$. In this formulation, the value of n needs to be considered as an additional, interpretable hyperparameter. However, we will show that all the sensible choices of n will correspond to faster convergence and better performance of the formulated algorithm in comparison to the standard ESN learning paradigm across all tasks considered. We note that, even for P_0, SpaRCe and a vanilla ESN would differ, because the threshold values for P_0 are non-zero and the activities of the reservoir would be shifted. More specifically, choosing P_0 will lead to a shift to the distributions of $|\tilde{\mathbf{V}}|$, whose minimum value will now become approximately zero. Throughout this paper, the same values of $P_n(|\tilde{\mathbf{V}}_i|)$ computed over the training dataset will be used for the validation and test dataset.

- An adaptable component $\tilde{\theta}_i$ optimised through gradient descent learning rules. This parameter adapts the sparsity level for the considered task and can rediscover the

standard learning paradigm in the case where $\tilde{\theta}_i = -P_n(|\tilde{\mathbf{V}}_i|)$, $\forall i$.

The additional complexity arising from Eq. 4 is summarized by Fig. 1 B, which depicts the difference between the read-out of a standard Echo-State network and our formulation. Eq. 4 acts as normalisation operator that directly controls sparsity in the reservoir representation and that, thanks to the learnable components $\tilde{\boldsymbol{\theta}}$, works as a feature-selection mechanism. In the latter sense, we can now demonstrate the interpretability of the updating equations on $\boldsymbol{\theta}$ derived from a gradient descent approach.

For the case of a mean squared error function, the learning rule can be separated in two terms:

$$\Delta\theta_k = \eta_\theta \Big[\Delta\theta_k^{(1)} + \Delta\theta_k^{(2)}\Big] \tag{6}$$

$$\Delta\theta_k^{(1)} = \sum_{j=1}^{N_{class}} y_j W_{jk}^o \mathrm{sign}(x_k) =$$

$$= \sum_{j=1}^{N_{class}} \sum_{l=1}^{N} \Big[W_{jl}^o x_l\Big] \Big[W_{jk}^o \mathrm{sign}(x_k)\Big] \tag{7}$$

$$\Delta\theta_k^{(2)} = -W_{\tilde{j}k}^o \mathrm{sign}(x_k) \tag{8}$$

where η_θ is the learning rate on the thresholds, $\tilde{j}$ refers to the correct output class for the sample considered, $\mathbf{W^o}$ is the output connectivity matrix. Eq. 7 and 8 are derived by assuming one-hot encoding (Appendix B.2). Considering $x_k > 0$ (analogous considerations are valid for $x_k < 0$), the factor $\Delta\theta^{(2)}$ decreases (increases) the threshold value of nodes with $W_{\tilde{j}k}^o > 0$ ($W_{\tilde{j}k}^o < 0$) that help (hinder) the network to reach the right classification. Thus, $\Delta\theta^{(2)}$ is driven by the output weight between the considered node (if it is active) and the desired class. In contrast, $\Delta\theta^{(1)}$ is a measure of correlation of weighted activities between different nodes in the reservoir and increases (decreases) the thresholds of neurons that have coherent (opposite sign) contributions and therefore reduces redundancy in the reservoir.

We examined the two factors $\Delta\theta^{(1)}$ and $\Delta\theta^{(2)}$ across learning for an example of se-

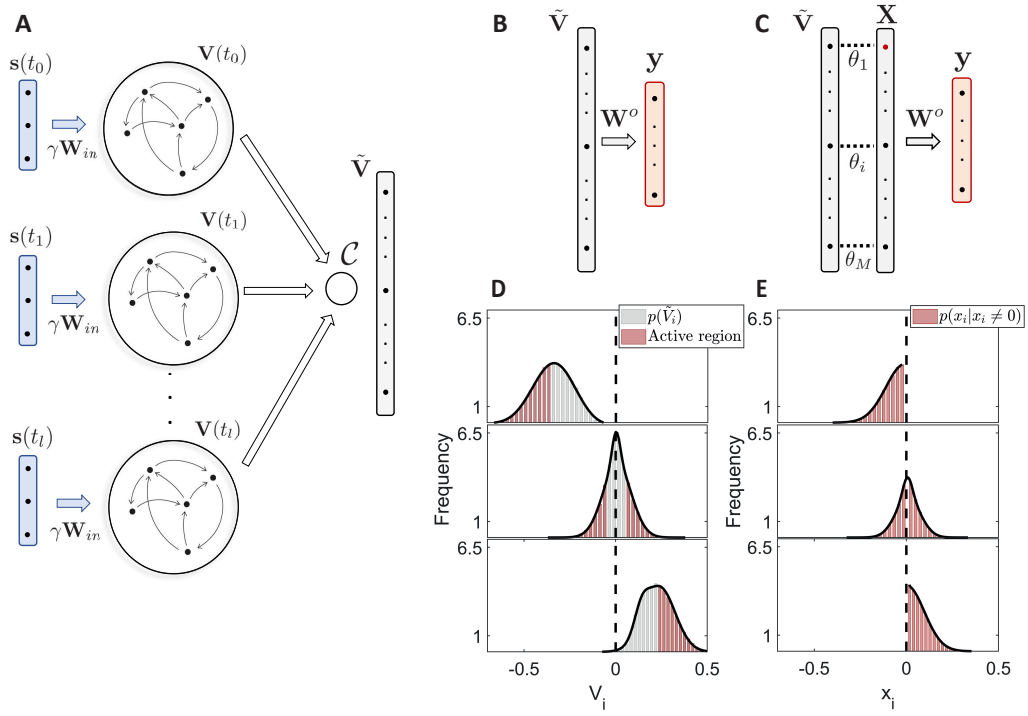

Figure 1: Comparison between the basic reservoir computing framework and SpaRCe, which exploits the concepts of adaptable thresholds to introduce sparsity in the representation of the ESN. **A**: The typical reservoir computing sampling paradigm for a time-series classification task. Time flows from top to bottom. The input signal $\mathbf{s}(t)$ is fed into the ESN through the input connectivity matrix $\gamma\mathbf{W}_{in}$. An ensemble of the activities $\mathbf{V}(t)$ across time is selected and concatenated to compose a vector $\tilde{\mathbf{V}}$, which is then used to define the read-out. Typical choices of ESN activities used to define the read-out are $\tilde{\mathbf{V}} = \mathbf{V}(T)$, where T denotes the final temporal step of the input sequence, or $\tilde{\mathbf{V}} = \mathcal{C}\Big(\{\mathbf{V}(t)\}_{\forall t}\Big)$, denoting the concatenation of all the reservoir dynamic across the temporal length of $\mathbf{s}(t)$. Of course, while the first approach relies on the ESN intrinsic memory capacity, the latter case corresponds to a dimensionality expansion that contributes artificially to the memory of the system. Both these approaches and intermediate cases, i.e. where the dynamic of the reservoir across time is sampled with low frequency to compose the read-out, will be exploited in the tasks studied. **B-C**: Comparison between the typical reservoir computing read-out (**B**) and the SpaRCe model (**C**). Considering that $\tilde{\mathbf{V}}$ is used for the read-out, we define a threshold for each dimension of $\tilde{\mathbf{V}}$. The result of this approach is the definition of different thresholds across time in the case where the representation is enriched through concatenation of the history of activities of the reservoir. In this way, the approach defined is general and can be applied regardless of the technique used to define the representation $\tilde{\mathbf{V}}$ for the read-out of the system. Furthermore, the initialisation of different thresholds across time can be helpful to the case where there is temporal drift in the dataset, leading the activity $\mathbf{x}$ of the reservoir to exhibit a more stationary behaviour across time. **B**: The ESN output $\mathbf{y} = \mathbf{W}^o\tilde{\mathbf{V}}$ is responsible for the classification process of the example sequence $\mathbf{s}(t)$. In this paradigm, learning occurs exclusively on $\mathbf{W}^o$. **C**: Scheme of the SpaRCe model. Thresholds are introduced at the level of the $\tilde{\mathbf{V}}$ vector, leaving unaffected the dynamic of the reservoir and making the approach applicable to any physical or virtual reservoir model. Each threshold value is composed by a normalisation term $\mathbf{P}_n(|\tilde{\mathbf{V}}_i|)$, defined as the n-th percentile of the activity distribution of the i-th component across the data, plus an adaptable term $\tilde{\theta}_i$ (see text for more details). **D-E**: Activity distributions of $\tilde{\mathbf{V}}$ (**D**) and $\mathbf{x}$ (**E**) before the training process, where $\tilde{\theta} \approx 0$, for three example nodes. Each distribution is fitted through two Gaussians for clarity purposes. The highlighted red region in **D** corresponds to the values for which the nodes would be active if the normalisation mechanism proposed in Eq.4 would be applied (percentile P_{50} in this case). From **E**, it is clear that Eq.4 also shifts the activity distributions acting as a normalisation mechanism.

quence classification and for different initial sparsity levels. These are plotted in Fig. 2C, with $\Delta\theta^{(2)}$ on the positive y-axis and $\Delta\theta^{(1)}$ on the negative y-axis. The two forces are almost symmetric, but their slight imbalance provides the direction to change the threshold values. Indeed, if the starting sparsity level is high, the total force is negative and the factor $\Delta\theta^{(1)}$ dominates, while if the sparsity level is low, the correlation term $\Delta\theta^{(2)}$ wins and the thresholds increase on average (compare P_{60} and P_{95} in Fig. 2D, which shows the effect of learning the thresholds in terms of average percentile (sparsity) change across the network. The reason that the magnitudes of the forces are larger for a higher starting sparsity (Fig. 2C) is that stimulus representations overlap less when the sparsity level is higher and neurons are more specialised, i.e. preferably fire for one pattern over the others. This leads to more coherent sign of the output weights of the nodes belonging to a cluster toward a specific class, which increases $\Delta\theta^{(1)}$ according to Eq. 7. A similar analysis of the learning rule for a cross entropy cost function is reported in Appendix B.2. Fig. 2A,B describes the benefits due to the application of the proposed normalisation mechanism: first, it directly controls sparsity and introduces specialised neurons (Fig. 2B); second, it shifts the activity distributions $\tilde{\mathbf{V}}$ of the network into a more stereotyped response $\mathbf{x}$, avoiding the possibility that learning could be dominated by the nodes with highest activities (Fig. 2A). Finally, it stabilises the learning process for a wide range of possible learning rates that would not be accessible without thresholds (Appendix B.2, Fig. 2). This specific formulation allows the model to use local information to learn the threshold values and optimize sparse representations. We note also how Eq. 4 does not affect the timescales of the network and consequently preserves the idea of reservoir computing as a fixed, dynamically rich representation. Furthermore, the method formulated constitutes a computationally inexpensive procedure that is easily applicable to any type of reservoir, virtually or physically defined.

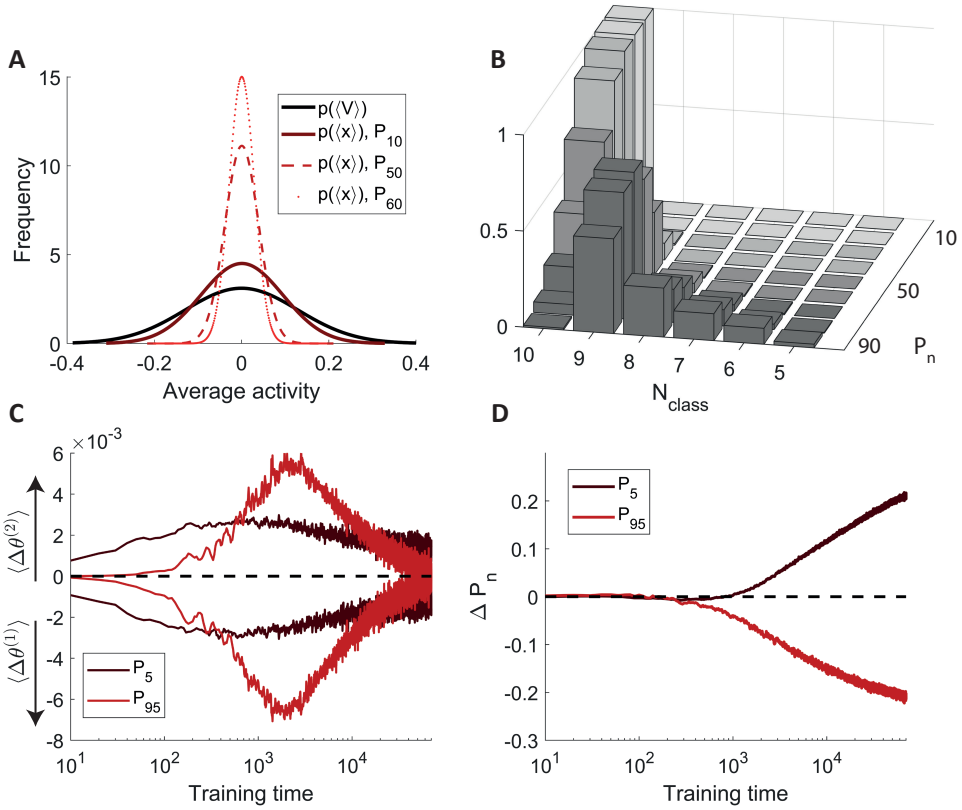

Figure 2: The SpaRCe read-out proposed leads to decreased variability in the ESN representation, to specialised responses, and to a interpretable learning rule that acts as feature selection mechanism. The learning rule for the thresholds is driven by the imbalance between two antagonist forces. **A** Distributions of the average activities of nodes across an example dataset (MNIST, section 2.3) for the standard ESN and for SpaRCe as the percentile of the normalisation mechanism changes at the beginning of training (when $\tilde{\boldsymbol{\theta}} = 0$). SpaRCe decreases the variability of the activity distributions, acting as a normalisation mechanism. **B**: Frequency of active nodes for different starting sparsity levels and different number of classes for a classification task with ten classes (MNIST, section 2.3) before training ($\tilde{\boldsymbol{\theta}} = 0$). We note that N_{class} refers to the total number of classes to which a node responds. In particular, nodes that are active for $N_{class} = 10$ are responding to all ten classes, while nodes that react for $N_{class} = 6$ are responding to six classes only. As sparsity (P_n) increases, nodes respond to a smaller number of classes becoming more specialised. **C**: Analysis of the two forces $\Delta\theta^{(1)}$ and $\Delta\theta^{(2)}$ involved in the learning rule for the thresholds. The positive y-axis shows a running average of $\Delta\theta^{(2)}$, while the negative y-axis shows a running average of $\Delta\theta^{(1)}$. $\langle\rangle$ indicates averaging across all neurons. $\Delta\theta^{(2)}$ increases (decreases) the threshold values for nodes that are equally (differently) contributing to the classification process. $\Delta\theta^{(1)}$ decreases (increases) the threshold values thanks to the positive (negative) contribution of the output weights connected to the correct output. Colours correspond to initial conditions (P_{60}, P_{95}). **D**: Average cumulative change of a threshold in terms of percentile change due to adaptation of $\tilde{\theta}$. If the starting level of sparsity is suboptimal and low (high) the average percentile change is positive (negative).

2.3 Benchmarks

We tested the proposed model on the following benchmarks, comparing its performance to the standard read-out of an ESN:

- A biologically inspired task to test the storage capacity of the system in the memorisation of associations between sequences and desired output values. In this case, the model is tested on the same associations used for training, but with different realisations of multiplicative white noise inserted on the sequences. The dependence of SpaRCe on the starting sparsity level is analysed along comparisons of the performance achieved with other methods for reading out from the ESN representation, including deep feedforward networks.

- Three variants of classification tasks based on the MNIST dataset, where data are presented to the system as temporal varying sequences. In the first task, we studied the performance of the model when each image, composed by 28×28 pixels, is presented column by column as a 28-dimensional sequence of 28 time steps (MNIST [41]). In the second, the input is processed in the same way, but a random permutation of the pixels is applied to all data to make the task more challenging and to randomise the structures of the images (pMNIST). In the third, we applied a random permutation of the data as in the second paradigm, but each image is presented pixel by pixel as a one dimensional sequence of 784 temporal steps (psMNIST [44]). When the initial conditions of the model are studied, the performance are computed and shown on the test set as P_n varies for comparison, but to select the best P_n value and report the highest performance we used a validation dataset. Of course, we selected the hyperparameters corresponding to the highest accuracy on the validation set and then computed the performance on the test dataset.

- Two tasks that involve sequential learning. In the first case, we applied ten different permutations to the MNIST data and trained the model processing the tasks one after the other [45] [46]. In the second, the network is trained on the MNIST data belonging to different classes sequentially [45] [46]. In both cases, the network can access a specific dataset (corresponding to a permutation or a class) only once during training. As before, a validation dataset is adopted when selecting the best hyperparameters (in particular P_n) of the proposed model.

3 Results

3.1 Threshold learning increases storage capacity

We evaluated the performance of a standard ESN and SpaRCe in classifying an ensemble of sequences $\{\mathbf{S}_i\}_{i=1,..,N_{seq}}$ of three successive stimuli, where the dimensionality of the signal is $N_{In} = 24$. Each stimulus of a sequence is derived from the simulated response of $N_{In} = 24$ projection neurons (PNs, in the fly olfactory system) to 110 different odors [47] [48]. This simulated activity, which we call $\mathbf{s}^{HO}$ (HO for Hallem-Olsen), has previously been used in computational analyses of fly olfaction [49] [50] [51].

Sequences are generated to test the storage capacity of the system in memorising associations between input signals and desired output values. The procedure for building different sequences from single stimuli is described the Appendix (section C.1, Fig. 1), but essentially, it guarantees that each sequence has to be classified as independently as possible from other sequences, and that there are no correlations of elements among different inputs that can inform the classification process. Thus, the system can only memorise the associations among a specific succession of elements and the corresponding output value.

There are three stimuli in a sequence, each of them is presented for a time interval

$\Delta t = 0.1s$ in order to allow the network to integrate the information. The total duration of an input sequence is $T = 0.3s$. Given a sequence $\mathbf{s}_i(t)$, we inserted multiplicative white noise to each separate dimension to make the task more complex. Thus, the i-th dimension of the final sequence $\mathbf{S}_i(t)$ to be classified is $S_i(t) = s_i(t) + \sigma_s \xi_i(t) s_i(t)$, where $\xi_i(t)$ is a Gaussian distributed random variable with zero mean and unitary variance and σ_s is the noise variance.

For this specific task, the activation function f of Eq. 1 is a rectified linear unit and the connections of the input adjacency matrix $\mathbf{W}_{in}$ follow a lognormal distribution. This particular form of $\mathbf{W}_{in}$ is inspired by the biological results in [13] [52] [53]. In this case, $\tilde{\mathbf{V}} = \mathbf{V}(T)$, meaning that only the activities at the last temporal step of a sequence are adopted for the read-out, and the memory of past events is left to the internal dynamic of the ESN. A schematic representation of the task can be found in the Appendix.

We first investigated how the model depends on initial sparsity, and found an optimal sparsity level for the task. We initialized the network with different initial sparsity percentiles (i.e., thresholds at different percentiles of the $\mathbf{V}$ distribution, $n = [10, 20, 30, 40, 50, , 60, 70, 80, 90]$), and tracked the mean square error over the learning process (Fig. 3A). Errors decreased as learning proceeded, but at each time point, the lowest error occurred for an initial sparsity of about 50%. Furthermore, models initialized with sparsity values other than 50% (an explanation of why 50% can be found in the Appendix) converged toward 50% as training proceeded, as shown by the black dashed lines connecting dots of training instances from the top to the bottom of the graph (Fig. 3A). This shows how the learning rule pushes the percentage of active nodes toward the optimal sparsity level.

Notably, the error is smallest when specialisation is highest (for a definition of specialisation, see Appendix A.2, Eq. A5; conceptually, specialisation represents to what extent a node is active for stimuli of one class, not stimuli of other classes), as shown in Fig. 3B, which reports the error as a function of sparsity and specialisation. For all training instances analysed, the lowest error corresponds to the highest specialisation value. Thus, specialisation provides a systematic way to choose the starting condition of the network.

Indeed, it is possible to select the thresholds using the percentile value that yields the highest specialisation measure. However, there is no need to excessively fine-tune the initialization, since the learning rule will optimize the threshold values anyway. We note also that this simulation is performed through a simple gradient descent algorithm, and that the model's dependence on initial conditions can be ameliorated by using more complex optimizers, as will be shown in section 3.2.

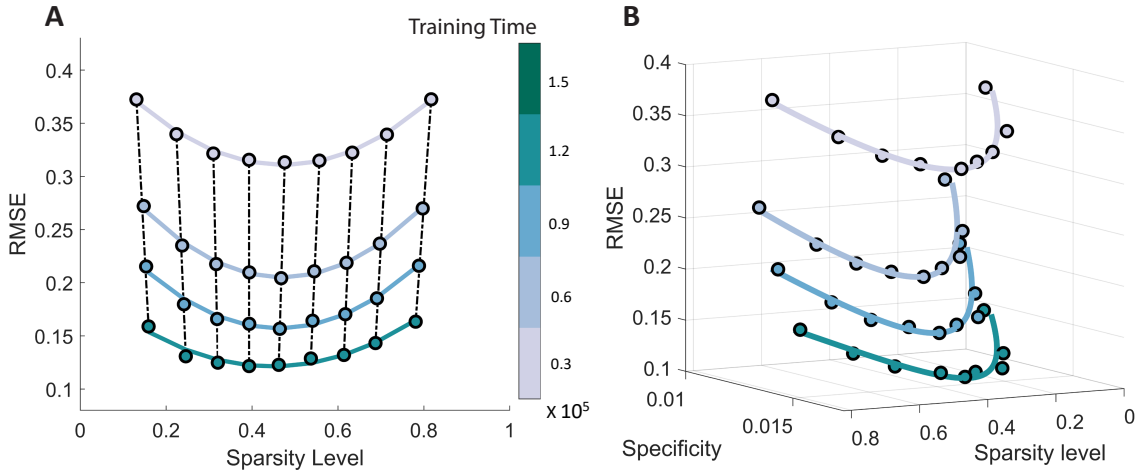

Figure 3: The learning process modulates the network's sparsity level toward an optimal percentage of active nodes. **A**: Performance as a function of sparsity for different training instances of the model (a color represents a specific training time, which increases from top to bottom). For each instance the results are fitted with a second degree polynomial ($\chi^2 = 310^-4$, $R^2 = 0.98$), demonstrating the existence of an optimal percentage of active nodes, which is around 50% for the sequence classification task (note that the optimal sparsity level is task-dependent). The dashed line connecting the results for diverse training time highlights the change in the sparsity level achieved through the learning rule. **B**: Performance as a function of sparsity and specialisation. The best performance corresponds to the highest specialisation values for all training instances, demonstrating the interpretability of the model.

Finally, we compared the performance of the SpaRCe model with:

1) Echo state network (ESN) without thresholds, where the same on-line learning is applied to the output weights W^{out} only. We note that the algorithm SpaRCe learns N more parameters (the thresholds) in comparison to the Echo State Network without thresholds.

2) Hidden layer (HL), where we added a full hidden layer of N_h nodes on the top of

the reservoir. This approach learns an additional connectivity matrix between the reservoir and the hidden layer, dramatically increasing the number of parameters by a factor of approximately $N_h N$.

3) Echo state network (ESN) with online learning and L_1 or L_2 regularization terms on the output weights.

The SpaRCe model outperforms the standard ESN with or without the regularization terms, based on classification accuracy and root mean square error (Fig. 4A,C,D). This advantage is consistent across different levels of external noise (σ_s) and different numbers of stimuli (Fig. 4C,D). Furthermore, SpaRCe performs comparably to a network with an additional full hidden layer with $N_h \approx 100$ nodes, even though the hidden layer dramatically increases the number of learnable parameters compared to SpaRCe (Fig. 4A,B,D). In comparison to the addition of a hidden layer, the model SpaRCe provides a cheap formulation to achieve an optimal and reliable sparsity level (see Fig. 4B, where the number of learnable parameters for the models are reported with the corresponding performance).

In general, when introducing a hidden layer of N_h neurons trained with backpropagation, for a network of N_o output neurons, we would learn a number of parameters equal to $N \times N_h + N_h + N_h \times N_o + N_o$ (weights + biases + output weights + biases) which can be approximated by $N_h \times N$ (assuming $N_o \ll N$, i.e. $1000 \times 100 + 100 + 100 \times 2 + 2 \approx 10^5$ for the case with $N_h = 100$), while for a classical reservoir we learn $N \times N_o + N_o \approx N \times N_o$ ($1000 \times 2 + 2 \approx 2 \times 10^3$) and for SpaRCe $N \times N_o + N + N_o \approx (N_o + 1) \times N$ ($1000 \times 2 + 1000 + 2 \approx 3 \times 10^3$). While adding a hidden layer goes against the principle of ESN of exploiting the network dynamics while using simple learning methods, it remains an interesting comparison for quantifying the efficiency of the proposed method. With $N_h = 100$ and while $N_o \ll N_h$ our proposed thresholded architecture is efficient in terms of numbers of learnable parameters in comparison to adding a fully trained hidden layer to an ESN.

We conclude that the SpaRCe model considerably improved the performance and the

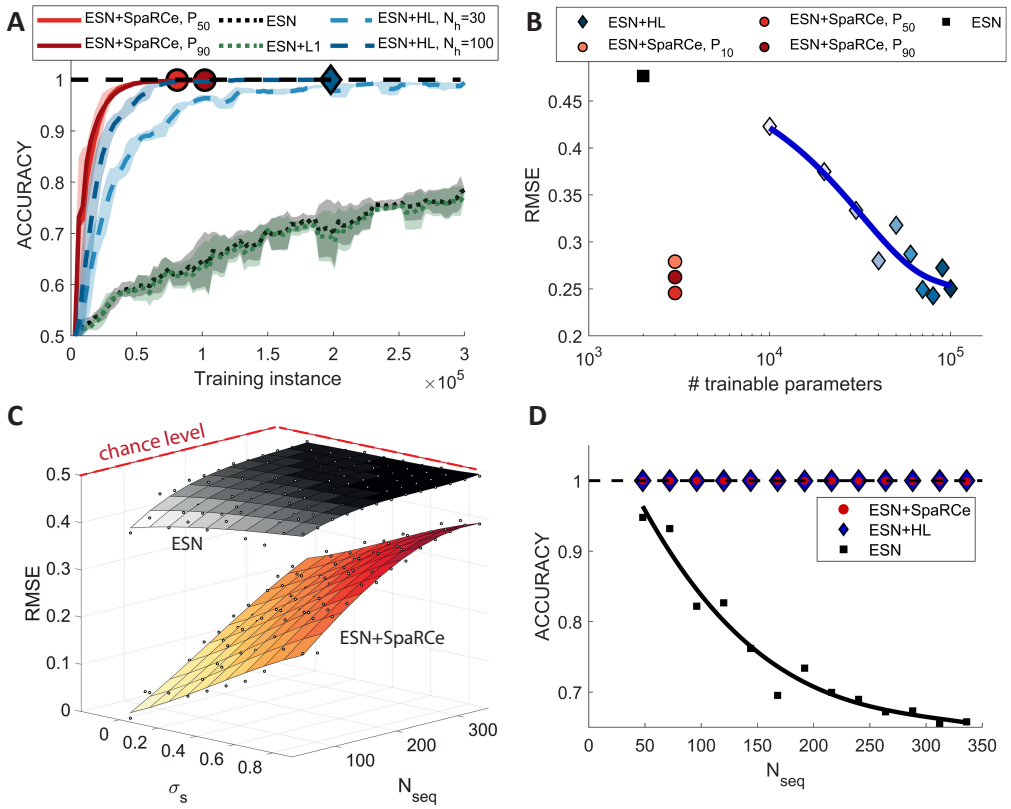

Figure 4: The SpaRCe algorithm increases the memory capacity of the ESN and the stability of the found solution. **A**: Classification accuracy and root mean square error of the models for a case where the number of sequences to be classified is 192. Each minibatch corresponds to the presentation of 20 training samples. In this case, the performance of SpaRCe are related to a starting $P_n = 50$, while the hidden layer has $N_h = 100$ nodes. Dots correspond to the training instances in which the considered models solve the task (for the models that can solve the task), thus showing the training speed of the algorithms analysed. **B**: Performance of SpaRCe for three different starting sparsity levels (red, orange colours), for a standard ESN (black) and of HL (blue colours) for diverse N_h nodes. The x-axis reports the number of trainable parameters and is shown in a logarithmic scale. The graph shows how SpaRCe is able to reach good performance while maintaining a low number of trainable parameters. **C**: Comparison of the root mean square error for the ESN and the ESN with thresholds as the external noise σ_s and the number of stimuli vary. The introduction of thresholds lead to robust result. **D**: Performance as the number of inputs to be classified increases, for the hidden layer model with $N_h = 100$ (blue), SpaRCe (red) with $P_n = 50$ and a standard ESN (black). SpaRCe and HL solves the tasks considered, but the latter uses a number of trainable parameters (reported along the performance with the colour scheme that reflects the referred model) that is of two magnitudes higher than the first. The accuracy of the standard ESN drops considerably as the number of sequences increases. The inset shows the root mean square error for SpaRCe and for HL, varying the starting condition P_n and number of nodes N_h respectively. The errors shown correspond to the training instance in which the fastest model reaches perfect classification accuracy. Numbers reflect the number of trainable parameters for example cases.

convergence time of a reservoir on this biologically inspired benchmark task, with the relatively small overhead of N additional parameters, one per reservoir node. Finally, we investigated the cost of the additional complexity of SpaRCe, composed by an initialisation procedure (the quantile operation over the training data) and threshold learning, in terms of computational time. Considering a training example for this task, SpaRCe required approximately 1234.6 seconds, while an ESN required 1233.98 seconds. The values of the hyperparameters adopted for the task and more details about the evaluation of the computational time required for different models, with the specifics of the computer used for the simulations, can be found in Appendix E.

3.2 Threshold learning increases performance

In this section we faced three variants of classification tasks using the MNIST dataset. Each image is fed into the network sequentially either one column at a time or one pixel at a time to make the task temporally dependent. Thus, one written digit corresponds to a sequence of 28 time steps of a 28 dimensional input in the column by column paradigm, or to a sequence of 728 time steps of a one dimensional input (psMNIST). The tasks considered require representations over multiple timescales to achieve competitive performance, and thus serve as benchmarks to study the ability of the network to discover temporal dependencies. In this regard, the psMNIST problem constitutes a challenge and a benchmark for recurrent neural networks and has been often considered as a metaphor of practical applications where long temporal dependencies are present. Prime examples of such applications are in the fields of language, decision-making, Reinforcement Learning in partially observable Markov processes. We want to emphasize how all the tasks considered, even the MNIST and pMNIST, are processed by the network temporally. For this reason, the performance reported of static networks, as multilayer perceptrons (MLP) and convolutional neural networks, serve only as a baseline, since these networks could not solve the temporally dependent task faced. The activation function f used for these tasks is a hyperbolic tangent.

MNIST, column by column

The application of ESNs on this specific task was previously analysed in [41], in which the original dataset was preprocessed and augmented by resizing and deforming the original images. Without such a preprocessing, the ESN could not outperform a simple perceptron [41]. In contrast, in this work we use the original dataset, without any additional transformations. In this experiment, the pixels of each image are fed to the ESN sequentially column by column; thus, the input signal $\mathbf{s}(t)$ corresponding to an example image is a 28-dimensional sequence of temporal length 28. In this case, $\tilde{\mathbf{V}} = \mathcal{C}\Big(\{\mathbf{V}(t)\}_{\forall t}\Big)$, meaning that all the dynamic of the recurrent network across time is used to define the read-out of the system. The cost function adopted is a sigmoidal cross entropy

$$E = -\Big[\sum_j \tilde{y}_j \log\Big(\sigma(y_j)\Big) + (1-\tilde{y}_j)\log\Big(1-\sigma(y_j)\Big)\Big] \tag{9}$$

which is analysed in B.2 (Appendix). The optimizer used is Adam [54]. We first tested the SpaRCe model with various initial sparsity levels (Fig. 5A, different colours). Regardless of the initial condition, the final performance was similar, as was the final level of sparsity (the size of dots in Fig. 5A shows the percentage of active nodes in the network). The error achieved by SpaRCe is 1.9%. The hyperparameter values are given in Appendix E. The model reaches performance levels comparable to those achieved with a three-layer neural network with backpropagation [55]. However, the task faced here with SpaRCe (column-by-column MNIST) is more challenging than the common approach used to train neural networks on the MNIST dataset, in which the whole image is fed into the network at once. The performance of convolutional neural networks and of multilayer perceptrons are reported in Table 4 as a benchmark for the results obtained with ESN with SpaRCe. Convolutional neural networks are best suited for image classification problems and nevertheless SpaRCe is only 0.2% from the lower reported performance. In this comparison, we have not matched the number of parameters between SpaRCe and the other models; we have used a reservoir with 1000 neurons. To demonstrate the importance of thresholds

in the SpaRCe model, we compared the performance of SpaRCe to that of an ESN without thresholds trained online with the same optimizer, using a learning rate optimized through grid search. SpaRCe outperformed the ESN in both classification accuracy and convergence time (Fig. 5B, 'MNIST'). We also found that the normalisation mechanism introduced thanks to SpaRCe stabilises the learning process for relatively higher learning rates vs standard ESNs. For this simulation, the total computational time required by SpaRCe is 579.81 seconds, while for a standard ESN is 474.51 seconds (as before, see Appendix E for more information). As a second task, we applied the same model to the MNIST dataset where the pixels of each image are reordered through a permutation of the data. Each image is again fed one column at a time. Again, SpaRCe outperformed the ESN without thresholds (Fig. 5B, 'pMNIST').

Results			
MNIST			
ESN			95.2
$SpaRCe$			98.1
MLP			97.0|98.5*
$Conv.$			98.3|99.6*
psMNIST			
Reservoir size	500	1000	1500
ESN^2	95.6	95.9	96.3
$SpaRCe^2$	96.2	96.6	96.9
$LSTM$	89.9**		
NRU	95.4**		

Table 1: The asterisks * ** indicate that the results are taken from [55] and [44] respectively. MLP stands for multilayer perceptron (three layers and without distortions), $Conv.$ for single convolutional networks (without distortions), NRU for Non-gated Recurrent Unit and $LSTM$ for Long Short Term Memory. The superscripts in ESN^2 and $SpaRCe^2$ indicate that each network was composed by two reservoirs (see text). The paramteter "reservoir size" corresponds to the total number of neurons in both reservoirs (see also Appendix E).

psMNIST

We next analysed the performance of the SpaRCe model and of ESNs in general on the psMNIST (permuted sequential MNIST) task, which has became a standard benchmark for recurrent neural networks [44]. In this case, each image is reordered through a fixed permutation and fed into the recurrent network one pixel at a time. The task is particularly challenging because temporal dependencies must be learned between widely-separated time steps of an image. Since each sequence is a succession of 784 pixels, we decided to sample the dynamic of the reservoir across time at constant steps multiples of 28, defining $\tilde{V} = \mathcal{C}\Big(\{\mathbf{V}(t) : 28|t\}\Big)$, where $28|t$ indicates that t has to be divisible by 28. This sampling procedure at constant temporal time steps could be suboptimal compared to an approach where the most informative time steps are selected. However, an analysis of optimal sampling procedures goes beyond the scope of this paper.

Our first attempts to solve the task with a single ESN gave performance comparable to or worse than a standard perceptron model trained on the whole image, because the ESN could not simultaneously discover long time dependencies and quickly adapt to new inputs. Indeed, in any network of randomly-connected low pass filters such as ours, it can be difficult to associate events that are distant in time, as it may be impossible to find a workable balance in the trade-off between keeping a memory of past events (which requires each node's activity to have a long time constant, i.e., slow decay) vs. allowing the network to evolve dynamically over time (which requires a short time constant, i.e., fast decay).

To overcome this difficulty, we used an architecture composed of two reservoirs. The first has 200 nodes with fast time constants, and the second has 300 nodes with slower time constants (the parameters for the two reservoirs are reported in Table 2 of Appendix E). The faster reservoir signals unidirectionally to the slower one, and nodes of both reservoirs are used for the read-out. This type of structure, where the timescales of the first network are faster than the second, was found to be optimal in previous works [35].

The sparsity level in the connectivity of the faster reservoir is important: it regulates a trade-off between too much connectivity (the relation between the input information to the second reservoir and the input signal becomes too complex) vs. not enough connectivity. The best performance arises when the network has the shortest path lengths between two nodes that could permit a sufficient amount of memory.

Using this architecture, SpaRCe outperforms published methods despite using a much simpler training algorithm. As with MNIST and pMNIST, on psMNIST SpaRCe reaches higher accuracy and converges faster than the threshold-less ESN (Fig. 5B). To compare SpaRCe with published models, two ESNs with a total of $N = 500$ nodes (thus 154×10^3 parameters) were adopted. The model gave an accuracy > 0.96, higher than the best, more complex recurrent neural networks that exploit backpropagation through time (BPTT), whose performance are 0.954 (NRU) and 0.899 (LSTM) [44] (Table 4) with a comparable number of parameters ($\approx 167 \times 10^3$). While BPTT needs to unroll all the dependencies of the neural network activities across time, ESNs have the advantage to train a much simpler perceptron on top of the reservoir representation. In particular, considering that the dynamic of a reservoir across datasets can be computed once only and then used to train a high dimensional perceptron, the computational cost of ESNs is lower than the computational cost of RNNs. We emphasize that the procedure of concatenating previous temporal representations is not simply a shortcut, but it is necessary to increase the dimensionality of the representation in order to solve complex machine learning tasks through a reservoir computing approach. Indeed, the idea behind reservoir computing is to exploit the temporal dynamic of a system as a fixed and higher dimensional representation that allows it to separate the classes of a classification task through an hyperplane. This approach contrasts with the learning process of a recurrent neural network with backpropagation through time, which trains the dynamics of the system and draws a nonlinear manifold to solve the classification task. Furthermore, the concatenating procedure does not guarantee any understanding of the long temporal dependencies among pixels that are necessary to effectively solve the problem. It is therefore of interest that ESNs using SpaRCe could

perform comparably to state of the art recurrent networks, whose parameters are trained via a far more complex algorithm, backpropagation through time. Finally, we repeated the experiment by changing the total number of nodes in the network, therefore increasing the dimensionality of the representation and the number of trainable parameters. In such a way, it is possible to understand if the performance difference between SpaRCe and vanilla ESN can be alleviated by simply increasing the network size of the latter. The results of this analysis are reported in Table 4 for three different numbers of nodes ($N = 500, 1000, 1500$). Since the model is composed by two reservoirs in this case, N indicates the total number of nodes in the network (see Appendix E for more details). The standard ESN reaches the accuracy of SpaRCe only when its number of trainable parameters is approximately tripled (similar performance of $SpaRCe^2$ with $N = 500$ and ESN^2 with $N = 1500$), while the performance difference remains unaltered if we increase the network size for both models. The difficulty found in the attempt to compensate for the performance improvement of SpaRCe permits us to evaluate the relatively small difference between the two models (ESN and SpaRCe) and to demonstrate how the impact of the proposed model is meaningful even on the psMNIST. This result, and the small computational time required by SpaRCe (reported and compared to ESN gradient descent training in Appendix), shows how the proposed model constitutes a relatively inexpensive procedure considering its impact in terms of performance benefit.

3.3 SpaRCe alleviates catastrophic forgetting

Catastrophic forgetting refers to the inability of standard neural networks to learn different tasks sequentially. If a neural network is trained on a specific dataset and then retrained to perform a novel task, it will probably forget what it has learned before. This unsolved problem [45] is critical for the future development of neural networks in general. Previous research formulated models that mitigate catastrophic forgetting, using a variety of techniques categorised by Kemker et al. [45]. These techniques include regularization (impose a cost to changing the weights that contribute to previous tasks, as in Elastic Weight

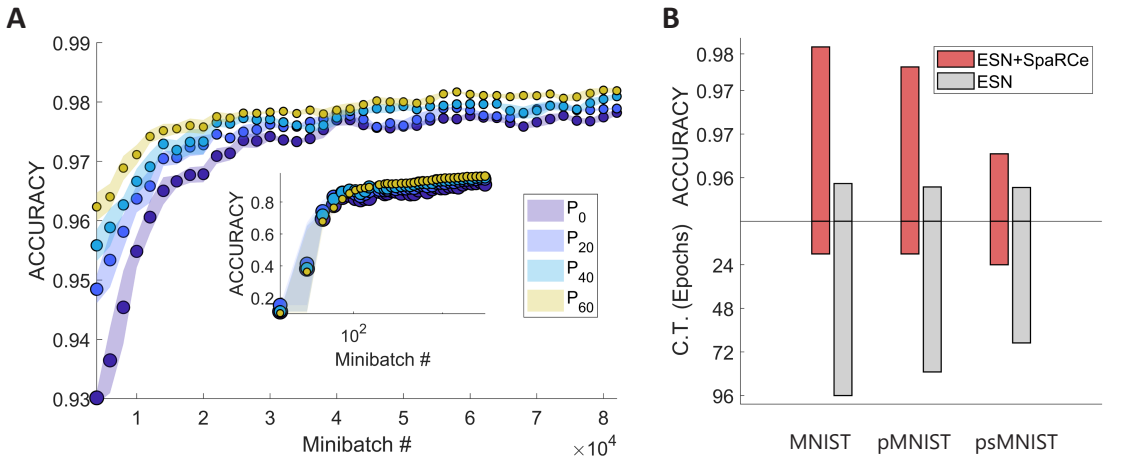

Figure 5: The SpaRCe model shows comparable performance to a 2/3 hidden layer neural network on the MNIST dataset and accuracy comparable to more complex RNNs trained with BPTT on the psMNIST task. On the contrary, the standard ESN with on-line training leads to lesser performance. **A**: The sizes of the dots reflect the percentage of active nodes (sparser network=smaller dots) in the network. Each minibatch corresponds to the presentation of 20 training samples. The abscissa of the inset figure is scaled logarithmically. **B**: Performance of ESN with (red) and without (grey) threshold learning on the three tasks analysed, measured by accuracy and convergence time (C.T.). The network with the SpaRCe model outperforms the standard ESN read-out on all the benchmarks, but the contribution of the thresholds decreases as the task becomes more complex. This can be understood by considering that the increasing complexity of the tasks from left to right of the graph arises from a greater demand of the network's ability to understand long term dependencies. This aspect depends on the system dynamics and is not strongly related to threshold learning. Furthermore, the SpaRCe model converges about 5 times faster than an ESN without thresholds.

Consolidation, or EWC), rehearsal (re-playing previously learned data during subsequent training, as in GeppNet), and sparse coding (reducing the fraction of active nodes, as in the Fixed Expansion Layer model, or FEL, and Hard Attention to the Task, or HAT [46]). Sparse coding is also the approach we use here, with SpaRCe. However, in contrast to these techniques that exploit additional information, such as model awareness of the task identity, and the computation of ad-hoc quantities, such as the importance of specific parameters (EWC) or nodes (HAT) for a given task, we will demonstrate how the application of Eq. 4 alone with a high starting sparsity level can alleviate catastrophic forgetting. Notably, the same methodology applied in the previous simulations, to improve convergence time and performance of reservoir computing, can improve the ability of an ESN to

learn different tasks sequentially. The difference between the application of SpaRCe in a single task and in a sequential tasks paradigm is the tuning of the hyperparameters of the proposed model, in particular the percentile value n and the learning rates η_W and η_θ. We will demonstrate how sparsity regulates a trade-off between initial learning speed vs. memory retention because high sparsity decreases overlaps among representations, which prevents new learning from disrupting old memories, whereas low sparsity means more nodes are active, allowing memories to be formed faster on new tasks.

This trade-off between initial learning vs. preventing forgetting is studied in detail for two paradigms that are commonly used to measure catastrophic forgetting in neural networks:

- Sequential data permutations. A different permutation is applied to the considered dataset N_{task} times. These new N_{task} reshuffled datasets are then learned sequentially by the system. In our simulation, $N_{task} = 10$. In such a scenario, the complexity of the different datasets is the same, and we trained SpaRCe for approximately two epochs for each task.

- Sequential classes. The first task is composed by the data corresponding to half of the possible classes, while the other classes are trained sequentially. Since in the dataset considered there will be 10 classes, the number of tasks that the network has to learn sequentially is six. In this case, we trained the model for approximately one epoch for each task.

The dataset used is MNIST, where the reservoir processed every image column by column as in Section 3.2, adopting $\tilde{\mathbf{V}} = \mathcal{C}\left(\{\mathbf{V}(t)\}_{\forall t}\right)$ as before. Of course, the model is able to learn from the data corresponding to a task only once. We learn from the training dataset and compute the accuracy on the testing dataset by varying the initial starting sparsity levels through grid search of the value of n in Eq. 4 and as the number of tasks considered varies (Fig. 6C,D). These performances are used for comparison and demonstration only, and a validation dataset will be used to select the best hyperparameters

setting. In comparison to previous simulations, the learning rate for the thresholds is smaller (see also Table 3 of Appendix E), because it was crucial to prevent a dramatic change of sparsity levels during learning, as such an abrupt change in the percentage of active nodes would alter stimulus representations and thereby affect previously learned tasks. In both tasks, lower sparsity levels allowed better initial learning on novel data (Accuracy across $N_{task} = 1$, Fig. 6C,D), while higher sparsity levels alleviated forgetting of previous tasks during subsequent training. In other words, low sparsity allows good performance when the number of tasks is low, but the accuracy decreases quickly when N_{task} increases. On the contrary, at optimal sparsity levels, accuracy remains high even as the number of tasks increases (highlighted path on the surface plot in Fig. 6C,D). These two conflicting trends combined make the total accuracy across all datasets an inverted U when plotted against sparsity level (Fig. 6A,B).

Moreover, we computed the following quantities to measure the alleviation of catastrophic forgetting in more detail:

$$\alpha_{Overall} = \frac{1}{N_{task} - 1} \sum_{n=2}^{N_{task}} \frac{acc_n}{acc(1,1)} \tag{10}$$

$$\alpha_{Memory} = \frac{1}{N_{task}} \sum_{n=1}^{N_{task}} [acc(n, N_{task}) - acc(n, n)] \tag{11}$$

$$\alpha_{New} = \frac{1}{N_{task}} \sum_{n=1}^{N_{task}} acc(n, n) \tag{12}$$

where acc_n is the accuracy computed on the datasets seen until task number n (included), and $acc(1, 1)$ is the accuracy of the first dataset immediately after its learning, which is considered as the ideal baseline. In general, we denote with $acc(n, m)$ the accuracy of the n-th dataset after the presentation of m datasets. The performance metric α_{Memory} measures model's ability to remember previous tasks, i.e., to alleviate catastrophic forgetting, while the metric α_{New} measures the model's ability to learn new tasks. In other terms, α_{Memory} is the average difference over tasks between the performance obtained after the whole training and immediately after the presentation of a specific dataset, while α_{New}

is the average performance of a dataset after its presentation. These metrics and $\alpha_{Overall}$ are taken from [45], where the performance of various models that alleviate catastrophic forgetting in multilayer perceptrons are compared.

Finally, we repeated the simulation ten times for each of the two paradigms considered, averaged over different possible permutations (sequential data permutations) and for different ways of dividing the data into separate classes and corresponding tasks (sequential classes), selected the best performing algorithm based on the validation dataset and computed its performance $\alpha_{overall}$ on the test dataset. The performance obtained is reported in Table 5 together with the results obtained by a standard ESN and published models that use a variety of strategies to prevent catastrophic forgetting in multilayer perceptrons. Of course, we do not expect to compete with newer Deep learning methods that exploit additional information and many more parameters. The results reported are obtained following, to the best of our knowledge, the same methodology as [45], and differ only in the necessity of an ESN to process the data temporally, rather than feeding the whole image to the network at once. In this aspect, the temporal processing of the data can make the task only more challenging for our model, since catastrophic forgetting appears to constitute an even more difficult problem to recurrent neural networks [56]. We need to highlight, however, a drawback of our model: the tuning of an additional hyperparameter, that is the starting sparsity level P_n regulated by the proposed normalisation mechanism. However, we note that all other methods that alleviate catastrophic forgetting in neural networks usually have additional hyperparameters (set through heuristics or grid search), as the learning rate of the penalty term in EWC.

If the results on multilayer perceptrons can be considered exclusively as a useful baseline, we notice from Table 5 the complete inability of an ESN to solve sequential learning. Indeed, the training of the read-out weights of an ESN in the standard reservoir computing paradigm causes a complete forgetting of previous tasks, resulting in an overall performance that corresponds to the learning of the last task processed by the model.

The success of SpaRCe on these catastrophic forgetting tasks arises from both the

initial sparsity and from threshold learning. We analysed the relative importance of the starting sparsity level vs. the online threshold adaptation introduced by the learning rule in the Appendix.

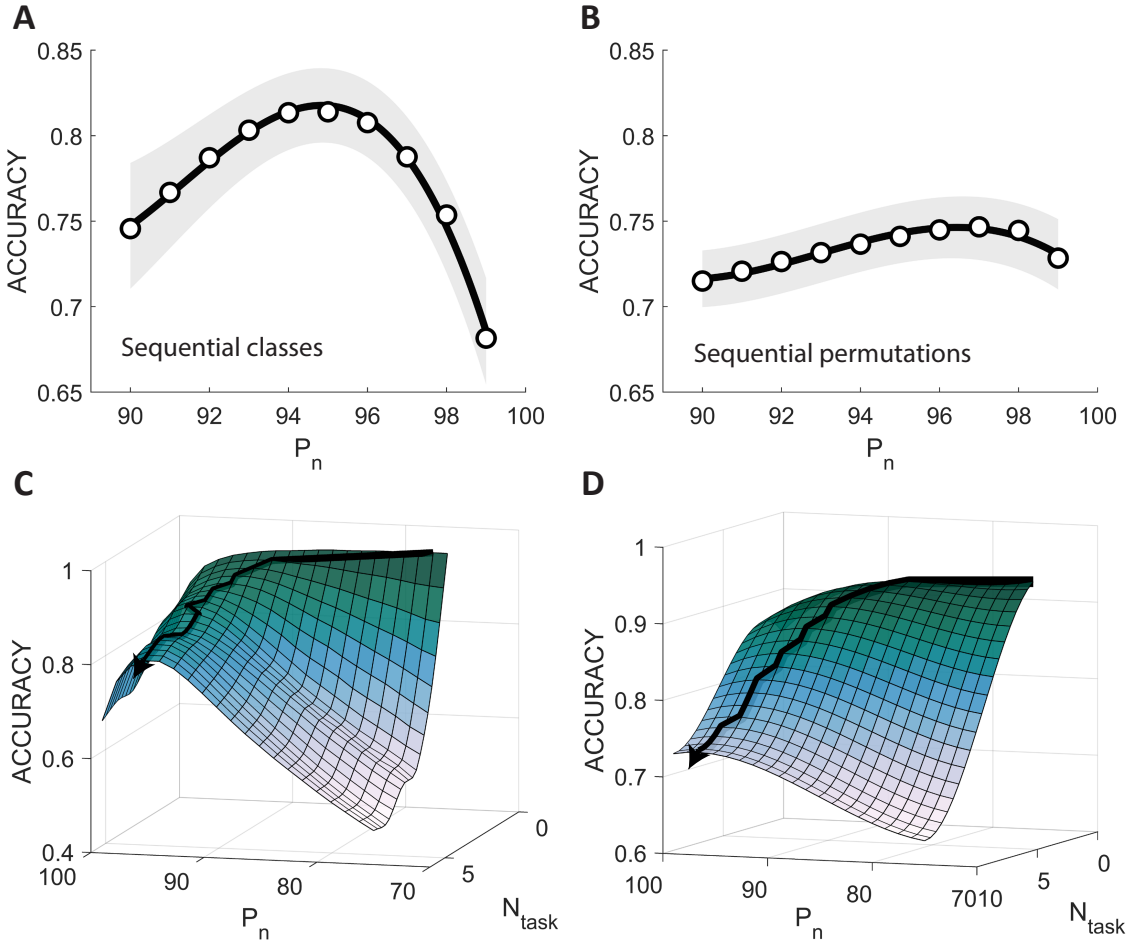

Figure 6: SpaRCe helps to prevent catastrophic forgetting on the two analysed benchmarks. Different data points correspond to diverse repetitions of the experiment. **A**: Results on the MNIST dataset in the sequential classes paradigm (see text). **B**: Results on the permuted datasets paradigm. **C**-**D**: Performance as function of the starting percentage of active nodes and the number of tasks that are learned in the catastrophic forgetting simulations (**C** refers to sequential classes and **D** for permuted datasets). The surface is a cubic interpolation of the accuracy as the number of datasets and the starting sparsity level vary. The path shows the best performing sparsity levels across various number of tasks; its movement from right to left demonstrates the necessity of adopting increasing level of sparsity as the number of datasets increases and the memory of previous tasks becomes more relevant.

Results, $\alpha_{overall}$			
Sequential classes		Sequential permutations	
ESN	0.1	ESN	0.1
$SpaRCe$	0.870	$SpaRCe$	0.897
EWC	0.133^{+}	EWC	0.746^{+}
FEL	0.439^{+}	FEL	0.279^{+}
$GeppNet$	0.922^{+}	$GeppNet$	0.364^{+}

Table 2: SpaRCe performs comparably to the best of the ad hoc models, which tend to perform well only on one of the two tasks analysed. The symbol $^{+}$ indicates that the results were taken from [45].

4 Discussion

It is customary in Machine Learning to introduce sparsity via regularisation: an ad-hoc penalty term is added to the network's error function to penalise the use of the weight parameters, leading to solutions with smaller (or sparser) weight values. In this work, inspired by the insect mushroom body, we propose a simple and elegant modification on the standard ESN that leads to sparse representations. We introduce a threshold per reservoir neuron, adaptable via *on-line* gradient learning, thereby associating sparsity directly to network performance. This results in active neurons that preferentially fire for one class vs another, which, in turn, leads to improved performance without disturbing the reservoir dynamics. In our setup, threshold learning and weight learning is a two-way interaction: the threshold changes are proportional to the weight values. In practice, we have found that in most cases an increase in neuronal specialisation follows large weight changes.

The threshold learning rule is structurally identical to the update rule for the bias in backpropagation. This is not surprising as our formulation is equivalent to the introduction of a hidden layer with as many neurons as the reservoir and a one-to-one fixed connectivity. The only learnable parameters are therefore the neuronal thresholds in the hidden layer. However, we demonstrated that it is advantageous to only learn these thresholds in comparison to the full hidden layer. In our simulations, a hidden layer with approximately 10^5 learnable parameters was required to catch up with SpaRCe, which

required only 3×10^3 learnable parameters, and training was twice slower. Our technique is also conceptually similar to the Extreme Learning Machines, which has been also applied in reservoir computing [57]. Nevertheless, allowing for threshold learning increases the memory capacity of the network and we have demonstrated that it can discover the optimal sparsity level.

We also found that the threshold learning helps stabilising the network in the case that a large learning rate has been selected. We have demonstrated (Appendix B.2) that high learning rates that lead to instabilities in the non-threshold model are appropriate for the threshold model: the thresholds act as a stabilisation mechanism, by decreasing the activity of the network through neuronal deactivation. This allows for larger parameter areas and reduces the requirement of fine tuning for the learning rates.

Notably, our model competes with feedforward and recurrent networks on standard benchmark problems (MNIST, sequential MNIST and permuted sequential MNIST), and is always best or close to the best alternative algorithm. This is an impressive result given the simplicity of our model. While in general an ESN is unlikely to compete with more complex networks, SpaRCe permits ESNs to achieve performance levels that were not possible before, at least not without augmentation of the dataset or in combination with other algorithms (e.g. [58]).

Perhaps less obviously, threshold initialisation is key to achieving consistent performances. If the initialisation is entirely random, neurons with excessively high initial thresholds would never fire during the stimulus presentation. Effectively, such neurons would be removed from the network for the whole duration of the simulation. To prevent this issue, the training dataset is first presented to the recurrent network, and we observe the operational activity range of each neuron. This allows us to set up a threshold within this regime in a "fair" way, making sure that each neuron is active for a predecided percentage of time, across all stimulus presentations. This process has a relatively small computational overhead over the ESN; however, it is possible to learn the activity percentile entirely on-line (Appendix B.3) without significant performance loss.

Our model competes with published models across two standard tests for catastrophic forgetting. In fact, sparsity alone, without threshold learning, significantly helps in the case of catastrophic forgetting, but threshold learning adds to the ability of better learning newer sets. In general, however, we do not expect to outperform complex new methods that exploit additional information: our model, following the principle of ESN and reservoir computing in general, uses inherent properties of the network (e.g. dynamics, sparsity) to boost performance in classification tasks and in catastrophic forgetting.

Our work may lead to a reinterpretation of the role of thresholds in neural networks. We have shown that by having a layer where learning takes place via threshold adaptation only and by disentangling the learning of the thresholds from the learning of the weights, via different learning rates, we were able to achieve sparse solutions. We were also able to demonstrate mathematically that sparsity is shaped by effectively "removing" redundant neurons from the reservoir. We believe that this work might be applicable to network structures beyond ESNs.

Finally, reservoir computing is of increasing interest to the neuromorphic computing community, particularly to those who aim to use material dynamics for computation. For instance, in the spintronic community, magnetic devices are proposed as reservoir replacements, and more complex methods such as deep learning could not be implemented in the material level. In the context of the ESN, the reservoir serves only as a spatiotemporal kernel [59, 60], i.e. it increases the dimensionality of the input signal in order to allow a linear model (a perceptron) to separate the classes. Therefore, it can be replaced by any highly non-linear but non-chaotic system able to transform its input to an appropriate higher dimensional space. Such proof of concept systems can be found for instance in [61] [62]. Our algorithm does not impose any modification to the reservoir itself, which allows its use even when the recurrent network is replaced by a physical material.

5 APPENDIX

APPENDIX A: INITIALISATIONS

Reservoir initialization

The equation describing the dynamic of reservoir of leaky integrators is

$$\mathbf{V}(t+1) = (1-\alpha)\mathbf{V}(t) + \alpha f\left[\gamma \mathbf{W}_{in}\mathbf{s}(t) + \rho \mathbf{W}\mathbf{V}(t)\right] \tag{A1}$$

where $\mathbf{W}$ is a random sparse connectivity matrix whose eigenvalues are uniformly distributed inside the unit circle of the imaginary plane, γ the gain factor of the input signal and $0 < \rho < 1$ is a constant. The rescaling factor ρ is called spectral radius and it is explicitly defined to control the maximum absolute value of the eigenvalues of the matrix $\rho\mathbf{W}$. The fact that the eigenvalues of the connectivity matrix $\rho\mathbf{W}$ are constrained inside the unit circle of the imaginary plane is a necessary condition for the Echo State property of the network. Given the eigenvalues $\lambda_{\mathbf{W}}$ of $\mathbf{W}$, the eigenvalues λ of the linearised dynamic system associated to Eq. A1 are

$$\lambda = (1-\alpha) + \alpha\rho\lambda_{\mathbf{W}} \tag{A2}$$

and thus $\lambda_{\mathbf{W}}$ are compressed by a factor α and translated by a factor $1-\alpha$ in the imaginary plane. As a consequence, λ follows the probability distribution

$$p(x,y) = \begin{cases} \dfrac{1}{\pi\alpha^2\rho^2}, & \text{if } \left[x-(1-\alpha)\right]^2 + y^2 \leq \alpha^2\rho^2 \\ 0, & \text{otherwise} \end{cases} \tag{A3}$$

where $x = Re(\lambda)$ and $y = Im(\lambda)$ for simplicity of notation. Since the real part of the eigenvalues is associated to the timescales τ of the dynamic system as $Re(\lambda) = \exp(-\dfrac{\delta t}{\tau}) \approx 1-\dfrac{\delta t}{\tau}$, it is possible to compute the marginal distribution over x of $p(x,y)$ for the real part, and then compute the distribution of timescales. A simple strategy to choose α and ρ by

knowing the range of the timescales $[\tau_m, \tau_M]$ that the network should exhibit is to notice how the fastest (slowest) timescale τ_m (τ_M) is given by the minimum (maximum) real eigenvalue of the dynamic system. Calling $\lambda_m = min\{Re(\lambda)\}$ and $\lambda_M = max\{Re(\lambda)\}$, and recalling we have

$$\begin{aligned}\lambda_m = 1 - \alpha + \alpha\rho\ (-1) = 1 - \alpha(1+\rho) = \\ = \exp(-\delta t/\tau_m) \approx 1 - \frac{\delta t}{\tau_m} \rightarrow \\ \rightarrow \alpha(1+\rho) \approx \frac{\delta t}{\tau_m}\end{aligned}$$

and

$$\begin{aligned}\lambda_M = 1 - \alpha + \alpha\rho\ (+1) = 1 - \alpha(1-\rho) = \\ = \exp(-\frac{\delta t}{\tau_M}) \approx 1 - \frac{\delta t}{\tau_M} \rightarrow \\ \rightarrow \alpha(1-\rho) \approx \frac{\delta t}{\tau_M}\end{aligned}$$

Solving the system above, we end up with

$$\alpha \approx \frac{\delta t}{2\tau_m} + \frac{\delta t}{2\tau_M}$$

and

$$\rho \approx \frac{\delta t}{2\alpha\tau_m} - \frac{\delta t}{2\alpha\tau_M}$$

that are relations between α, ρ and the minimum and maximum timescales that the model can exhibit. In this way, it is possible to choose the hyperparameters α and ρ by selecting

a priori the more interpretable parameters τ_m and τ_M. We want to emphasize that this procedure does not guarantee an optimal choice of the hyperparameters, but it can guide the search and it assures a good choice in terms of temporal memory of the reservoir.

Thresholds initialization

Imposing a democratic initialisation where each node has the same probability to be active, the initial condition and sparsity level are defined by the choice of the starting percentile P_n. Here we defined two approaches to choose P_n:

- A simple grid search over n. Here, N_P reservoirs are trained in parallel for the first 10% of time steps in the training instance, and the best performing reservoir is selected for the remainder of training. From our results, a small fraction of the training time is enough to choose the starting condition without any loss in the performance.
- Select the sparse representation that leads to the highest value of specialisation, a measure of the quality of the sparse representations that is defined below.

The measure of *specialisation* (Sp) reflects how a level of sparsity can facilitate the learning process in a classification task. The assumption behind the following formulation is that for a good sparse representation the ensembles of active nodes for different classes should overlap as little as possible. Let us consider two classes j and k and a neuron i. The node is specific if there is an asymmetry in the number of times it is active for one class with respect to the other. Generalizing this idea it is possible to build a measure $spec_{ijk}$ for a node i defined as

$$spec_{ijk} = |\frac{N_{ij}}{M_j} - \frac{N_{ik}}{M_k}| \tag{A4}$$

where N_{ij} (N_{ik}) is the number of times the neuron i was active after the presentation of a stimulus of class j (k) and M_j (M_k) is the total number of presentations of the stimuli belonging to class j (k). Since the denominator of Eq. A4 contains the total number of

presentations, $spec_{ijk}$ does not simply increase with the level of sparsity introduced. Let us focus on the particular case where $M_j \approx M_k$. A too high level of sparsity would lead the node to be almost silent, with a consequent poor specialisation value due to N_{ij} and N_{ik} being both close to zero. On the contrary, a too low sparsity level would lead the neuron to be excessively responsive, and $spec_{ijk}$ would be poor because $N_{ij} \approx N_{ik}$ even if N_{ij} and N_{ik} are both high.

Given $spec_{ijk}$ it is possible to compute a measure of specialisation for each single neuron as

$$Sp_i = \langle spec_{ijk} \rangle_{jk}^{(>0)} \tag{A5}$$

where $\langle \cdot \rangle_{jk}^{(>0)}$ is the average over positive elements for the indexes jk. It is possible to select the starting initial values of the thresholds as the n-th percentile of the distribution $\mathbf{V}$ that leads to the highest specialisation measure. Figure 3 (Main text) shows how the best performing sparse representation corresponds approximately to the maximum value of the average specialisation across neurons $Sp = \frac{1}{N} \sum_i^N Sp_i$.

APPENDIX B: THRESHOLD LEARNING

Gradients on thresholds

The training procedure minimizes a measure of the distance $E(t)$ between the output $\mathbf{y} = \mathbf{W}^o\mathbf{x}$ of the neural network and the desired value $\tilde{\mathbf{y}}$. Theoretically,

$$E = dist^2\Big(\tilde{\mathbf{y}}, \mathbf{y}\Big) \tag{B1}$$

We will now apply a gradient based optimization on an example cost function, and show how the resulting learning rule for the thresholds can be interpreted.

Gradient on θ, Mean Square Error (MSE)

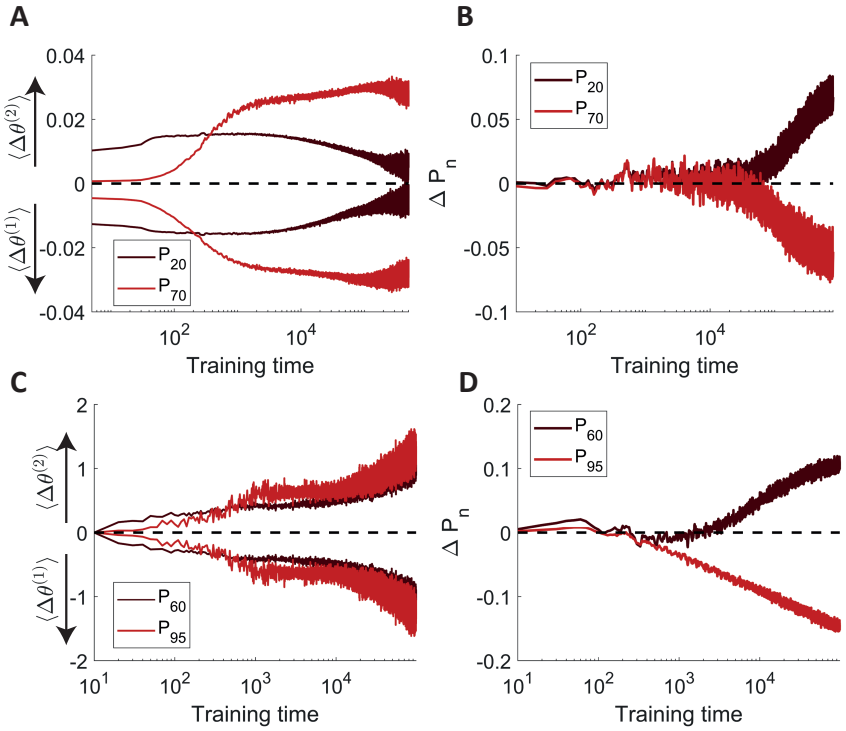

Figure 7: $\Delta\theta^{(1)}$ and $\Delta\theta^{(2)}$ drive the learning process of the thresholds. **A-C:** Values of $\Delta\theta^{(1)}$ and $\Delta\theta^{(2)}$ averaged across the population of nodes for two different starting sparsity levels. These two factors have interpretable meaning (see Main text). **B-D:** Average of the cumulative threshold change. If the starting condition is suboptimal and low (high), such an average will be positive (negative) and consequently increasing (decreasing) the level of sparsity. Panels **A** and **B** refer to the storage capacity task of Section 3.1, where the cost function is a mean square error, while panels **C** and **D** refer to the classification of the MNIST dataset in the column by column paradigm of Section 3.2, where a sigmoid cross-entropy function was adopted. In the latter case, the two factors are defined in Eq. B10 and B9.

Let us consider the mean square cost function, given by

$$\begin{aligned} E = dist\Big(\tilde{\mathbf{y}}, \mathbf{y}\Big) = \\ = \sum_j \Big[\tilde{y}_j - y_j\Big]^2 = \\ = \sum_j \Big[\tilde{y}_j - \sum_i W^o_{ji} x_i\Big]^2 = \\ = \sum_j \Big[\tilde{y}_j - \sum_i W^o_{ji} \text{sign}(\tilde{V}_i) \text{ReLU}\Big(|\tilde{V}_i| - P_n(|\tilde{\mathbf{V}}_i|) - \tilde{\theta}_i\Big)\Big]^2 \end{aligned} \tag{B2}$$

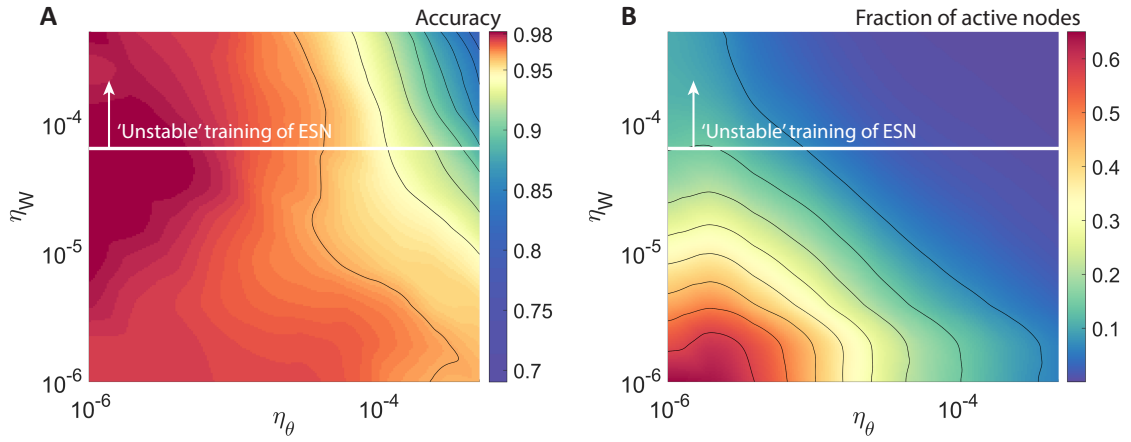

Figure 8: The proposed model acts as a stabilisation mechanism that permits the utilisations of higher learning rates in comparison to the values adopted in the standard ESN read-out. The result are obtained on the MNIST task of Section 3.2 in the column by column paradigm. For all the results shown, the model is initialised from a starting sparsity level $P_n = 50$. We plot the accuracy (**A**) and the sparsity level (**B**) as the learning rates η_W and η_θ vary. **A**: The best performance of the algorithm corresponds to a region where the learning rate on the thresholds is at least 10^{-1} times lower than the learning rate on the output weights. This result is expected, considering that the learning rule on the thresholds depend explicitly on the output weights (Eq. B3), and thus thresholds can be accurately learnt when the output weights carry information on the classification process, i.e. weights are learnt faster than thresholds. **B**: The percentage of active nodes decreases as the learning rates η_W and η_θ increase. This can be understood considering that, when the learning rates are high, the model avoids abrupt changes on the output $\mathbf{y}$ by decreasing the activities of the representation, i.e. increasing the sparsity level of the network. The parameter space above the white horizontal line corresponds to a region where learning with the standard ESN read-out is unstable because of the too high value of η_W adopted. In that region, training is characterised by an undesirable increase of the cost function across learning for the ESN but not for SpaRCe.

where we have used a read-out of Eq. 2 (Main text) to define the output of the neural network. A gradient based approach that minimizes E leads to the following learning rule on the output weights

$$\Delta W^o_{lk} = -\eta_W \frac{\partial E}{\partial W^o_{lk}} = \\ = \eta_W \Big[\tilde{y}_l - y_l\Big] x_k$$

and to the following learning rule for the thresholds

$$\begin{aligned}
\Delta\theta_k = -\eta_\theta \frac{\partial E}{\partial \theta_k} = -\eta_\theta \frac{\partial E}{\partial \tilde{\theta}_k} = \\
= \eta_\theta \sum_{j=1}^{N_{class}} [\tilde{y}_j - y_j] \frac{\partial}{\partial \tilde{\theta}_k} \Big\{ \sum_i W^o_{ji} x_i \Big\} = \\
= -\eta_\theta \sum_{j=1}^{N_{class}} [\tilde{y}_j - y_j] W^o_{jk} \text{sign}(\tilde{V}_k) H\Big(|\tilde{V}_k| - \\
- P_n(|\tilde{\mathbf{V}}_k|) - \tilde{\theta}_k \Big) = \\
= -\eta_\theta \sum_{j=1}^{N_{class}} [\tilde{y}_j - y_j] W^o_{jk} \text{sign}(x_k) = \\
= -\eta_\theta \sum_{j=1}^{N_{class}} \tilde{y}_j W^o_{jk} \text{sign}(x_k) + \\
+ \eta_\theta \sum_{j=1}^{N_{class}} y_j W^o_{jk} \text{sign}(x_k)
\end{aligned} \tag{B3}$$

By taking into account the specific case of a classification task where $\tilde{y}_j$ is positive for j that corresponds to the desired class and zero otherwise, it is possible to manipulate Eq. B3 and to separate it in two terms to uncover the meaning of the learning on the thresholds.

$$\begin{aligned}
\Delta\theta_k = -\eta_\theta \beta W^o_{\tilde{j}k} \text{sign}(x_k) + \\
+ \eta_\theta \sum_{j=1}^{N_{class}} y_j W^o_{jk} \text{sign}(x_k) \\
= -\eta_\theta \beta W^o_{\tilde{j}k} \text{sign}(x_k) + \\
+ \eta_\theta \sum_{j=1}^{N_{class}} \sum_{l=1}^{N} W^o_{jl} W^o_{jk} x_l \text{sign}(x_k)
\end{aligned} \tag{B4}$$

where $\tilde{j}$ indicates the correct class for the considered input, and β is the positive quantity equal to the correct desired output value $\tilde{y}_{\tilde{j}}$ (in main text β=1). Eq. B4 contains two

clearly interpretable factors:

$$\Delta\theta^{(1)} = \sum_{j=1}^{N_{class}} \sum_{l=1}^{N} W^o_{jl} W^o_{jk} x_l \text{sign}(x_k) \tag{B5}$$

$$\Delta\theta^{(2)} = -\beta W^o_{\tilde{j}k} \text{sign}(x_k) \tag{B6}$$

Gradient on θ, cross entropy

The error function has the form

$$E = -\Big[\sum_j \tilde{y}_j \log\Big(\sigma(y_j)\Big) + (1-\tilde{y}_j)\log\Big(1-\sigma(y_j)\Big)\Big]$$

In this case, the learning rule for the thresholds is

$$\begin{aligned}
\Delta\theta_k = &-\eta_\theta \sum_j \tilde{y}_j (1-\sigma(y_j)) W^o_{jk} \text{sign}(x_k) + \\
&-\eta_\theta \sum_j (1-\tilde{y}_j)\sigma(y_j) W^o_{jk} \text{sign}(x_k) = \\
&= -\eta_\theta \sum_j \tilde{y}_j W^o_{jk} \text{sign}(x_k) + \\
&+\eta_\theta \sum_j y_j \sigma(y_j) W^o_{jk} \text{sign}(x_k)
\end{aligned} \tag{B7}$$

The two terms in Eq. B7 have comparable meaning to $\Delta\theta^{(2)}$ and $\Delta\theta^{(1)}$ of Eq. B5 and B6 computed for the mean square error, To demonstrate this, we can consider the case of a classification task where $y_j^{true} = 1$ for the correct class and zero otherwise. Furthermore, considering that the neural network output is not in the saturating regime of the sigmoid function when the majority of the learning happens, we can use the dominant first term

of the Taylor series of the sigmoid and approximate the second term of Eq. B7

$$\begin{aligned}
\Delta\theta_k &= -\eta_\theta W^o_{\tilde{j}k}\text{sign}(x_k)+ \\
&+\eta_\theta \sum_j \sigma(y_j) W^o_{jk}\text{sign}(x_k) = \\
&= -\eta_\theta W^o_{\tilde{j}k}\text{sign}(x_k)+ \\
&+\eta_\theta \sum_j [\frac{1}{2} + \frac{1}{4}y_j + ...] W^o_{jk}\text{sign}(x_k) = \\
&= -\frac{1}{2}\eta_\theta W^o_{\tilde{j}k}\text{sign}(x_k) + \frac{1}{2}\eta_\theta \sum_{j\neq\tilde{j}} W^o_{jk}\text{sign}(x_k) \\
&+\eta_\theta \sum_j [\frac{1}{4}y_j + ...] W^o_{jk}\text{sign}(x_k)
\end{aligned} \tag{B8}$$

where we can define

$$\Delta\theta^{(2)} = -\frac{1}{2}\eta_\theta W^o_{\tilde{j}k}\text{sign}(x_k) + \frac{1}{2}\eta_\theta \sum_{j\neq\tilde{j}} W^o_{jk}\text{sign}(x_k) \tag{B9}$$

$$\Delta\theta^{(1)} = \eta_\theta \sum_j [\frac{1}{4}y_j + ...] W^o_{jk}\text{sign}(x_k) \tag{B10}$$

Eq. B9 has an analogous meaning to Eq. B6 with the additional term $\sum_{j\neq\tilde{j}} W^o_{jk}\text{sign}(x_k)$ that increases (decreases) thresholds of nodes that are helping (impeding) the wrong classification process. Considering only the linear term of the Taylor expansion of the sigmoid function, Eq. B10 has the exact same form as Eq. B5. To demonstrate its role as balancing term that deactivates nodes that are helping in a similar way the classification process (their contribution has the same sign of y_j), we estimated Eq. B10 by subtracting $\Delta\theta^{(2)}$ to the value of the gradient. The result of this procedure is shown in Panel C of Fig. 7 (Appendix).

Finally, the sparsity level reached after optimisation depends on the values of the learning rates η_W and η_θ used. Indeed, threshold learning can act as a balancing force

that, in the case of high learning rates, decreases the percentage of active nodes in the network and stabilizes the training phase. In this regard, Fig. 8 (Appendix) shows the accuracy and the corresponding sparsity level after optimisation as η_W and η_θ vary for the MNIST classification task faced in Section 3.2 where the input is given to the network column by column (as a 28-th dimensional sequence of 28 time steps).

Initialisation and threshold adaptation in sequential learning

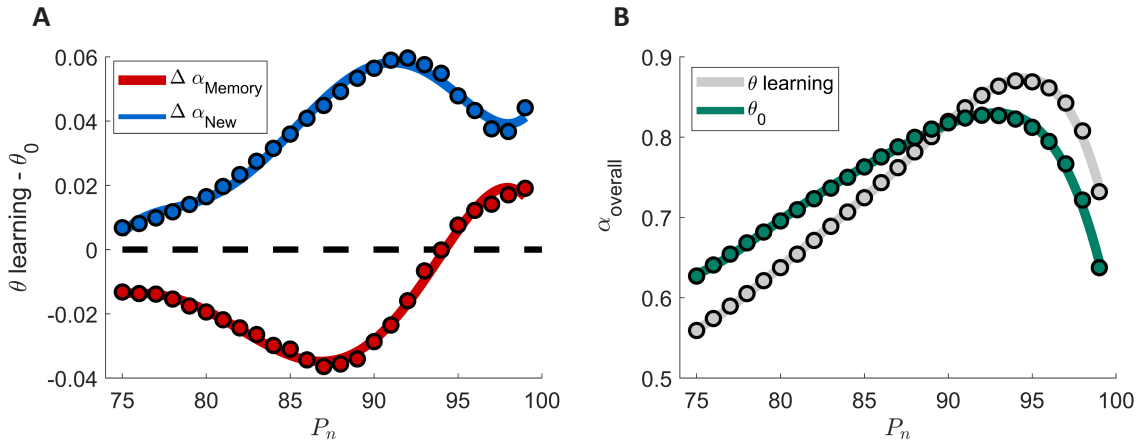

Figure 9: The initialisation procedure is the key element to alleviate catastrophic forgetting. **A**: Difference between the full SpaRCe model and a network where the thresholds are exclusively initialised for two metrics (Eq. B11 and B12), which measure the ability of the network to remember (red) and to learn novel tasks (blue) respectively. **B**: Overall accuracy with and without thresholds adaptation, which leads to an increase in the maximum performance of approximately five percent. The high accuracy reached exclusively with the initialisation demonstrates its importance to alleviate catastrophic forgetting.

We used Eq. 10, 11 and 12 introduced in the Main Text to calculate the differences

$$\Delta\alpha_{Memory} = \alpha_{Memory} - \alpha_{Memory,\theta_0} \tag{B11}$$

$$\Delta\alpha_{New} = \alpha_{New} - \alpha_{New,\theta_0} \tag{B12}$$

between a model where thresholds are initialised and then learned and a model where the thresholds are initialised only (as marked by the subscript θ_0). Up to high sparsity levels, threshold learning reduces the model's ability to remember previous tasks by changing

and compromising the parameters optimised on past datasets (seen as $\Delta\alpha_{Memory} < 0$ in Fig. 9A, Appendix). However, at initial sparsity above $P_n = 94$, the initial memory capacity of the network is reduced by the small number of active nodes; threshold learning helps to recruit inactive neurons and to increase the amount of resources available. This leads to a consequent increase of α_{Memory} for high sparsity levels. At the same time, threshold optimisation facilitates the learning of new datasets by adapting the parameters to the new incoming stimuli ($\Delta\alpha_{New} > 0$, Fig. 9A, Appendix, blue trend). Altogether, the trade-off between memory and adaptability on novel data results in the trend seen in Fig. 9B (Appendix): at lower initial sparsity levels, threshold learning worsens overall performance (because it reduces α_{Memory}), while at higher initial sparsity levels, threshold learning improves overall performance (because high sparsity prevents threshold learning from worsening α_{Memory} while still allowing it to improve α_{New}). In summary, threshold learning modulates the starting condition and increases the maximum performance ($\alpha_{overall}$) by about five percent.

Alternatives to threshold initialisation

In the main text, thresholds are initialised by computing percentiles over the ESN activities across a training dataset. Thus, data needs to be available before learning and the initialisation technique seems to be restricted to applications where a training dataset is known beforehand, apparently limiting the application of the proposed model. In this section we will discuss two alternatives to the procedure adopted in the main text, showing how SpaRCe initialisation criterion can be easily generalised to cases where a training dataset is not available a priori.

- Use of mini-batch statistics. Taking inspiration from batch normalisation, it is possible to compute $P_n(|V|)$ using the mini-batch data sampled at each training iteration. Of course, the overall computational time of the model will increase considerably, since the percentile operation needs to be computed every training step. For the

testing phase of the algorithm, a better estimate of the percentile over a larger sample can be computed through Eq. (B13), where the suffix l of $P_{n,m}(|V|_i)$ indicates the training step number l, while n and i refer to the percentile and node numbers as usual.

- Accumulation of data. An efficient alternative to the proposed initialisation is to accumulate M data samples and use Eq. B13 to set the percentile operation also for the training phase. Then, the update of the percentile estimate is stopped once its variation becomes negligible and the estimate has converged. While this method is computationally cheaper than the previous one and can be applied also to full online learning (where training is accomplished one sample at a time), it needs to accumulate statistics across a phase of training during which the dynamics of the network needs to be unchanged. Thus, the application of this methodology becomes more difficult when online adaptation of parameters that affect the ESN dynamic, as α or ρ [31] [35], is desirable.

$$P_{n,m}(|V_i|) = \frac{m-1}{m} P_{n,m-1}(|V_i|) + \frac{1}{m} P_{n,m}(|V_i|) \tag{B13}$$

Both these alternative methods reached comparable performance to the initialisation criterion adopted on the benchmarks of Section 3.2 of the main text.

APPENDIX C: Procedure for building sequences

Given an ensemble of elements $\mathcal{E} = \{A, B, C, ...\}$, we formulated a systematic procedure to build sequences of N_t (in the case analysed, $N_t = 3$) elements from $\mathcal{E}$. Our goal is to test the storage capacity of the model to learn associations between sequences and desired outputs. To achieve this, we prevented correlations between similar sequences from helping classification, by placing similar sequences in different classes. As described in Fig. 11A (Appendix), the procedure is based on the repetition of the following two steps:

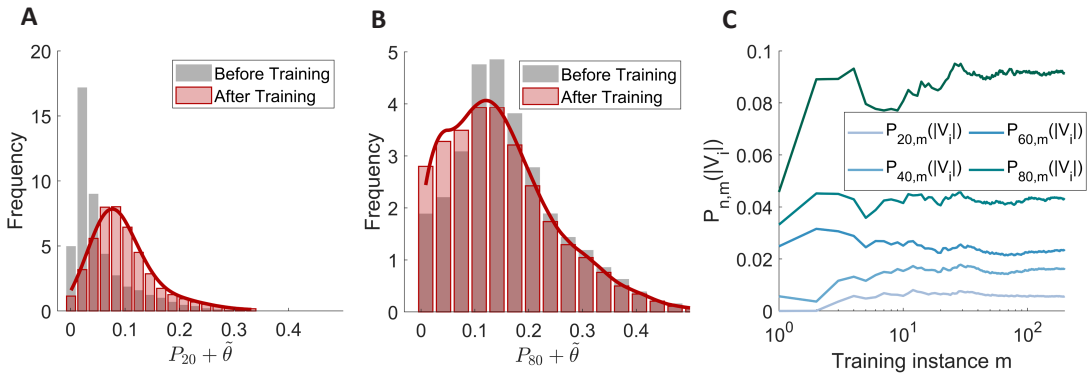

Figure 10: Example of threshold distributions before and after training. **A:** Distributions of thresholds when the initial percentile is twenty. It is evident how the learning process of $\tilde{\theta}$ increases the values of the thresholds on average. **B:** Distributions of thresholds when the initial percentile is eighty. The learning process decreases the values of the thresholds in this case. The results are in agreement with the optimal sparsity level for the task, that was about $25-40\%$ for the MNIST (see also Fig. 2 in the Appendix). **C:** Online estimates of percentiles through Eq. B13. The estimates converge and becomes accurate after 100 iterations, showing how a fully online computation of the percentiles is possible.

1. Given a number of output classes N_{class}, we picked $N_t N_{class}$ random elements with repetitions from $\mathcal{E}$ and composed the successions as in Fig. 11 (Appendix), where the simple case of two output classes is considered.

2. We picked other $N_t N_{class}$ random elements with no repetitions. Then, we changed the last temporal elements of the sequences generated before with N_{class} of the new picked stimuli, associating the new sequences to different desired output values as shown in the figure. Finally, we proceed in this way for all the previous temporal elements of the considered N_{class} successions from which we started (point 1).

We notice how the similarities between sequences can not be used to infer the right classification, since correlations among elements are associated to different output classes. Each element is associated to a multidimensional signal that, with addition of multiplicative white noise, is presented to the network (Fig. 11 B, Appendix).

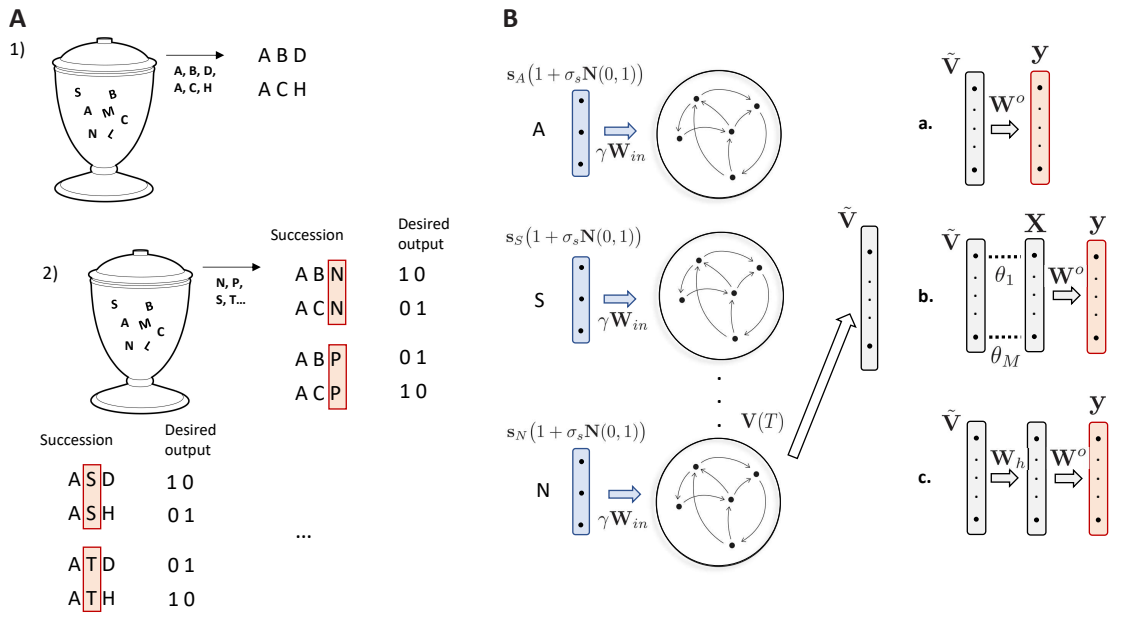

Figure 11: Task diagrams. **A:** Schematic of the procedure to define sequences. We notice how similarities among sequences are associated to different desired outputs, making the task a test of the storage capacity of the network. **B:** Schematic of the task on the biological data. Input example, succession of three stimuli (A,S,N) of time duration $\Delta t = 0.1s$ each. Multiplicative white noise $\sigma_s \mathbf{N}(0,1)$ is added throughout all the sequence, with different realisations of $\mathbf{N}(0,1)$ (gaussian with zero mean and unitary variance) for each temporal step. Only the last activities of the nodes in the reservoir is used for the read-out, which is divided into three models (**a**, standard ESN, **b**, SpaRCe, **c**, additional hidden layer) of increasing complexity and whose performance are tested and reported in the main text.

5.1 APPENDIX D: Mathematical Insights

Necessity of neuronal specialisation for obtaining an optimal solution with positive output weights

The following section contains a formal proof of how sparsity is necessary to find an optimal solution of a classification problem when the output weights are constrained to be positive. The cost function E in a classification task and for the entire dataset can be written as:

$$E = \sum_k \sum_{\mu \varepsilon \mathcal{C}_k} \sum_j^{N_{class}} \left[\tilde{y}_j^{\mu_k} - y_j^{\mu_k}\right]^2 \tag{D1}$$

where k indicates the correct class and the expression $\mu \; \varepsilon \; \mathcal{C}_k$ can be read as all datapoints μ that belongs to the cluster $\mathcal{C}_k$ from which we desire output k. We assume that the target

is 1 for the correct class and 0 otherwise. An ideal solution of this problem with an online learning learning algorithm can be defined as:

$$\forall\, \varepsilon > 0,\ \exists\, n \mid E_n \leq \varepsilon \tag{D2}$$

where n is the minibatch (or batch) number and it is a measure of the training time. Thus, Eq. D2 means that the cost function can be made as small as desired by going through enough training instances. The fact that the sum of the quadratic terms of the cost function has to be less than or equal to a desired ϵ value means that

$$\begin{aligned} [\tilde{y}_j^{\mu_k} - y_j^{\mu_k}]^2 \leq \varepsilon,\ \forall\mu,\ \forall k \\ \rightarrow -\sqrt{\varepsilon} \leq \tilde{y}_j^{\mu_k} - y_j^{\mu_k} \leq \sqrt{\varepsilon},\ \forall\mu,\ \forall k \end{aligned}$$

Thus, for each datapoint μ_k and each output class j

$$\begin{cases} 1 - \sqrt{\varepsilon} \leq y_j^{\mu_k} \leq 1 + \sqrt{\epsilon}, & \text{if } j = k \\ -\sqrt{\varepsilon} \leq y_j^{\mu_k} \leq \sqrt{\epsilon} & \text{if } j \neq k \end{cases}$$

which, by considering separately all different classes, becomes

$$\begin{cases} 1 - \sqrt{\varepsilon} \leq y_1^{\mu_1} \leq 1 + \sqrt{\epsilon}, & \\ -\sqrt{\varepsilon} \leq y_1^{\mu_k} \leq \sqrt{\epsilon} & \text{if } k \neq 1 \\ 1 - \sqrt{\varepsilon} \leq y_2^{\mu_2} \leq 1 + \sqrt{\epsilon}, & \\ -\sqrt{\varepsilon} \leq y_2^{\mu_k} \leq \sqrt{\epsilon} & \text{if } k \neq 2 \\ \vdots & \\ 1 - \sqrt{\varepsilon} \leq y_C^{\mu_C} \leq 1 + \sqrt{\epsilon}, & \\ -\sqrt{\varepsilon} \leq y_C^{\mu_k} \leq \sqrt{\epsilon} & \text{if } k \neq C \end{cases} \tag{D3}$$

Trivially, the difficulty of the classification task lies in the requirement that the activity of the same output node has to be close to one for some datapoints and close to zero for others; this condition is highlighted explicitly by the above set of inequalities, which can be rewritten

$$\begin{cases} 1-\sqrt{\varepsilon} \leq \sum_l W^o_{1l}\mathrm{ReLU}(V_l^{\mu_1}-\theta_l) \leq 1+\sqrt{\epsilon}, \\ -\sqrt{\varepsilon} \leq \sum_l W^o_{1l}\mathrm{ReLU}(V_l^{\mu_k}-\theta_l) \leq \sqrt{\epsilon}, \text{ if } k \neq 1 \\ \vdots \end{cases}$$

Since we required positive output weights we find

$$-\sqrt{\varepsilon} \leq \sum_l W^o_{1l}\mathrm{ReLU}(V_l^{\mu_k}-\theta_l) \leq \sqrt{\varepsilon} \tag{D4}$$

$$\rightarrow W^o_{1q} \approx \mathcal{O}(\sqrt{\varepsilon}),\ \forall q,k \mid V_q^{\mu_k} > \theta_q \tag{D5}$$

$$1-\sqrt{\varepsilon} \leq \sum_l W^o_{1l}\mathrm{ReLU}(V_l^{\mu_1}-\theta_l) \leq 1+\sqrt{\varepsilon} \tag{D6}$$

$$\rightarrow W^o_{1m} \gg \mathcal{O}(\sqrt{\varepsilon}),\ \forall m \mid V_m^{\mu_1} > \theta_m \tag{D7}$$

the conditions above can be satisfied only when the indexes q do not completely overlap with the indexes m, i.e. the representations do not totally overlap and a vector of thresholds is introduced to separate the ensemble of nodes that are active for different classes. We note that this simple proof holds in the case of positive output weights and positive reservoir activities only.

Interpretation of the optimal fifty percent of active nodes for memory capacity

The aim of the task faced in section 2.2 (Main text) is to measure the memory capacity of the model and the stability of the solution found by the model. The results show a robust optimal level of sparsity of 0.5 despite the specific values of noise level and number of output classes. Such level of sparsity maximises the probability that different representations have at least one node that is not in common. Given N nodes and an un-

defined input s_i, the ensemble of active nodes for that signal can be imagined as a random sample of $p \times N$ nodes from the total possible ensemble of neurons. Thus, each representation can be imagined as the extraction of $p \times N$ elements from an urn of N elements, where p is the imposed percentage of active nodes. In order to have representations that do not completely overlap we need to maximise the number of possible outcomes of the extraction, and this will guarantee that at least one node is different among the various representations. The number of possible extractions of active nodes corresponds to the number of combinations without repetitions:

$$\tilde{N} = \frac{N!}{(N - p \times N)!(p \times N)!} \tag{D8}$$

which has a maximum at $p = 0.5$. Thus, $p = 0.5$ is the sparsity level that maximises the probability that, given an undefined ensemble of input stimuli, the corresponding representations will have at least one non-overlapping neuron.

APPENDIX E: Hyperparameters

The parameters for the tasks faced are reported in the following three tables, respectively related to the storage capacity task of Section 3.1, the benchmarks that adopt the MNIST dataset of Section 3.2, and the application on catastrophic forgetting of Section 3.3 in the main text. The simulations were performed on a Windows PC with a gtx 1050Ti and Intel(R) Core(TM) i7-8750H CPU using Tensorflow on the GPU.

Storage capacity Task		
Hyperparameters		
σ		0.3
Δ_t		$0.1s$
T		$0.3s$
δt		$0.01s$
α		0.1
ρ		0.95
N		1000
γ		1
p_{ER}		0.001
η_W		4×10^{-2}
η_θ		4×10^{-3}
minibatch size		20
Computational Time		
	Initialisation	Training
ESN	0s	1233.98s
$SpaRCe$	0.3s	1234.31s

Table 3: The table reports the parameters defining the task, the hyperparameters of the ESN and the training hyperparameters for the storage capacity task. The computational time for the models is reported in terms of initialisation, computation of the percentile operation before learning, and training time, which corresponds to 3×10^5 iterations (minibatches) and ten evaluations of the performance across training.

MNIST based benchmarks			
MNIST/sMNIST		psMNIST	
α	0.17	α_2	0.017
ρ	0.97	ρ_2	0.99
N	1000	N_2	300|600|900
γ	0.1	γ_{21}	0.15
p_{ER}	0.01	p_{ER_2}	0.01
		α_1	1
		ρ_1	1
		N_1	200|400|600
		γ_1	1
		p_{ER_1}	0.01
η_W	1×10^{-5}	η_W	1×10^{-5}
η_θ	1×10^{-6}	η_θ	1×10^{-6}
minibatch	20	minibatch	20
Computational Time			
		Initialisation	Training
ESN		0s	474.51s
SpaRCe		34.84s	544.97s

Table 4: Table of the hyperparameters for the three benchmark tasks in Section 3.2. The hyperparameters for the psMNIST are double since two ESNs are used for this task (as the symbol ^{2} over SpaRCe highlights). The three values reported for the two network sizes correspond to the three different simulations performed as the total number of nodes (500,1000,1500) changes (Main text, Table 1). The suffix one corresponds to the first reservoir and the suffix two to the second reservoir. γ_{21} indicates the input gain of the adjacency matrix from the first to the second ESN. The computational time for the models is reported in terms of initialisation, computation of the percentile operation before learning, and training time, which corresponds to 1×10^5 iterations (minibatches) and ten evaluations of the performance across training for the MNIST task with $N = 1000$.

Catastrophic Forgetting			
Sequential classes		Sequential permutations	
α	0.17	α	0.17
ρ	0.97	ρ	0.97
N	1000	N	1000
γ	0.1	γ	0.1
p_{ER}	0.01	p_{ER}	0.01
η_W	2×10^{-6}	η_W	5×10^{-6}
η_θ	2×10^{-7}	η_θ	5×10^{-8}
minibatch	20	minibatch	20

Table 5: Table of parameters used in the catastrophic forgetting tasks. We note that the learning rates used for the thresholds is smaller than in the previous simulations, since we needed to avoid changing the sparsity level quickly, in order to keep the optimised sparsity level near the starting value imposed (P_n).

References

[1] Mikhail V Tsodyks and Mikhail V Feigel'man. The enhanced storage capacity in neural networks with low activity level. *EPL (Europhysics Letters)*, 6(2):101, 1988.

[2] MV Tsodyks. Associative memory in asymmetric diluted network with low level of activity. *EPL (Europhysics Letters)*, 7(3):203, 1988.

[3] Bernard Derrida, Elizabeth Gardner, and Anne Zippelius. An exactly solvable asymmetric neural network model. *EPL (Europhysics Letters)*, 4(2):167, 1987.

[4] Daniel J Amit, Hanoch Gutfreund, and Haim Sompolinsky. Storing infinite numbers of patterns in a spin-glass model of neural networks. *Physical Review Letters*, 55(14):1530, 1985.

[5] Sandro Romani, Itai Pinkoviezky, Alon Rubin, and Misha Tsodyks. Scaling laws of associative memory retrieval. *Neural computation*, 25(10):2523–2544, 2013.

[6] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC, 2015.

[7] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2074–2082. Curran Associates, Inc., 2016.

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[9] Peter M Rasmussen, Lars K Hansen, Kristoffer H Madsen, Nathan W Churchill, and Stephen C Strother. Model sparsity and brain pattern interpretation of classification models in neuroimaging. *Pattern Recognition*, 45(6):2085–2100, 2012.

[10] Edmund T Rolls and Martin J Tovee. Sparseness of the neuronal representation of stimuli in the primate temporal visual cortex. *Journal of neurophysiology*, 73(2):713–726, 1995.

[11] Kyle S Honegger, Robert AA Campbell, and Glenn C Turner. Cellular-resolution population imaging reveals robust sparse coding in the drosophila mushroom body. *Journal of Neuroscience*, 31(33):11772–11785, 2011.

[12] Andrew C Lin, Alexei M Bygrave, Alix De Calignon, Tzumin Lee, and Gero Miesenböck. Sparse, decorrelated odor coding in the mushroom body enhances learned odor discrimination. *Nature neuroscience*, 17(4):559, 2014.

[13] Glenn C Turner, Maxim Bazhenov, and Gilles Laurent. Olfactory representations by drosophila mushroom body neurons. *Journal of neurophysiology*, 99(2):734–746, 2008.

[14] Eyal Gruntman and Glenn C Turner. Integration of the olfactory code across dendritic claws of single mushroom body neurons. *Nature neuroscience*, 16(12):1821, 2013.

[15] Hao Li, Yiming Li, Zhengchang Lei, Kaiyu Wang, and Aike Guo. Transformation of odor selectivity from projection neurons to single mushroom body neurons mapped with dual-color calcium imaging. *Proceedings of the National Academy of Sciences*, 110(29):12084–12089, 2013.

[16] Javier Perez-Orive, Ofer Mazor, Glenn C Turner, Stijn Cassenaer, Rachel I Wilson, and Gilles Laurent. Oscillations and sparsening of odor representations in the mushroom body. *Science*, 297(5580):359–365, 2002.

[17] James M Jeanne and Rachel I Wilson. Convergence, divergence, and reconvergence in a feedforward network improves neural speed and accuracy. *Neuron*, 88(5):1014–1026, 2015.

[18] Rony Azouz and Charles M Gray. Dynamic spike threshold reveals a mechanism for synaptic coincidence detection in cortical neurons in vivo. *Proceedings of the National Academy of Sciences*, 97(14):8110–8115, 2000.

[19] Matthew S Grubb and Juan Burrone. Activity-dependent relocation of the axon initial segment fine-tunes neuronal excitability. *Nature*, 465(7301):1070, 2010.

[20] Nathan Intrator and Leon N Cooper. Objective function formulation of the bcm theory of visual cortical plasticity: Statistical connections, stability conditions. *Neural Networks*, 5(1):3–17, 1992.

[21] Jochen Triesch. A gradient rule for the plasticity of a neuron's intrinsic excitability. In Włodzisław Duch, Janusz Kacprzyk, Erkki Oja, and Sławomir Zadrożny, editors, *Artificial Neural Networks: Biological Inspirations – ICANN 2005*, pages 65–70, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[22] Mohd-Hanif Yusoff, Joseph Chrol-Cannon, and Yaochu Jin. Modeling neural plasticity in echo state networks for classification and regression. *Information Sciences*, 364-365:184–196, 2016.

[23] Jochen J. Steil. Online reservoir adaptation by intrinsic plasticity for backpropagationâdecorrelation and echo state learning. *Neural Networks*, 20(3):353–364, 2007. Echo State Networks and Liquid State Machines.

[24] Benjamin Schrauwen, Marion Wardermann, David Verstraeten, Jochen J. Steil, and Dirk Stroobandt. Improving reservoirs using intrinsic plasticity. *Neurocomputing*, 71(7):1159–1171, 2008. Progress in Modeling, Theory, and Application of Computational Intelligenc.

[25] Chao Du, Fuxi Cai, Mohammed A Zidan, Wen Ma, Seung Hwan Lee, and Wei D Lu. Reservoir computing using dynamic memristors for temporal information processing. *Nature communications*, 8(1):1–10, 2017.

[26] Manjari S Kulkarni and Christof Teuscher. Memristor-based reservoir computing. In *2012 IEEE/ACM international symposium on nanoscale architectures (NANOARCH)*, pages 226–232. IEEE, 2012.

[27] Xiaojian Zhu, Qiwen Wang, and Wei D Lu. Memristor networks for real-time neural activity analysis. *Nature communications*, 11(1):1–9, 2020.

[28] Kristof Vandoorne, Pauline Mechet, Thomas Van Vaerenbergh, Martin Fiers, Geert Morthier, David Verstraeten, Benjamin Schrauwen, Joni Dambre, and Peter Bienstman. Experimental demonstration of reservoir computing on a silicon photonics chip. *Nature communications*, 5(1):1–6, 2014.

[29] Yvan Paquot, Francois Duport, Antoneo Smerieri, Joni Dambre, Benjamin Schrauwen, Marc Haelterman, and Serge Massar. Optoelectronic reservoir computing. *Scientific reports*, 2:287, 2012.

[30] Kohei Nakajima. Physical reservoir computingâan introductory perspective. *Japanese Journal of Applied Physics*, 59(6):060501, 2020.

[31] Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural networks*, 20(3):335–352, 2007.

[32] Ali Rodan and Peter Tino. Minimum complexity echo state network. *IEEE transactions on neural networks*, 22(1):131–144, 2010.

[33] Hongyan Cui, Xiang Liu, and Lixiang Li. The architecture of dynamic reservoir in the echo state network. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(3):033127, 2012.

[34] Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli. Deep reservoir computing: A critical experimental analysis. *Neurocomputing*, 268:87–99, 2017.

[35] Luca Manneschi, Matthew O. A. Ellis, Guido Gigante, Andrew C. Lin, Paolo Del Giudice, and Eleni Vasilaki. Exploiting multiple timescales in hierarchical echo state networks. *Frontiers*, 2021.

[36] Filippo Maria Bianchi, Simone Scardapane, Sigurd Løkse, and Robert Jenssen. Reservoir computing approaches for representation and classification of multivariate time series. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[37] Junzhou Huang, Tong Zhang, and Dimitris Metaxas. Learning with structured sparsity. *Journal of Machine Learning Research*, 12(Nov):3371–3412, 2011.

[38] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted l 1 minimization. *Journal of Fourier analysis and applications*, 14(5-6):877–905, 2008.

[39] Mantas Lukoševičius. A practical guide to applying echo state networks. In *Neural networks: Tricks of the trade*, pages 659–686. Springer, 2012.

[40] Qianli Ma, Lifeng Shen, Weibiao Chen, Jiabin Wang, Jia Wei, and Zhiwen Yu. Functional echo state network for time series classification. *Information Sciences*, 373:1–20, 2016.

[41] Nils Schaetti, Michel Salomon, and Raphaël Couturier. Echo state networks-based reservoir computing for mnist handwritten digits recognition. In *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, pages 484–491. IEEE, 2016.

[42] Jacob Torrejon, Mathieu Riou, Flavio Abreu Araujo, Sumito Tsunegi, Guru Khalsa, Damien Querlioz, Paolo Bortolotti, Vincent Cros, Kay Yakushiji, Akio Fukushima, et al. Neuromorphic computing with nanoscale spintronic oscillators. *Nature*, 547(7664):428–431, 2017.

[43] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123, 2019.

[44] Sarath Chandar, Chinnadhurai Sankar, Eugene Vorontsov, Samira Ebrahimi Kahou, and Yoshua Bengio. Towards non-saturating recurrent units for modelling long-term dependencies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3280–3287, 2019.

[45] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler L Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[46] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *Proceedings of the 35th International Conference on Machine Learning*, 80:4548–4557, 10–15 Jul 2018.

[47] Elissa A Hallem and John R Carlson. Coding of odors by a receptor repertoire. *Cell*, 125(1):143–160, 2006.

[48] Shawn R Olsen, Vikas Bhandawat, and Rachel I Wilson. Divisive normalization in olfactory population codes. *Neuron*, 66(2):287–299, 2010.

[49] Sean X Luo, Richard Axel, and LF Abbott. Generating sparse and selective third-order responses in the olfactory system of the fly. *Proceedings of the National Academy of Sciences*, 107(23):10713–10718, 2010.

[50] Moshe Parnas, Andrew C Lin, Wolf Huetteroth, and Gero Miesenböck. Odor discrimination in drosophila: from neural population codes to behavior. *Neuron*, 79(5):932–944, 2013.

[51] Kamesh Krishnamurthy, Ann M Hermundstad, Thierry Mora, Aleksandra M Walczak, and Vijay Balasubramanian. Disorder and the neural representation of complex odors: smelling in the real world. *arXiv preprint arXiv:1707.01962*, 2017.

[52] Sophie JC Caron, Vanessa Ruta, LF Abbott, and Richard Axel. Random convergence of olfactory inputs in the drosophila mushroom body. *Nature*, 497(7447):113, 2013.

[53] Sen Song, Per Jesper Sjöström, Markus Reigl, Sacha Nelson, and Dmitri B Chklovskii. Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLoS biology*, 3(3):e68, March 2005.

[54] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[55] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[56] Gaurav Arora, Afshin Rahimi, and Timothy Baldwin. Does an lstm forget more than a cnn? an empirical study of catastrophic forgetting in nlp. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association*, pages 77–86, 2019.

[57] John B Butcher, David Verstraeten, Benjamin Schrauwen, Charles R Day, and Peter W Haycock. Reservoir computing and extreme learning machines for non-linear time-series data analysis. *Neural networks*, 38:76–89, 2013.

[58] Levy Boccato, Amauri Lopes, Romis Attux, and Fernando J Von Zuben. An extended echo state network using volterra filtering and principal component analysis. *Neural Networks*, 32:292–302, 2012.

[59] Michiel Hermans and Benjamin Schrauwen. Recurrent kernel machines: Computing with infinite echo state networks. *Neural Computation*, 24(1):104–133, 2012.

[60] Jonathan Dong, Ruben Ohana, Mushegh Rafayelyan, and Florent Krzakala. Reservoir computing meets recurrent kernels and structured transforms. *Advances in Neural Information Processing Systems*, 33, 2020.

[61] Danijela Marković, Nathan Leroux, Mathieu Riou, Flavio Abreu Araujo, Jacob Torrejon, Damien Querlioz, Akio Fukushima, Shinji Yuasa, Juan Trastoy, Paolo Bortolotti, et al. Reservoir computing with the frequency, phase, and amplitude of spin-torque nano-oscillators. *Applied Physics Letters*, 114(1):012409, 2019.

[62] Miguel Romera, Philippe Talatchian, Sumito Tsunegi, Flavio Abreu Araujo, Vincent Cros, Paolo Bortolotti, Juan Trastoy, Kay Yakushiji, Akio Fukushima, Hitoshi Kubota, et al. Vowel recognition with four coupled spin-torque nano-oscillators. *Nature*, 563(7730):230, 2018.

2.3 Paper III

The aim of this work is to define a decision-making model from abstract, desirable principles and to demonstrate its improved behaviour in terms of robustness, performance and agreement with neuroscientific findings in comparison to alternative model published in the literature [17]. In this research, we found that three key concepts were fundamental for the achievement of the proposed goal. First, the model should exploit the concept of multiple timescales, since diverse environments and tasks require accumulation of information at different speeds. Thus, the concept of multiple timescales is, as in the first papers, an important principle of the work, now encoded explicitly through the characteristic times of multiple integrators. Second, the model should have awareness of the passage of time and exploit an internal clock mechanism. The ability to estimate temporal lengths is an important requirement when actions needs to be timed and decisions must be taken in a given temporal window. For instance, classical musicians need to measure the time before the start of a specific part, and must be able to play at a high level of synchronism. Finally, adaptation of the parameters of the model should be driven by reward maximisation. This feature is necessary from a theoretically optimal perspective, where the simulated agent seeks to maximise a notion of the reward delivered by the environment. The model formulated is simple, describable as an actor-critic reinforcement learning agent parametrised by the read-out of a reservoir of unconnected, leaky integrators with different characteristic times. Yet, its behaviour is rich and capable of capturing different features of established experimental results. The model would also show improved robustness to environmental changes in comparison to other, more "standard", alternatives where accumulation of information occurs on a single and fixed timescale. The presence of a maximum allowed time to make a decision and the knowledge of the passage of time will lead to the discovery of a non-constant decision threshold which is in agreement with the mathematical findings in [18].

The paper is under correction after positive review on PLOS Computational Biology.
My contributions to the work were: model development, coding, testing, writing of the paper. The author GG equally contributed to the writing of this manuscript, improving extensively the first version that I personally wrote. GG also formulated the analytical argument to support the necessity of multiple time constants.

Signal neutrality, scalar property, and collapsing boundaries as consequences of a learned multi-time scale strategy

Luca Manneschi [1,+]***, Guido Gigante*** [2,+]***, Eleni Vasilaki*** [1] ***,Paolo Del Giudice*** [2,*]

[1] Department of Computer Science, The University of Sheffield, Sheffield, United Kingdom,
[2] National Center for Radiation Protection and Computational Physics, Italian Institute of Health, Rome, Italy,
[+] Joint first authorship
[*] Deceased

Abstract

We postulate that three fundamental elements underlie a decision making process: perception of time passing, information processing in multiple timescales and reward maximisation. We build a simple reinforcement learning agent upon these principles that we train on a random dot-like task. Our results, similar to the experimental data, demonstrate three emerging signatures. (1) Signal neutrality: insensitivity to the signal coherence in the interval preceding the decision. (2) Scalar property: the mean of the response times varies widely for different signal coherences, yet the shape of the distributions stays almost unchanged. (3) Collapsing boundaries: the "effective" decision-making boundary changes over time in a manner reminiscent of the theoretical optimal. Removing the perception of time or the multiple timescales from the model does not preserve the distinguishing signatures. Our results suggest an alternative explanation for signal neutrality. We propose that it is not part of motor planning. It is part of the decision-making process and emerges from information processing on multiple timescales.

Author summary

Humans and animals integrate sensory information before making a decision. The integration rate varies depending on the task. While driving could require quick reactions, evaluating the authenticity of a painting typically requires long observations. Consequently, the concept of representations created over multiple timescales appears necessary. Nevertheless, there is a lack of theoretical research that exploits multiple timescales, despite experimental evidence for the variety of integration rates. We, therefore, developed a decision-making model based on simple integrators with multiple characteristic times. We analysed its behaviour on a highly volatile, biologically relevant task. Through reward maximisation based on trial and error, the model discovers an effective strategy that is surprisingly different and more robust than the "classical" single timescale approach. This

learned strategy exhibits a remarkable agreement with experimental findings, suggesting a fundamental role of multiple timescales for decision-making. Our abstract model achieves a degree of biological realism while performing robustly in different environments.

1 Introduction

Perceptual decision-making is one of the most fundamental interactions of a biological agent with its environment. Perceptual decision-making processes have been long studied in the context of operant conditioning [1]. In these scenarios, an animal learns to associate choices and consequences by trial and error. Sub-optimal performance is considered a consequence of imperfect learning or the reflex of the learning strategy itself [2].

Outside this context, the research on perceptual decision-making has mainly focused on tasks where uncertainty (typically in the form of noisy signals) and time (e.g., duration of the observation and response delays) play a pivotal role [3–7]. In such scenarios, the errors made by the subject at the end of a training phase, as well as the relevant performance metrics (*e.g.* accuracy or speed of response), are deemed informative of the cognitive mechanisms involved [8–11]. There have been numerous attempts to compare the behaviour of animal subjects to the performance of different algorithms and determine how optimal the displayed behaviour is [8, 12–16].

One of the key ideas in perceptual decision-making is accumulating evidence over time [6, 8, 17–20]. The drift-diffusion model (also known as the 'bounded evidence accumulation' model) consists of two or more competing traces. These traces accumulate sensory evidence for different choices; the first trace to hit a threshold makes the associated option the final decision [21]. The drift-diffusion model is a continuous-time variant of the sequential probability ratio test [22, 23]. In the case of two-alternative forced choices, it is optimal in selecting between two hypotheses. Despite its simplicity, this model accounts for many psychophysical and neural observations. Examples are the distribution of response times and performance when varying sensory coherence [22, 23].

Notwithstanding its success, there are several alternatives to the standard drift-diffusion model [8, 24, 25] to account for unexplained phenomena such as primacy and recency effects, asymptotic accuracy, and "fast errors" [26–28]. Of notable importance is the Ornstein–Uhlenbeck model, which modifies the standard drift-diffusion model by including a decay term in the dynamics of the accumulation. Although the Ornstein–Uhlenbeck model can account for many experimental observations, including neurophysiological ones [24, 28], it introduces a characteristic timescale over which the model 'forgets' the past sensory information. A common approach in the literature is to treat the timescale of the accumulation as a free parameter that is optimised to match experimental data [28, 29].

Here we take a different approach. We study a decision-making problem within the context of reinforcement learning. The task is is intended to mimic a typical perceptual decision making setup [30]: an actor-critic agent has the task of determining whether a noisy signal has a positive or negative mean value. This agent can also decide when to decide, i.e., it can choose to wait instead of making a decision. We, thereby, postulate that the concept of reward maximisation is inherent in such problems.

Whilst not theoretically impossible, it is not straightforward to devise a biologically plausible mechanism to tune a single timescale parameter to the statistics of a task. To circumvent this issue, we propose a more biologically plausible process. The agent receives the signal from multiple integrators, each with a different time constant. Via reinforcement learning, the agent learns how to weigh them appropriately to maximise the collected reward. We hypothesise that multiple timescales lead to robust performance across different tasks since it is unrealistic to expect one time constant to fit any problem. In the context of our model, we will explore robustness when varying the task difficulty, i.e. the signal to noise ratio, and contrast it with models of one time constant.

Beyond the computational advantage, such approach is consistent with the ample evidence of the coexistence of many timescales in brain functionality [31–35], even at the single neuron level [36–38], and for reward memory in reinforcement learning [39].

Another fundamental element of our model is that the agent perceives the passage of

time. The agent has a "clock" available, several integrators with various time constants that increase by a fixed amount at each time step. In our model, we pair the clock's time constants with the time constants of the signal integrators. We do this to facilitate our mathematical analysis. However, we expect multiple time constants in the clock to implement a scalable population code for time, akin to what experimentally observed [40]. And, more specifically, to allow for more complex decision-making boundaries. We contrast an agent without any clock mechanism, an agent with a "single time constant" clock, and an agent with a multiple timescales clock. Our results highlight the performance advantages that a multiple timescales clock brings in.

We evaluate our agent concerning three properties observed in experimental data or theoretical analyses of decision-making processes. (1) Signal neutrality. We use this term as a shorthand to denote the observation that, for several hundred of milliseconds before the decision, the neurons in the lateral intraparietal cortex that correlate with the decision show the same response to different signal-to-noise ratios, with a time course of the firing activity that is indistinguishable in the different cases [5,41]. One prior explanation is that the signals in that stage prepare the motor action. Here we evaluate this behaviour as part of the decision making process. (2) Scalar property or Weber's law [42]. The coefficient of variation (CV, the ratio of the standard deviation to the mean) remains constant as the task difficulty varies. (3) Collapsing boundaries. In the beginning, the agent should wait to integrate information to make an informed decision. However, the decision time is not unlimited; as time passes, the decision boundaries decrease to force the agent to act.

Our setup has similarities to a Partially Observable Markov Decision Process [43] with opportunity costs. The agent cannot access its state consisting of the signal sign and the clock. Instead, it has access to several observations at each time step. These observations are continuous variables that integrate noisy information about the state in terms of signal information and the time passed. These observations progressively correlate with the agent's "true" state as the integration filters out the noise. The option to defer this decision in case of insufficient evidence complements the desirable action to find the sign of the

stimulus. Yet, the presence of a time limit effectively imposes a cost on deferring the decision to accumulate more evidence.

2 Methods

2.1 Task definition

Inspired by classical random dots experiments [30], we model a two-alternative forced-choice task as a decision over the sign of the mean value of a noisy signal $\mathrm{s}(t)$ (see Fig 1). The signal (black line) consists of independent samples from a Gaussian distribution of mean μ and standard deviation σ, each drawn every time step $\Delta t = 10$ ms.

The agent is not required to decide at a prescribed time, it has the option to wait and then see another sample, or to perform one of two actions, 'left' and 'right', respectively associated with the decision $\mu < 0$ and $\mu > 0$ at each step. When an action is made, the episode ends, and a reward is delivered only if the agent correctly guessed the sign of μ; otherwise, the agent receives nothing. Each episode has a maximum duration $T_{\max}$. When $T_{\max}$ is reached, another 'wait' from the agent leads to the end of the episode and no reward is delivered.

Whilst σ is constant, the value of μ is instead re-sampled at the beginning of each episode from a Gaussian distribution $p(\mu)$ of zero mean and variance σ_μ. This second-order uncertainty makes the agent experience a wide range of values of μ, putting severely to the test its ability to generalise to episodes of varying signal-to-noise ratios.

2.2 Relationship between μ and random dots coherence

In random dots experiments, usually a number of dots moves randomly on a screen, with a fraction of them moving instead coherently in one direction (either left or right in different episodes). The percentage of coherently moving dots ('coherence') is a measure of how difficult an episode is, not unlike $|\mu|$ in the model (with sign of μ corresponding to a coherent movement towards left or towards right respectively). To make the parallel

between the present task and the experimental settings more evident, in the following we will show results using either $|\mu|$ or the coherence of the signal, the two measures being related by:

$$|\mu| = 0.216 \frac{\text{coherence}}{\sqrt{100 - \text{coherence}}}. \tag{1}$$

In fact, in [5], every three frames on the screen, a fraction c ('coherence') of dots are moved coherently in the chosen direction by $\mathrm{d}x$, while the other $1 - c$ dots are randomly displaced. We assume that each of the randomly moving dots is subjected to a change Δx in their position following a probability distribution, with $\langle \Delta x \rangle = 0$ and $\mathrm{Var}[\Delta x] = \sigma_x^2$. Imagining that neurons with different receptive fields help to estimate the average movement of the dots at each time step, we end up with a signal s of mean:

$$\mu \equiv \langle s \rangle = c\,\mathrm{d}x \tag{2}$$

and variance:

$$\sigma^2 \equiv \mathrm{Var}[s] = (1 - c)\,\sigma_x^2 \tag{3}$$

Then, we have the relationship:

$$\frac{\mu}{\sigma} = \frac{c}{\sqrt{1 - c}} \frac{\mathrm{d}x}{\sigma_x} \tag{4}$$

or:

$$\mu \propto \frac{\text{coherence}}{\sqrt{100 - \text{coherence}}}, \tag{5}$$

where we have expressed the coherence as a percentage. Eq 1 is a special case of this one, with a proportionality constant chosen to match experimental ranges.

2.3 An agent over multiple timescales

The section is dedicated to the definition of the proposed model. In contrast to previous research works on the decision making process, the agent makes decisions thanks to a reservoir of multiple timescales of integration and an estimate of the passage of time.
The agent comprises $n_\tau = 10$ leaky integrators x^s_τ (dark blue to cyan lines in Fig 1) that independently integrate the noisy signal $s(t)$ over different timescales τ:

$$\dot{x}^s_\tau = -\frac{x^s_\tau - \mathrm{s}(t)}{\tau}, \tag{6}$$

and correspondingly n_τ leaky integrators x^c_τ (yellow to red lines in Fig 1) that integrate a constant input (a 'time signal', here valued 1), to account for the possible effects of an internal 'clock':

$$\dot{x}^c_\tau = -\frac{x^c_\tau - 1}{\tau}. \tag{7}$$

Both the x^s_τ and the x^c_τ are reset to 0 at the beginning of each episode (note, therefore, that $x^c_\tau(t) = 1 - \mathrm{e}^{-\frac{t}{\tau}} \geq 0$ for all t). Moreover, we added noise to the values of the integrators at a given time (Eqs 6 and 7) redefining:

$$x^s_\tau(t) \leftarrow x^s_\tau(t) + \xi^s_\tau \tag{8}$$

$$x^c_\tau(t) \leftarrow x^c_\tau(t) + \xi^c_\tau \tag{9}$$

$\xi^s_\tau(t)$ and $\xi^c_\tau(t)$ are drawn independently for each t and each τ from a Gaussian distribution with zero mean and standard deviation σ_I. The $\xi^s_\tau(t)$s and $\xi^c_\tau(t)$s are introduced to model the intrinsic noise implied in any plausible biological implementation of the integration process, such as fluctuations in the instantaneous firing rate of a network of neurons.

The τs are chosen on a logarithmic scale (*i.e.*, $\tau_i = \alpha\,\tau_{i-1}$, with α a suitable constant), with $\tau_1 = \tau_{\min} = 100$ ms and $\tau_{n_\tau} = \tau_{\max} = 10$ s, so as to allow the agent to accumulate information over a wide range of different timescales. The specific choice of the distribution

of timescales is not critical to the following results, assuming that the values of τs are densely spread over a wide range (see Results and Supplementary Material).

At each time step t, the agent computes six weighted sums, three for the signal $x^s_\tau(t)$ and three for the clock $x^c_\tau(t)$. The first four of these weighted sums are related to the two possible actions:

$$\Sigma^s_{\text{right}}(t) \equiv \sum_\tau \theta^s_{\text{right},\tau}\, x^s_\tau(t) \tag{10}$$

$$\Sigma^c_{\text{right}}(t) \equiv \sum_\tau \theta^c_{\text{right},\tau}\, x^c_\tau(t) + b_{\text{right}} \tag{11}$$

$$\Sigma^s_{\text{left}}(t) \equiv \sum_\tau \theta^s_{\text{left},\tau}\, x_\tau(t) \tag{12}$$

$$\Sigma^c_{\text{left}}(t) \equiv \sum_\tau \theta^c_{\text{left},\tau}\, x^c_\tau(t) + b_{\text{left}} \tag{13}$$

where b_{right} and b_{left} are constants and can be described as the propensity of the agent to make the corresponding actions before the beginning of an episode. The Σ^ss and the Σ^c carry information, respectively, on the signal and the time elapsed since the beginning of each episode. Even though the x^c_τ increase with time, the Σ^cs can be non-monotonic, something that will play an important in role in implementing an effective 'moving threshold' for the decision mechanism.

The other two sums are instead related to the 'wait' option:

$$\Sigma^s_{\text{wait}}(t) \equiv \sum_\tau \theta^s_{\text{wait},\tau}\, |x^s_\tau(t)| \tag{14}$$

$$\Sigma^c_{\text{wait}}(t) \equiv \sum_\tau \theta^c_{\text{wait},\tau}\, x^c_\tau(t) + b_{\text{wait}}, \tag{15}$$

where the absolute value in Eq 14 is taken to account for the intuition that a signal and its negative mirror should equally affect the agent's propensity to defer a decision. The constant b_{wait} has similar meaning to the biases b_{right} and b_{left}, but related to the 'wait' action. By setting:

$$\Sigma_x \equiv \Sigma^s_x + \Sigma^c_x \tag{16}$$

(with $x \in \{\text{left, right, wait}\}$), the six sums are then non-linearly combined through a softmax function (the circles corresponding to the actor on the right of Fig 1) to define a probability distribution over the possible actions:

$$p_{\text{right}}(t) = \frac{e^{\Sigma_{\text{right}}(t)}}{e^{\Sigma_{\text{left}}(t)} + e^{\Sigma_{\text{wait}}(t)} + e^{\Sigma_{\text{right}}(t)}} \tag{17}$$

and analogous expressions for 'left' and 'wait'. By definition, $p_{\text{left}}(t) + p_{\text{wait}}(t) + p_{\text{right}}(t) = 1$ for every t. The agent then randomly chooses an option according to the three probabilities.

The agent is thus completely determined by the choice of the six sets of n_τ weights: $\theta^s_{\text{left},\tau}$, $\theta^s_{\text{wait},\tau}$, $\theta^s_{\text{right},\tau}$, $\theta^c_{\text{left},\tau}$, $\theta^c_{\text{wait},\tau}$, $\theta^c_{\text{right},\tau}$, and three constant offsets b_{left}, b_{wait}, and b_{right}. We note how this set of parameters is redundant, because of the way they enter Eq 17. For example, we could make the substitution $b_{\text{right}} \leftarrow b_{\text{right}} - b_{\text{wait}}$, $b_{\text{left}} \leftarrow b_{\text{left}} - b_{\text{wait}}$, and $b_{\text{wait}} = 0$ and the resulting agent would be mathematically equivalent to the original one. We use such redundant definition in order to simplify the description of the model, making it the most symmetric for 'left', 'right', and 'wait'. These weights and offsets are learned by trial-and-error through a reinforcement learning procedure aiming to maximise reward. All the results shown, if not otherwise stated, are obtained using the same set of weights, at the end of the training procedure, with $T_{\max} = 2$ s, $\sigma = 0.18\,\text{s}^{-\frac{1}{2}}$, $\sigma_\mu = 0.25$, and $\sigma_I = 0.02$. Training of the parameters of the model is achieved through a standard actor-critic reinforcement learning algorithm [43], which is described in Supplementary Material. During learning, the model estimates at each step t the total future expected reward $\text{V}(t)$ for the current episode. Such estimate is computed by a linear summation of the integrators (Fig 1, bottom-right) and is used to establish a moving baseline to modulate the changes in the model's weights during training. The parameters of the actor and the critic are then updated thanks to the utilisation of eligibility traces [43].

2.4 Comparative models

To understand the role of multiple timescales and of the internal clock in the results, we compare the performance of the proposed agent with other decision making models.

1. *Single integrator with optimised threshold.* This refers to the Ornstein-Uhlenback decision process [24, 28], which is a generalisation of the standard drift diffusion model [21]. The model is composed by an integrator over one timescale and a threshold. The dynamic of the integrator is given by Eqs 6 and 8. A decision is triggered when the latter activity reaches ± a threshold value Θ_τ. In our case, the action 'right' is made when $x^s_\tau(t) \geq \Theta_\tau$, while the agent performs the 'left' action when $x^s_\tau(t) \leq -\Theta_\tau$. Considering the presence of a single timescale of integration, we will consider multiple versions of the process, each with a different value of τ. For each model with a specific τ, the threshold Θ_τ will be optimised through grid search by maximising the accuracy on the considered task. In this way, we are certain that the process will exhibit the highest possible performance on the considered task, or performance that are negligibly distant to its theoretical optimal.

2. *Agent with a single timescale.* The model refers to a reinforcement learning agent similar to the proposed one, but with only one timescale of integration. Practically, the agent definition is again based on Eqs 10– 14, but every summation over τ reduces to a single term. The total number of parameters in this case is thus nine (θ^s_{left}, θ^s_{wait}, θ^s_{right}, θ^c_{left}, θ^c_{wait}, θ^c_{right}, b_{left}, b_{wait}, and b_{right}). As for the single integrator with optimised threshold, we will simulate multiple versions of the model to vary the timescale of integration τ. We note how, in contrast to the previous comparative model, this process has an estimate of the passage of time over one single time constant. For this feature, the process departs from the other decision making models in the literature. This agent will help us to understand the role of multiple timescales further, providing a baseline where a basic knowledge of the internal clock is present, but where integration occurs over a single τ.

3. Agent with multiple signal integrators, but without internal clock. The model is again defined by Eqs 10– 14, but without temporal information, that is $\theta^c_{\text{left}} = \theta^c_{\text{wait}} = \theta^c_{\text{right}} \equiv 0$. The model will constitute an additional comparison to separate the roles of the availability of multiple timescales on the signal and on the internal clock mechanism.

Because of the presence of multiple integrators, the proposed agent effectively lowers the total noise by summing up n_τ integrators x^s_τ affected by independent sources of noise ξ^s_τ (Eq 8). Thus, when comparing the proposed agent with one of the above models that exploits a single time constant, we rescaled the amount of noise σ_I affecting the single integrator by a factor α_I, defined as

$$\alpha_I = \frac{1}{\sqrt{\sum_\tau \theta^2_{*,\tau} / \max_\tau(\theta^2_{*,\tau})}} \leq 1 \tag{18}$$

where $\theta_{*,\tau}$ refers to the optimal weights $\theta_{\text{right},\tau}$ found after training of the proposed model (we could equivalently use the optimal $\theta_{\text{left},\tau}$, since after training $\theta_{\text{left},\tau} \simeq -\theta_{\text{right},\tau}$ as it will be shown in Fig 10 D). Thus, $\alpha_I = 1$ when just one of the $\theta_{*,\tau}$ is different from 0, *i.e.* when the agent utilises just one integrator. On the other hand, the maximum $\alpha_I = \frac{1}{\sqrt{n_\tau}}$ is attained when the agent weights equally all the integrators. In this way, the total amount of noise in the single timescale model is effectively equivalent to the one present in the multiple timescales agent.

2.5 Signal neutrality and scalar property measures

To measure signal neutrality, we take the average $\Delta\Sigma_{\text{right}}(t)$ (see Eq 21), aligned to decision time, for six different coherences (0%, 3.2%, 6.4%, 12.8%, 25.6%, 51.2%); each curve is considered for an interval between 0 and 600 ms before the decision is taken; if the number of points to average for a given coherence drops below 100 before the 600 ms, the interval of definition of that curve is shrunk accordingly. We then rescale all the curves to fit inside

the range 0-1, so that the minimum of the minimum values attained by each curve is 0; and the maximum of the maxima is 1. Then we compute, for each time, the maximum distance between any pairs of rescaled curves (this distance is of course always ≤ 1 thanks to the rescaling). Finally we take the average of such maximum distance, and take the inverse: this is the operative measure of signal neutrality used throughout the paper.

To give a measure of scalar property, we compute the coefficient of variation CV for the distribution of response times corresponding to six values of coherence (0%, 3.2%, 6.4%, 12.8%, 25.6%, 51.2%). We then take the inverse of the difference between the maximum and the minimum value of CV: this is the reported measure of the scalar proprety (see Fig 7).

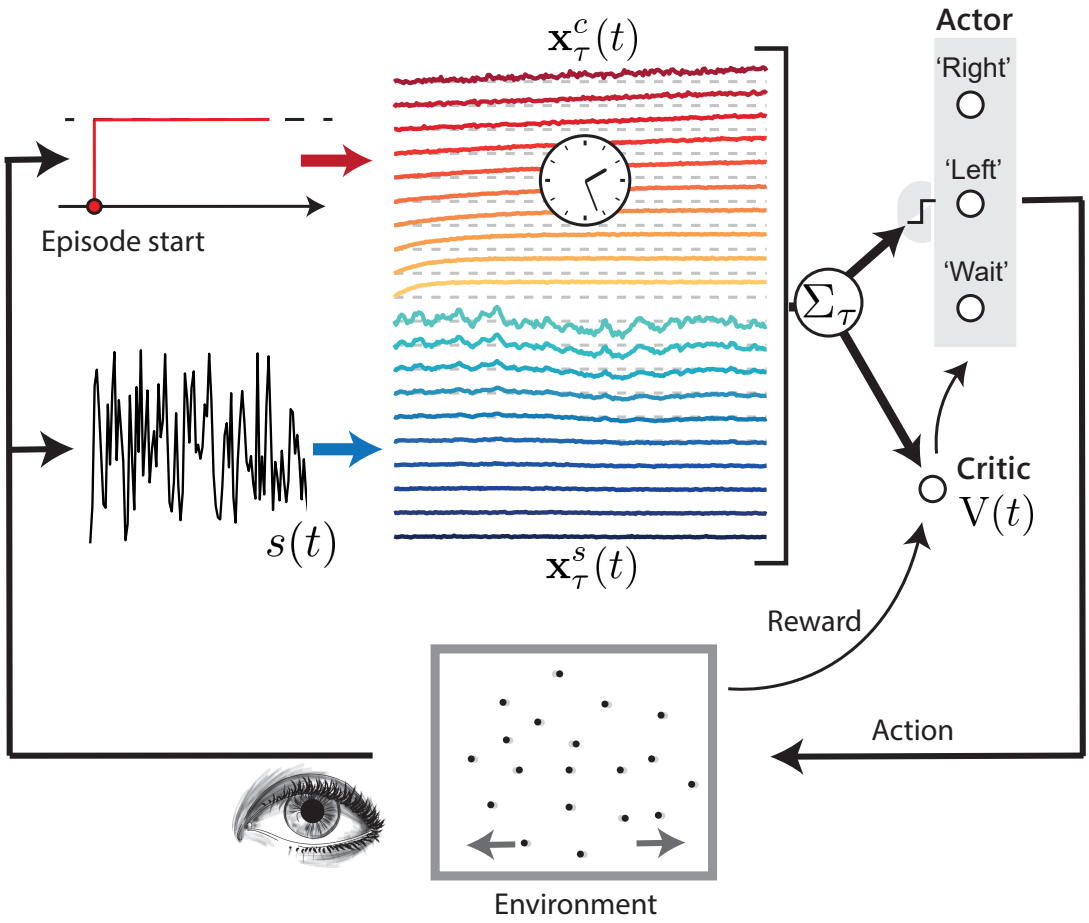

Figure 1. Task and model schematic. The environment corresponds to the random movement of a group of dots on a screen, which is represented as a uni-dimensional noisy signal $s(t)$ (black line), sampled at discrete time steps $\Delta t = 10$ ms from a Gaussian distribution of mean μ and variance σ^2. The task requires the subject to guess the sign of μ, by moving a lever to the right (positive sign) or to the left (negative sign); the subject can 'choose when to choose', within a maximum episode duration $T_{\max}$. The learning agent integrates the signal over different timescales τ ($x_\tau^s(t)$s, blue lines); the agent integrates a constant input (depicted in red as a constant from the start of the episode) over the same timescales ($x_\tau^c(t)$s, yellow-red lines) to simulate an internal clock mechanism estimating the passage of time. In both cases, the darker the colour the longer the corresponding timescale. At each time instance, the weighted sums of the integrators (far right) are fed into a decision layer (the actor) that computes the probability of choosing 'left' and 'right', thus terminating the episode, or to 'wait' to see another sample of $s(t)$. If the subject gives the correct answer (the guessed sign coincides with the actual sign of μ) within the time limit, a reward is delivered; otherwise, nothing happens. In any case, a new episode starts. The agent learns by observing the consequences (obtained rewards) of its actions, adapting the weights assigned to the $x_\tau^s(t)$s and $x_\tau^c(t)$s. During learning, the model estimates at each step t the total future expected reward $V(t)$ (the critic) for the current episode as a linear summation of the integrators. Learning of the parameters is accomplished through a standard actor-critic reinforcement learning model, where the reward delivered by the environment is used to update the V value function, which is then used to update the actor's parameters (see Supplementary Material for more details)

3 Results

First, we analyse how the behaviour of the optimised agent is different than the standard drift to diffusion model by exploiting integration over a variety of timescales. Fig 2 shows the evolution of $p_{\text{right}}(t)$ (blue line) and $p_{\text{left}}(t)$ (red) during an episode where the correct action is 'right' (that is, $\mu > 0$). As expected, $p_{\text{right}}(t)$ is for the most part greater than $p_{\text{left}}(t)$ (although this is unnoticeable in the plot where the probabilities are very small), signalling that the agent favours the action associated with the correct decision. Nevertheless, both probabilities are very low most of the time, implying that $p_{\text{wait}}(t)$ is often close to one (not shown). Thus, the agent appears to select a strategy in which decisions are made within short 'active' windows of time during which fleeting bursts of $p_{\text{left}}(t)$ or $p_{\text{right}}(t)$ make an action possible. Such strategy is not trivially associated with the intuitive picture of a process accumulating information over time until some threshold is met (for instance, see model 1. in section 2.4).

In fact, the agent exploits the information carried by the different integrators by waiting for their consensus, akin to a majority vote. A short-lived fluctuation in the fastest integrators would not be enough for a decision. Yet, in conjunction with a longer-lived fluctuation of the slower integrators, a burst in one of the actions is triggered. Such probability bursts are usually quite low (they often stay below a probability of 0.1) and therefore function as 'open windows' paving the way to a decision, more than as 'funnels' forcing it. Decisions therefore happen when the different timescales stay in agreement for an extended period (roughly 100 ms).

This is illustrated in Fig 2 with coloured circles, each row representing the evolution of one integrator (for a subset of 5 of the 10 integrators, with slow to fast timescales from top to bottom). As expected, inside a burst of $p_{\text{right}}(t)$ almost all the integrators present large positive values (dark blue, see for example temporal instance number 1 in Fig 2). On the other hand, integrators typically assume negative values (light to dark red) in correspondence of bursts of $p_{\text{left}}(t)$, as it is shown in the temporal instance number 2. The

converse is not true: in absence of probability bursts, not all the integrators assume low absolute values (see, for example, coloured circles corresponding to number 3). This is due to the fact that the integrators, though correlated, detect fluctuations in the signal over different timescales. Moreover, the non-linear nature of the probability function (Eq 17) dampens integrators' fluctuations falling below a given range of values. When a burst fades away (see for example points between 2 and 4) not all the integrators go down together. Initially the faster integrators become neutral or even slightly change sign. Afterwards the slower integrators follow suit. Of course, the process is not completely linear, and intermediate integrators can assume (see instance number 4 and neighbouring points) higher values, while the slowest (fastest) ones are still decreasing (fluctuating rapidly). A more detailed analysis of the behaviour of the agent can be found in Supplementary Material and Fig 11.

3.1 Model's performance

Fig 3A shows the fraction of correct choices as a function of the decision time, both for the agent at the end of training (black line) and for the optimal fixed-t observer (blue line) that, at each time t, simply chooses according to the sign of the sum of the signal up to time t. Its performance can be derived analytically:

$$\text{Fraction Correct}(t) = \frac{1}{2} + \frac{1}{\pi} \arctan \sqrt{\frac{\sigma_\mu^2 \, t}{\sigma^2}} \tag{19}$$

If the task were to decide exactly at time t, no other decision maker could outperform it; for this reason it is deemed optimal. The comparison with the fixed-t observer sheds light on the agent's strategy and the underlying trade-offs.

The agent is free to"choose when to choose", thus it is not surprising that its performance is higher than the optimal fixed-t observer for shorter decision times (the inset of Fig 3A shows the distribution of decision times for the agent). We see that the two performances cross slightly above the average decision time for the agent. Beyond this point, the fixed-t

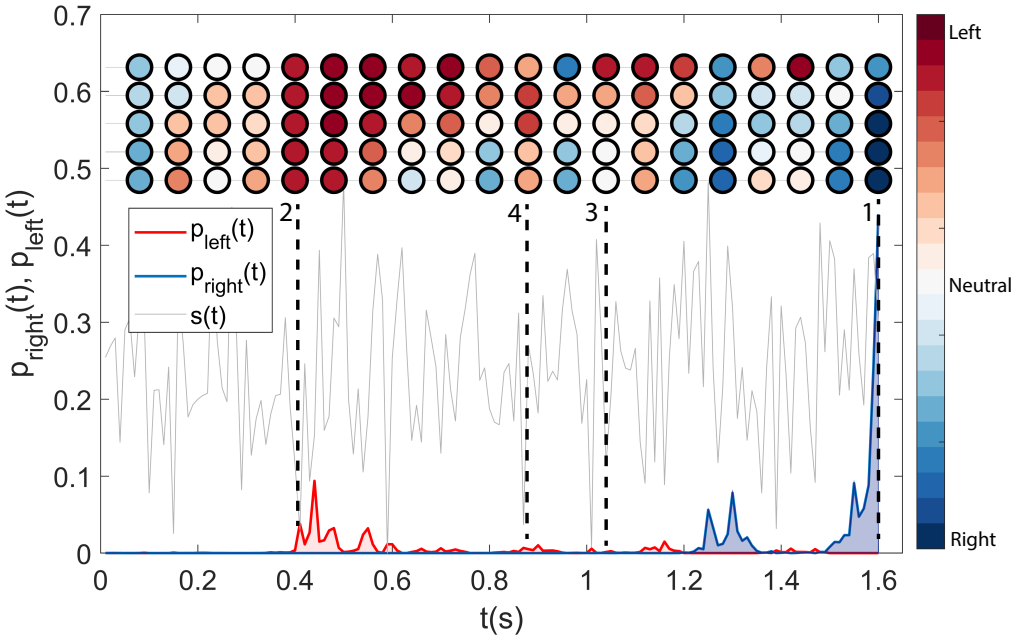

Figure 2. Learned decision strategy. Evolution of $p_{\text{right}}(t)$ (blue line) and $p_{\text{left}}(t)$ (red) during an episode (signal s(t) in dashed grey) where the correct action is 'right' (that is, $\mu > 0$). Decisions are made within short 'active' windows of time during which fleeting bursts of $p_{\text{left}}(t)$ or $p_{\text{right}}(t)$, corresponding to the alignment of many integrators, make an action possible. The coloured circles correspond to the values of a subset of 5 of the 10 integrators (slow to fast associated timescales from top to bottom). The colours (blue to red) represent the 'tendency' of an integrator toward a decision. Blues correspond to positive (toward the 'right' action) values, while reds to negative values (toward the 'left' action). These tendencies are computed using the average behaviour of the specific integrator as a reference value. In other words, if the circle is blue, it means that the value assumed by the integrator was higher than usual at that specific time. Uniformly positive (negative) values for the integrators are associated with bursts of p_{right} (p_{left}, see times denoted with 1 and 2 in the plot). Outside bursts (point 3) or when a burst withers (point 4), not all the integrators assume low absolute values.

observer dominates. Indeed, the agent can make the easy decisions early on and wait to see how the signal evolves when the choice appears more uncertain. In contrast, the fixed-t observer is bound to decide at time t, no matter how clear or ambiguous the observed signal was up to that point. The steep rise of the agent's performance for very short decision times is mainly a reflection of its ability to tell apart the easy episodes from the hard ones. The fixed-t observer catches up for longer times, where the agent is left with only the most difficult decisions and its performance consequently declines. For the fixed-t observer, instead, larger ts always mean more information and therefore its performance monotonically increases. We notice how at the crossing point, the agent has already made

the large part of its decisions, as it is apparent from the distribution of decision times.

Fig 3B shows how the agent (horizontal line) outperforms all the single integrators with optimised thresholds (circles, see section 2.4 for the model definition). The performance of the single-timescale integrator peaks for intermediate values of the associated timescale τ, though it always stays well below the performance attained by the agent. The agent, therefore, is able to leverage the information on multiple timescales from the signal and the internal clock to gain a clear performance advantage with respect to the drift-diffusion model on the whole spectrum of τs. A more detailed comparison between the performance of the different models considered will be given in section 3.5.

Fig 3C and 3D show the accuracy and the mean response time of the agent as the coherence of the signal varies (Eq 1). The black line in panel Fig 3C is computed as:

$$\text{Fraction Correct}(\text{coherence}) = 1 - \frac{1}{2}\exp\Big[-\big(\frac{\text{coherence}}{7.97}\big)^{1.62}\Big] \tag{20}$$

as in Fig 3 of [5], where the parameters of the curve were fitted to experimental data. The match between the experimental fit and the result of the agent is striking. In Fig 3D, instead, the black line is a generic sigmoidal function plotted for illustration purposes. As found in the experiments, the agent's responses become faster as the task becomes easier (larger coherences).

3.2 Signal neutrality

A more microscopic look at the decision process surprisingly uncovers shared features between the internal dynamics of the artificial agent and the activity observed in neurons in the lateral intraparietal cortex (LIP) during a random dots task [5, 41].

We now define a key observable of the model that will be central in the following (see Eqs 16, 10 and 11):

$$\Delta\Sigma_{\text{right}}(t) \equiv \Sigma_{\text{right}}(t) - \Sigma_{\text{wait}}(t) \tag{21}$$

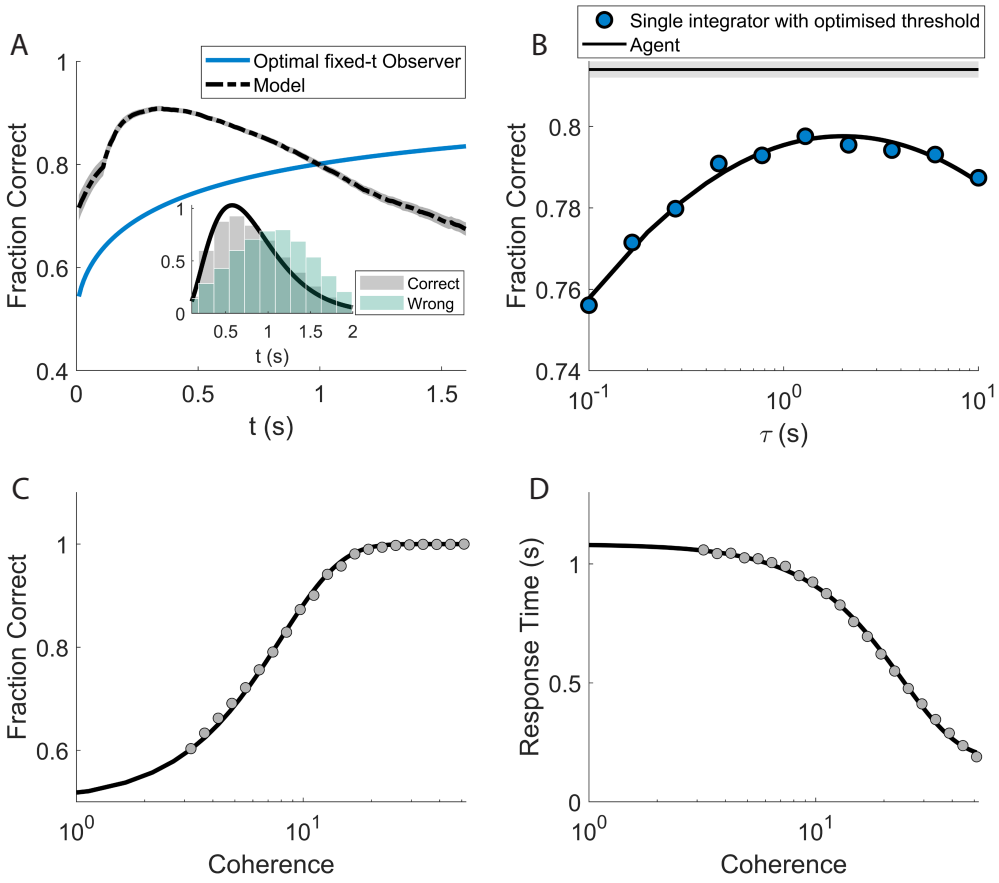

Figure 3. Performance after training. **A**: Fraction of correct choices as a function of the decision time, both for the agent at end of training (black line) and for optimal fixed-t observer (blue line) that simply chooses according to the sign of the accumulated signal up to time t (see text). The agent clearly outperforms the fixed-t observer for shorter decision times, thanks to its freedom to 'choose when to choose'. The steep rise of the agent's performance for very short decision times is mainly a reflection of its ability to tell apart the easy episodes from the hard ones. Inset: response time histograms for correct (grey) and wrong (green) decisions **B**: the agent (horizontal line) outperforms, considering the fraction of correct choices on a sample of episodes, all the single-timescale integrators with optimised decision threshold (dots; the continuous line is a second-degree polynomial fit for illustration purposes). The performance of the single-timescale integrator peaks for intermediate values of the associated timescale τ, though it always stays below the performance attained by the agent. The grey strip around the agent's line marks the 25%-75% of the values obtained for the performance upon 100 repetitions of the training procedure (see Fig 5 for further details). **C** and **D**: Accuracy and mean response times for different values of coherence (dots). **C**: The accuracy curve for the agent is in very good agreement with experimental findings: the black line is the result of a fit on experimental data ([5]; see text for more details). **D**: As accuracy increases, responses become faster, as found in experiments (black line: fit with a sigmoid-like function).

and its 'left' counterpart $\Delta\Sigma_{\text{left}}(t) \equiv \Sigma_{\text{left}}(t) - \Sigma_{\text{wait}}(t)$. Eq 21 ($\Delta\Sigma_{left}$) provides a direct measure of the propensity of the agent to make a 'right' ('left') decision at time t.

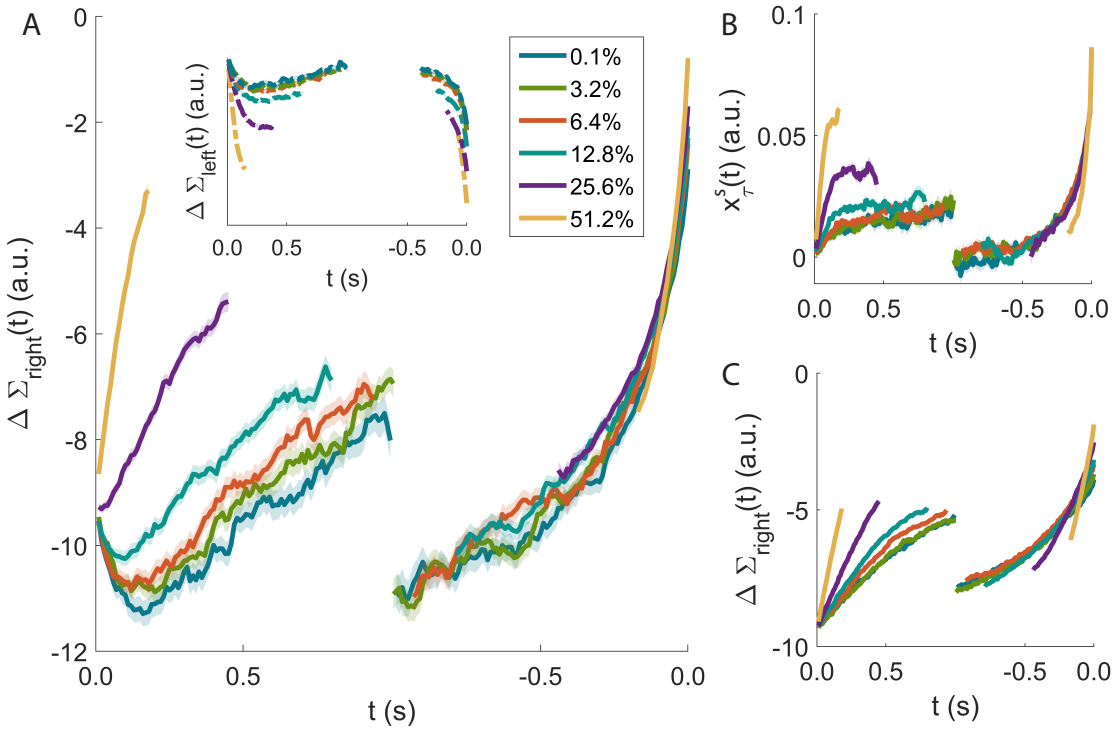

Figure 4. Signal neutrality. $\Delta\Sigma_{\text{right}}(t)$ (see Eq 21) provides a direct measure of the propensity of the agent to make a 'right' decision at time t. **A** Evolution of $\Delta\Sigma_{\text{right}}$, averaged over many successful episodes with the same signal coherence. On the left, the episodes are aligned to the beginning of the episode and $\Delta\Sigma_{\text{right}}$ shows a marked sensitivity to the coherence of the signal. When the average is performed by aligning all the episodes to the time of the decision (right), signal neutrality clearly appears: the sensitivity to the signal strength is completely lost and all the lines collapse on the same curve for several hundreds of milliseconds. Inset: the same analysis on wrong episodes. The similarities with what is found in the discharge of LIP neurons during a motion-discrimination task are striking (see, *e.g.*, Fig 7 in [5]). **B**: Time course of x^s_τ for a single-timescale integrator with $\tau = 2s$ and optimised decision threshold (x^s_τ, for an integrator with threshold, plays the role that $\Delta\Sigma_{\text{right}}$ has in the agent). **C**: Time course of $\Delta\Sigma_{\text{right}}$ (see Eq 21 for an agent optimised with a single timescale $\tau = 2s$). In both **B** and **C** the collapse of the curves for different signal coherences is imperfect (rightmost part of the plots).

Fig 4A shows the evolution of $\Delta\Sigma_{\text{right}}$, averaged over many episodes in which the agent has made the correct decision 'right'. The traces are grouped by signal coherence. The left part of Fig 4A shows the evolution of the average $\Delta\Sigma_{\text{right}}$, with traces aligned to the beginning of the episode (onset of the external signal). $\Delta\Sigma_{\text{right}}$ shows a marked sensitivity to the coherence of the signal. Moreover, the traces do not saturate over several hundreds of milliseconds, highlighting how the agent is making use of its slower integrators.

Ramp-like changes in the discharge of LIP neurons have been repeatedly observed,

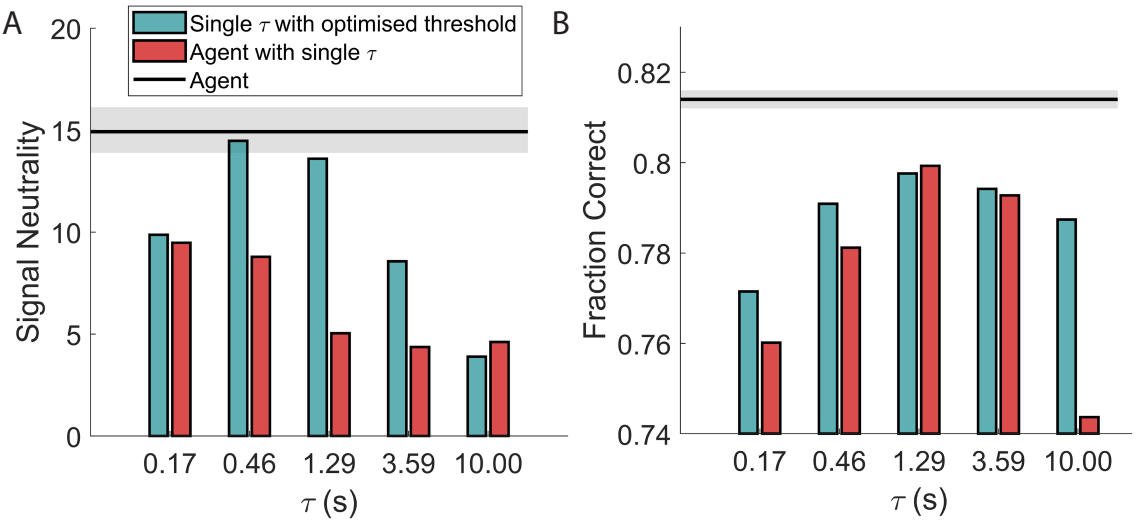

Figure 5. Comparison of signal neutrality (**A**) and performance (**B**) for the single-τ agent and the single-timescale integrator as τ varies. The proposed model (black horizontal lines) shows better accuracy while exhibiting the experimentally observed collapse of the time course of neuronal activity aligned at the decision time. The grey area marks the 25%-75% of the values obtained for each of the two observables upon 100 repetitions of the training procedure; more specifically, each independent training has been halted where signal neutrality peaked, conditioned to having already reached a performance of 0.81 or above; this translates to an average training length of about 73000 episodes (10%-90% range: 39000–110000).

with steeper rise in spike rate for higher stimulus coherence (see, *e.g.*, Fig 7 in [5]). Such ramps, originating in the extrastriate visual cortex in the case of LIP neurons, have been interpreted as a signature of the accumulation of evidence for or against a specific behavioural response [10, 17]. This interpretation is fully compatible with what is seen in the agent.

However, when the averages of the $\Delta\Sigma_{\text{right}}$ traces (or of the activity of LIP neurons) are performed by aligning the episodes to the time of the decision, a clear signature of signal neutrality emerges. The sensitivity to the stimulus's coherence is lost and all the lines surprisingly collapse on the same curve for several hundreds of milliseconds (Fig 4A, right). We emphasise that such collapse over an extended period of time is key to recognise signal neutrality: any decision model with a deterministic threshold, for example, would display a collapse at decision time (exactly at the threshold), but not necessarily at previous times; in this case, according to our definition, the model would not display signal neutrality.

For the experimental data, a reasonable explanation for such collapse is that the neuronal circuitry is engaged in stereotyped dynamics, independent from the signal, just after a decision is made and before it is manifested with a physical action, perhaps as the result of a feedback from downstream areas.

But this cannot hold for the agent, where instead signal neutrality arises precisely from the presence of multiple timescales. Fig 4B and 4C show the time course of the equivalent of $\Delta\Sigma_{\text{right}}$ for the models with a single timescale (see Section 2.4). For both these models, we display the results obtained from an example time constants of $\tau = 2.0$ s. In the single integrator with optimised threshold, x^s_τ plays the role that $\Delta\Sigma_{\text{right}}$ has in the agent.

In the latter, the collapse of the curves for different signal coherences is not as evident (Fig 4B and 4C, rightmost part). To make this statement more systematic, we introduce an operative measure of signal neutrality. We computed the inverse of the maximum distance between the curves for different coherences averaged over an interval of up to 600 ms prior to the decision (see Methods). In Fig 5A we report this measure for the agent (horizontal line) and the models with a single timecale (coloured upper bars). The comparative models report lower values in terms of signal neutrality and accuracy (Fig 5B).

The propensity of the agent $\Delta\Sigma_{\text{left}}$ to make the erroneous 'left' decision does not display signal neutrality. The same holds true for its experimental counterpart, that is the activity of LIP neurons when the random dot motion is away from their receptive field (see Fig 7 in [5], dashed lines). Finally, the comparison between the models in Fig 4 emphasises how in our simulations the signal neutrality is a consequence of the availability of multiple timescales.

3.3 The scalar property

The agent's behaviour conforms to one of the hallmarks of temporal cognition: the scalar property or Weber's law for interval timing [42]. This is illustrated in Fig 6A, where the distributions of response times of the agent are shown for three different values of coherence. As the coherence increases, the average response time of the agent decreases from 4.6 s to

370 ms.

Simply stated, the scalar property — as observed for example in interval timing [42], and multistable perception [44] — implies that higher moments of the intervals' distribution scale as appropriate powers of the mean. This implies a constant coefficient of variation. In other words, the shape of the distribution does not change when its mean varies even over wide ranges.

Notwithstanding a mean value that varies by more than one order of magnitude, the coefficient of variation of the agent moves in a very narrow range which is compatible with the experimental findings [42, 44]. The invariance of the shape of the distribution is made immediately evident in the inset of Fig 6A. Here the fitted Gamma distributions (black lines in the main plot) are rescaled to have mean equal to 1. The similarity of the three curves is striking. Fig 6B shows the coefficient of variation CV as the coherence varies for the proposed agent (black) and the comparative models (blue and red colours, see Section 2.4 for more details). The coefficient of variation has an approximately constant value for the proposed agent only. We remark that an agent with a single integrator has information regarding the passage of time over a single time constant, and that the model depicted in blue has multiple integrators but lacks any explicit temporal information. Thus, the key ingredient for the scalar property is again the availability of multiple timescales, in particular on the estimate of the passage of time.

On the other hand, it is not surprising that the single integrator with optimised threshold is unable to display the scalar property. In fact, for the pure drift-diffusion model ($\tau = \infty$), the coefficient of variation as a function of the coherence c can be computed analytically [45] (see also Eq 1):

$$\mathrm{CV} = \Big(\frac{100 - c}{c^2}\Big)^{1/4}, \tag{22}$$

and it is clearly not constant.

Lastly, we note how the highest values of coherence reported in the plots are very

unlikely under the distribution used during the training phase. A coherence of 50% roughly corresponds to a value of μ that is five times the standard deviation σ_μ of the distribution of μ. Thus, the scalar property appears to be a very robust property of the learned decision strategy of the proposed agent, holding well beyond the range of functioning to which the agent has been accustomed during training.

In view of the above considerations, signal neutrality and the scalar property share a similar origin. Further evidence of this can be found in the evolution of the two measures during the training phase.

Fig 7 shows the average evolution of signal neutrality (black line; the same measure reported in Fig 4D), scalar property (blue line; see Methods for the definition of the metric), and accuracy (dashed red line, scale on the right y-axis) during training. All the lines are computed by averaging the results of 100 different realisations of the training.

The evolution of signal neutrality and the scalar property are highly correlated for much of the training phase, with an initial fast increase that continues up to about $10^4 - 10^5$ episodes, where the accuracy has almost plateaued (the region used for the results of Figs 4A and 6; note how, after the first 10^5 episodes, the following $9 \cdot 10^5$ lead to a modest performance gain of $\simeq 1\%$). Such correlated progress naturally hints to a common origin for the two measures, and makes us advance the hypothesis that a behavioural policy displaying these two properties could represent an 'optimal' information-extraction strategy for dealing with a decision task in a volatile environment. It wouldn't be by chance that the agent robustly finds such a strategy by tuning its parameters in a ecologically plausible way.

Yet, after about 10^5 training episodes, and therefore probably far beyond the experimental training duration, the behaviour of the two curves in Fig 7 starts to diverge. Whilst the scalar property keeps improving, signal neutrality attains a broad peak, after which it gradually breaks down in the face of very modest performance gains. Therefore, the scalar property seems to be more fundamental than signal neutrality, at least for what concerns the strategy asymptotically discovered by the learning agent.

In this sense, signal neutrality cannot be viewed *per se* as signature of an optimal strategy for the agent, but rather of a 'satisficing' one [46]. Faced with a wide distribution of coherences, the agent pretty quickly finds a robust strategy that, at around decision time, disregards coherence by relying on fluctuations to make decisions, and still ensures a very good performance. Nevertheless, the agent can do slightly better, given enough training time, by giving more weight to the 'drift' component and less to the 'diffusion' component: this is what happens on the far right of the plot. In this region, we postulate, the learning enters an 'overfitting' phase, meaning that the agents become finely attuned to the exact statistics of the task: any slight changes, for example, in the shape of $p(\mu)$ would require many training episodes to revert to a good performance. In this sense, the signal neutral strategy generalises better to novel situations. This is something we plan to study elsewhere. Finally, it is tempting to hypothesise that animal subjects, during perceptual decision experiments, display signal neutrality as a reflex of adopting such a satisficing strategy, given also the high number of training episodes the model needs to refine its strategy beyond signal neutrality.

3.4 Collapsing boundaries

It is known that in the presence of a distribution of signal-to-noise ratios and limited decision time, as in the task at hand, the drift-diffusion model is not optimal anymore [15]. More specifically, one ingredient that allows to re-establish optimality is a time-varying threshold. As it has been observed in [9] [11], the optimal decision threshold is not constant when the agent has a finite amount of time to make decisions, but is characterised by a non-monotonic trend across time. This optimal moving threshold is defined as collapsing boundaries. In this sense, the hypothesised optimality of the agent's strategy finds indirect support in the behaviour displayed by the component of $\Delta\Sigma$ that depends only on the passage of time and not on the signal. As we will show, this perception of the passage of time, defined in the model as integration of a constant input over multiple timescales, permits

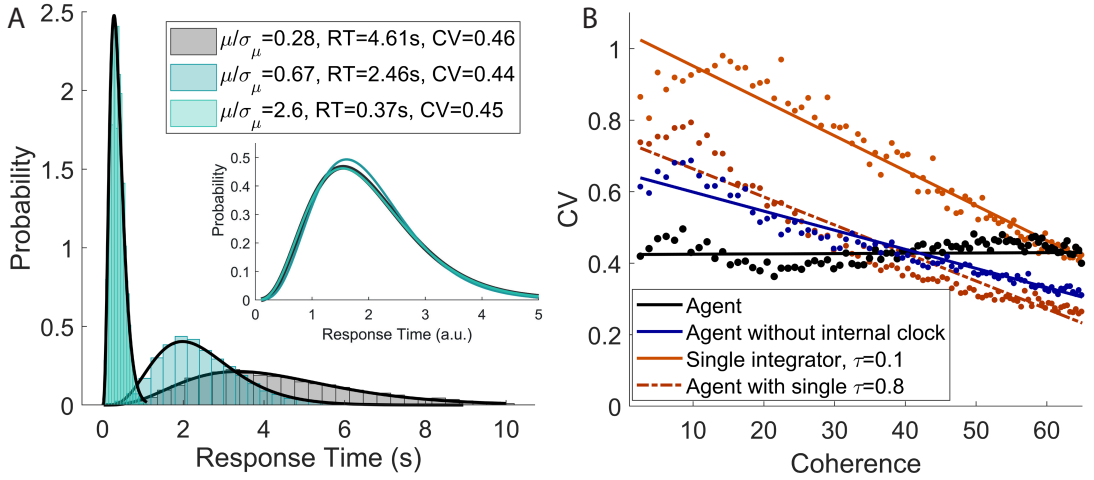

Figure 6. Scalar property. **A**: The average response time of the agent decreases as the signal coherence increases; still the coefficient of variation of the response times varies in a very narrow range (see legend). The black lines are the best fit of the simulation histograms with a Gamma distribution. Inset: the fitted Gamma distributions are rescaled to have mean equal to 1, making immediately evident how the shape of the distribution stays almost unchanged as its average moves over almost one order of magnitude (colours consistent with the histograms in the main plot). Note how the highest value of coherence is very unlikely under the distribution used for training the agent (corresponding to a value of μ five times the standard deviation σ_μ of the distribution of μ). The 'invariant shape' property of the response time distribution therefore holds well beyond the typical range of functioning of the agent. **B**: Coefficient of variation (CV) of the different models as the coherence increases. The scalar property is satisfied exclusively by the proposed agent (black line). The single timescale models are reported with two different values of τ. Other choices of τ give comparable results.

the agent to discover the collapsing boundaries. We rewrite Eq 21 as (see Eqs 10-16):

$$\Delta\Sigma_{\text{right}} = \Delta\Sigma^s_{\text{c}} - \Delta\Sigma^c \tag{23}$$

where:

$$\Delta\Sigma^s_{\text{right}} \equiv \Sigma^s_{\text{right}} - \Sigma^s_{\text{wait}} \tag{24}$$

is a term that provides information on the signal only. And:

$$\Delta\Sigma^c \equiv \Sigma^c_{\text{wait}} - \Sigma^c_{\text{right}} \tag{25}$$

carries information on the passage of time only. We note that on the r.h.s. of Eq 25 we could insert Σ^c_{left} in place of Σ^c_{right} with no notable numerical difference in the result.

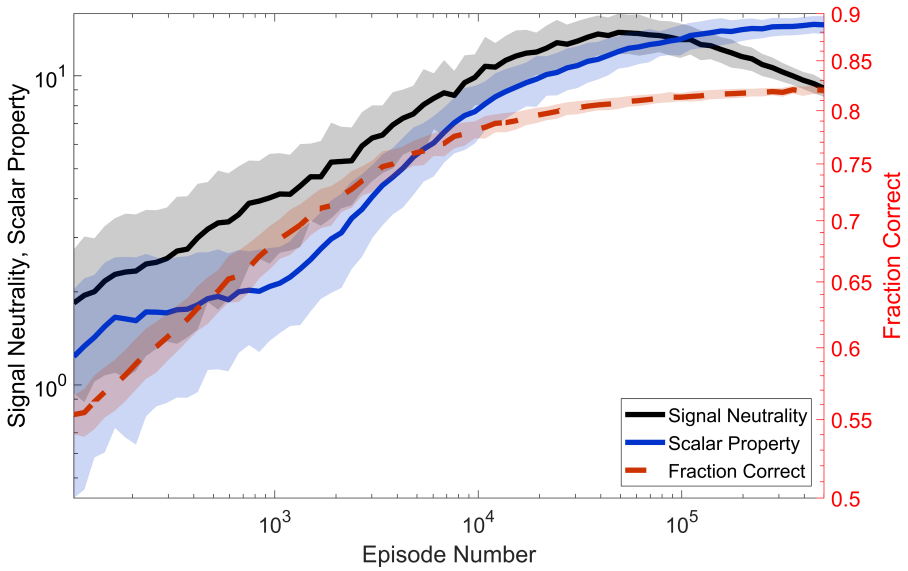

Figure 7. Signal neutrality and scalar property during training. Evolution of signal neutrality (black line), scalar property (blue line), and accuracy (dashed red line, scale on the right) as the training progresses. Signal neutrality attains a broad maximum where the performance has almost plateaued. Thus signal neutrality can be interpreted as the signature of a 'satisficing' strategy, rather than of an optimal one. The scalar property, on the other hand, keeps growing even for very long training. Yet, the evolution of signal neutrality and the scalar property are highly correlated, suggesting a common origin for the two (see text for discussion).

This is because the right and left choices are *a priori* equivalent in the present task, and therefore the inferred $\theta^c_{\text{right},\tau}$ and $\theta^c_{\text{left},\tau}$ are in fact very similar. For this reason $\Delta\Sigma^c$ does not carry a 'right' label.

$\Delta\Sigma^c(t)$ measures the propensity of the agent at time t to wait for another input instead of making a (either right or left) decision, independently from the signal. Looking back at Eq 23, $\Delta\Sigma^c$ effectively acts as a time-dependent bias term that, in the context of a drift-diffusion model, could be easily interpreted as a time-dependent threshold. Despite the lack of an explicit threshold mechanism for the proposed agent, it is reasonable to expect that the range of values attained by Σ^s_{right} at decision time shifts in accordance with the time-dependent bias. This is indeed the case.

Fig 8A shows (black thick line) the evolution of $\Delta\Sigma^c(t)$ from 0 to $T_{\max} = 2$ s for the proposed agent. In addition, three sample trajectories of $\Delta\Sigma^s(t)$ (coloured lines) are shown from $t = 0$ to decision time (marked by the big coloured circles). The shaded grey area

marks the region of values assumed by Σ^s_{right} where 80% of the (correct) decisions are made. As expected, this region mostly run parallel to $\Delta\Sigma^c(t)$, demonstrating how the latter observable can be interpreted as a soft threshold for the decision that arises from the time integrators. Such threshold drops at longer times, a behaviour that finds normative support in the study of perceptual decision making [20, 47]. Conversely, looking at Eq 23, one can view $-\Delta\Sigma^c$ as an 'urgency' signal that pushes for a decision as the episode time elapses, not unlike what has been observed experimentally in the lateral intraparietal area [48].

Fig 8B and 8C report the same analysis for an agent with a single timescale (panel B) and an agent with multiple timescales on the signal but without the internal clock (panel C). It is evident how the agent in Fig 8B exploits the unique timescale available for the internal clock to implement a monotonically decaying threshold. In contrast, the agent without internal clock is unable to create such mechanism, considering that the large majority of the decisions occur in an area that is parallel to the constant bias $b_{wait} - b_{right}$. The agent of panel C is unable to clearly infer the passage of time from the multiple timescale of the signal. If this limited behaviour can be surprising at first, it can be understood by considering that the present task is highly volatile, with a broad range of signal to noise ratios. Since specific values of the signal integrators $\mathbf{x}^s_\tau$ can be reached rapidly (slowly) for episodes with high (low) coherences of the signal, such features do not constitute a reliable estimate of the passage of time. Indeed, the model in panel C fails in the implementation of any form of urgency signal.

In this respect we want to point out how the soft threshold $\Delta\Sigma^c$ of the proposed model (panel A) does not simply behave as an urgency signal. In fact the decision is made more and more likely as the time passes only after about 200 ms (when $\Delta\Sigma^c$ reaches a peak). Initially, earlier decisions are discouraged by a rise of the threshold. Interestingly, such a non-monotonic trend of the moving threshold has been demonstrated to be theoretically optimal in [9] (see Fig 2B therein; see also [11]).

Even if the models in the references and in the present paper are not structurally

equivalent, it is nonetheless striking that the agent can approximate such optimal behaviour by trial-and-error. We note how the monotonically decreasing $\Delta\Sigma^c$ shown in panel B is consequently suboptimal. Thus, the results of Fig 8 demonstrate the necessity of multiple timescales also for an efficient implementation of the collapsing boundaries.

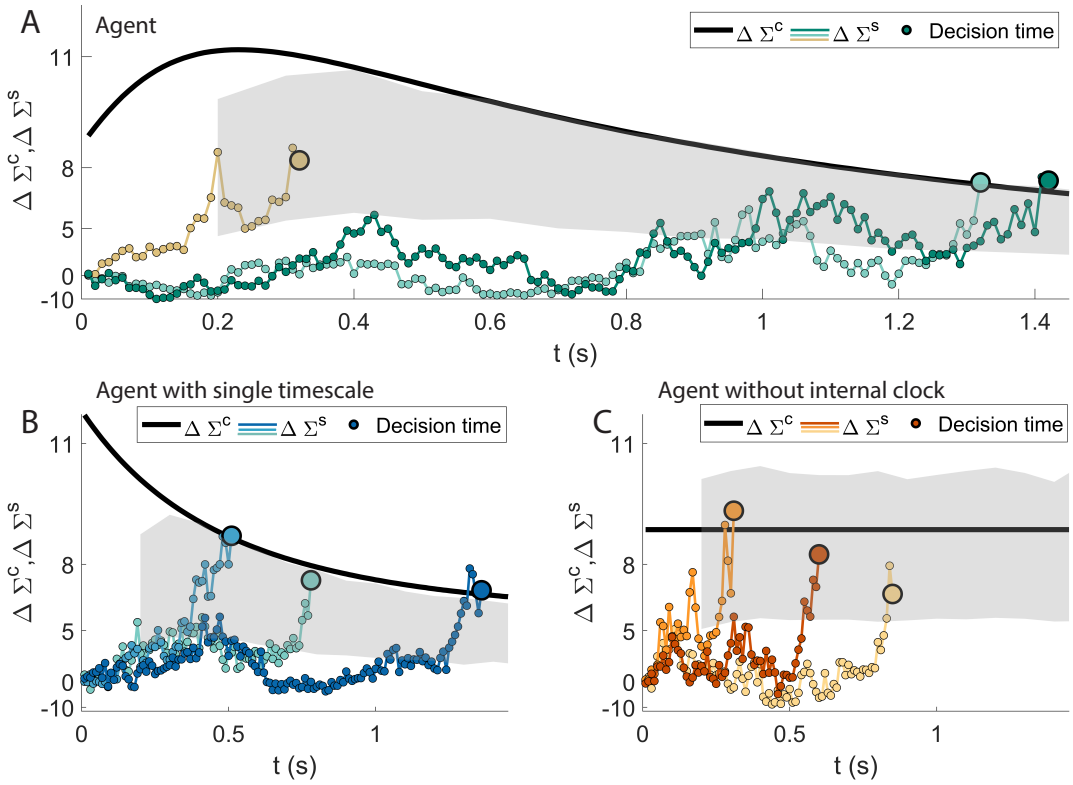

Figure 8. Collapsing boundaries. $\Delta\Sigma_{\text{right}}$ (see Eqs 21 and 23) can be decomposed in a signal-dependent part ($\Delta\Sigma^s_{\text{right}}$) and a time-dependent part ($\Delta\Sigma^c$; see Eq 25), that measures the propensity of the agent at each time to wait for another input instead of making a decision. In all panels, $\Delta\Sigma^s_{\text{right}}$ (coloured lines) is depicted for three sample episodes, alongside $\Delta\Sigma^c$ (thick black line). The big coloured circles correspond to the decision times. **A**: The behaviour of the proposed agent. $\Delta\Sigma^c$ acts as a time-dependent threshold: most of the decisions fall inside a strip running parallel to it (the grey area is where 80% of the decisions are made). The resulting boundaries collapse for longer response times. Until about 200 ms, a rise of the effective threshold discourages early decisions. This trend is analogous to the theoretically optimal decision threshold when the trial has a maximum allowed time to make a decision [9, 11]. **B**: The behaviour of an agent with a single integrator (see Section 2.4). Thanks to the internal clock over a single timescale, the agent can implement a suboptimal, monotonically decreasing threshold. **C:** Behaviour of the agent without internal clock, but with multiple signal integrators. The model is unable to exhibit the collapsing boundaries.

3.5 Robustness

The utilisation of a wide range of timescales makes the performance of the agent robust to variation of the task and to the intrinsic noise. This is shown in Fig 9A and 9B. We varied $T_{\max}$ (the maximum duration of an episode) and σ_I (the standard deviation of the intrinsic noise, ξ_τ^ss and ξ_τ^cs in Eqs 10-15) systematically and, for each value, run the learning process from scratch. The results of the agent are then compared to the models of Section 2.4. While Fig 9 reports the comparison with the single integrators with optimised thresholds, the results for the agent with a single timescale can be found in Supplementary Material and Fig 11.

In Fig 9A, as $T_{\max}$ increases (and σ_I stays at its reference point of 0.02), the fraction of correct responses rises monotonically for all models, with the performance of the agent staying superior on the whole range of $T_{\max}$ explored. Two features are noteworthy. First, the lines for the single integrators ($\tau = 0.1$ s and $\tau = 10$ s respectively) cross at intermediate values of $T_{\max}$, with the longer τ surpassing the shorter ones for higher episode durations. Second, the advantage of the proposed agent shrinks in comparison to the longer τ for longer $T_{\max}$. These features have a common origin. From Eq 6, a signal s(t) of mean μ will asymptotically lead all the integrators to the same (statistically) stationary value of μ, but with different levels of noise. Integrators with longer τs will have a smaller variance and thus will be more reliable in detecting whether $\mu > 0$ or $\mu < 0$. On the other hand, the time needed to reach the stationary state will be longer for larger τs. Slower integrators will still be integrating the signal for shorter $T_{\max}$ and, as a consequence, their value will carry less information on the μ. Hence, the smaller τs will dominate for shorter $T_{\max}$, the larger τ for longer $T_{\max}$.

In Fig 9B, the level σ_I of intrinsic noise is varied, with $T_{\max}$ kept constant at 2 s. The performance of the agent (black line) is always substantially higher than that of the single integrators (coloured lines). As expected, performance deteriorates as σ_I increases from 0 to 0.2; yet the decrease is only surprisingly slight, considering that the maximum

value attained by σ_I is comparable with the typical dynamical range of the integrators x_τ. Such range is determined by the distribution $p(\mu)$ (here, a Gaussian of standard deviation $\sigma_\mu = 0.25$). It is then clear that the highest levels of intrinsic noise really affect the typical value of the integrators. This is even more true taking into account that the slowest integrators operate far from the asymptotic value, given the limited integration time. This consideration is clearly reflected in the behaviour of the single integrators. The fast integrators ($\tau = 0.1$ s and $\tau = 2.1$ s) indeed are scarcely affected by the increase in noise. On the other hand, the slowest integrator ($\tau = 10$ s) shows good accuracy for very low levels of noise, but then becomes rapidly ineffective for higher values of σ_I.

The agent without the internal clock reports robust performance as T_{max} and σ_I vary, demonstrating its capability to select the appropriate signal integrator for different conditions. However, this agent can also report lower performance in comparison to the best single integrators. This phenomenon is a consequence of the reduction of the noise performed on the model with optimised threshold. Indeed, integration of the signal over a single timescale can carry the large majority of the relevant information for a specific simulation. In a specific parameters' setting and in terms of accuracy only, the performance of this agent can be consequently inferior to the model with a single integrator with an optimal value of τ. However, it is clear how the agent depicted in blue is more robust over the range of values shown in the figure, having the possibility to choose the appropriate integration time. For the sake of accuracy optimisation, the role of the reservoir of integrators is to select the appropriate timescale for the considered situation. Instead, the advantage of an internal clock parameterised by multiple timescales is to implement the optimal shape of collapsing boundaries of Fig 8. The latter statement is emphasised by the improved performance of the proposed model also over the agent with a single timescale (S3 Fig), which has information about the passage of time limited over a single τ.

Fig 9C shows the evolution of the 'moving threshold' $\Delta\Sigma^c$ (Eq 25, proposed agent) for three values of $T_{\max}$. For very low $T_{\max}$ (black line) the threshold only decays, always pushing for a decision. For higher values of $T_{\max}$, instead, as we have already seen in Fig 8,

the moving threshold initially rises; it reaches a peak and then decays afterwards, making a decision ever more likely. Such peak shifts with $T_{\max}$ and so does, even more clearly, the time at which the threshold reaches back its initial value (around 1 s for $T_{\max} = 2.0$ s, and around 5 seconds for $T_{\max} = 10$ s).

Fig 9D shows $\theta^s_{\text{right},\tau}$ after training ($\theta^s_{\text{right},\tau} \simeq -\theta^s_{\text{left},\tau}$ for the symmetry of the problem after optimisation, as it is shown in Fig 10 D) for different values of intrinsic noise σ_I (continuous lines are fourth degree polynomial fits for illustrative purposes). Coherently with what we have seen in Fig 9B, the peak of the lines, corresponding to the most exploited timescale, shifts towards lower τ values as σ_I increases.

3.6 Evolution during training

Fig 10 illustrates how the behaviour of the agent evolves as it encounters new episodes during learning. Fig 10A shows the performance attained on average for four different values of signal coherence at different times during the training phase. The performance is of course always higher for higher values of coherence ('easier' episodes), and tends to increase monotonically for all the values of coherence during training.

This monotonic trend is not preserved, instead, looking at the average response time (Fig 10B). The response time drops at the beginning of training with values that are very close for every value of coherence. The reason for such behaviour is related to how the agent is initialised. At the beginning, the agent is ignorant about the rules of the task and pre-programmed to make a random choice after having waited for a finite random length of time. Without such random initialisation, the learning would not proceed, since the agent needs to perform actions to learn the relative consequences. While the agent is unable to tell apart signals with different coherences, the response time then decreases. In fact, longer average response times are detrimental due to late responses (no decision before the maximum time allowed $T_{\max}$) that are not rewarded.

This is made clear in Fig 10B, that shows how the fraction of late responses quickly drops to almost zero, and it stays there. Afterwards, the model starts to statistically

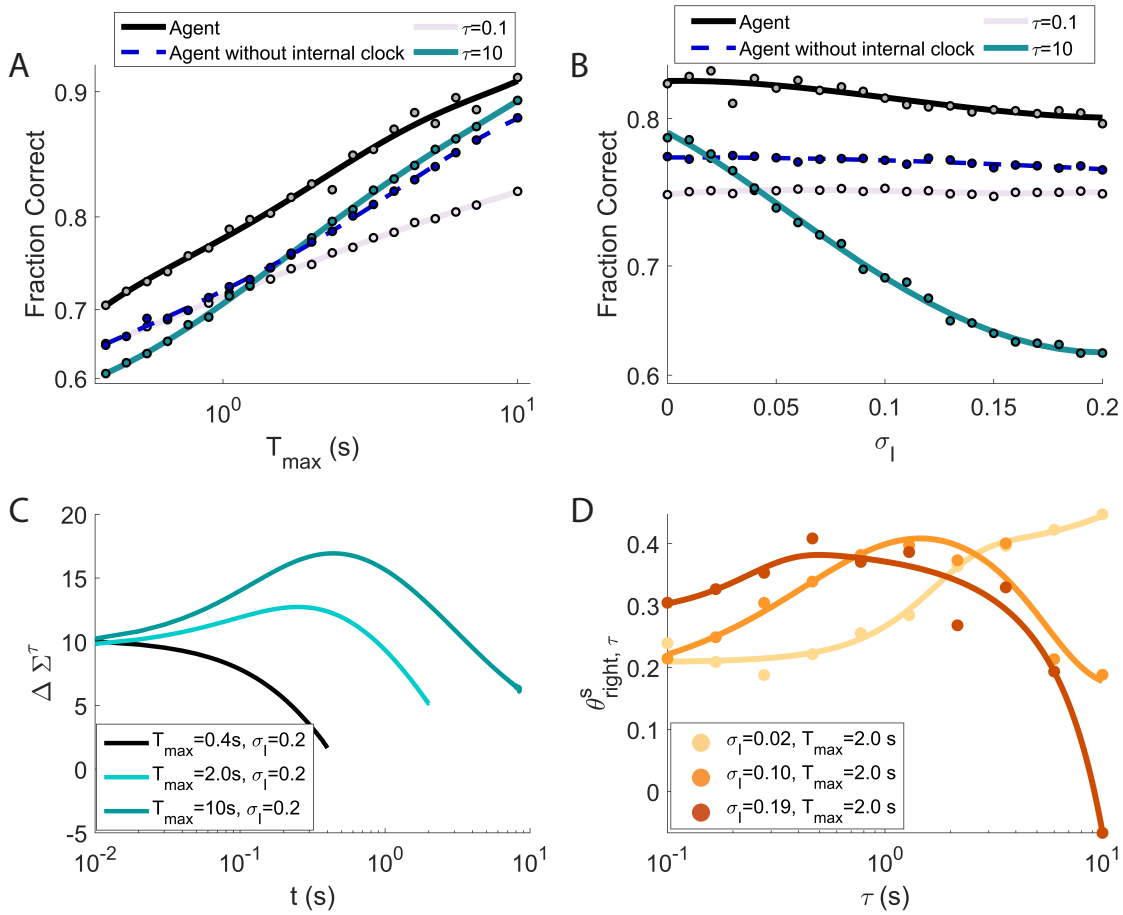

Figure 9. The wide range of timescales makes the agent's performance robust to variations of the task and to the intrinsic noise. **A**: as $T_{\max}$ increases, the fraction of correct responses rises monotonically both for the agent (dashed black line) and for all the single integrators, with the performance of the agent staying superior on the whole range of $T_{\max}$ explored. **B**: varying the level σ_I of intrinsic noise, the performance of the agent (dashed black line) stays always substantially higher than that of the single integrators, notably for stronger noise. As expected, the performance does deteriorate, but the decrease is surprisingly slight, considering that the maximum value attained by σ_I is comparable with the typical dynamical range of the integrators x_τ. The performance of the agent without internal clock (dashed blue in panels **A** and **B**) are close (or superior to) the best single integrators reported. However, this agent is more robust than the single integrators over the range of parameters' values considered. Thus, the results show how the model (dashed blue) is able to select the appropriate timescale for different situations (see text for more details). **C**: evolution of the 'moving threshold' $\Delta\Sigma^c$ (Eq 25) for three values of $T_{\max}$. For higher values of $T_{\max}$ (see also Fig 8), the moving threshold presents a peak whose position shifts with $T_{\max}$. **D**: $\theta^s_{\text{right},\tau}$ after training (Eq 10) for different values of intrinsic noise σ_I (continuous lines are fourth degree polynomial fits for illustrative purposes). The peak of the lines, corresponding to the most exploited timescale, shifts towards lower τ values as σ_I increases.

differentiate between signals with different coherences (the four lines diverge in Fig 10B) and the response time begins to rise. In this regime, waiting means accumulating more

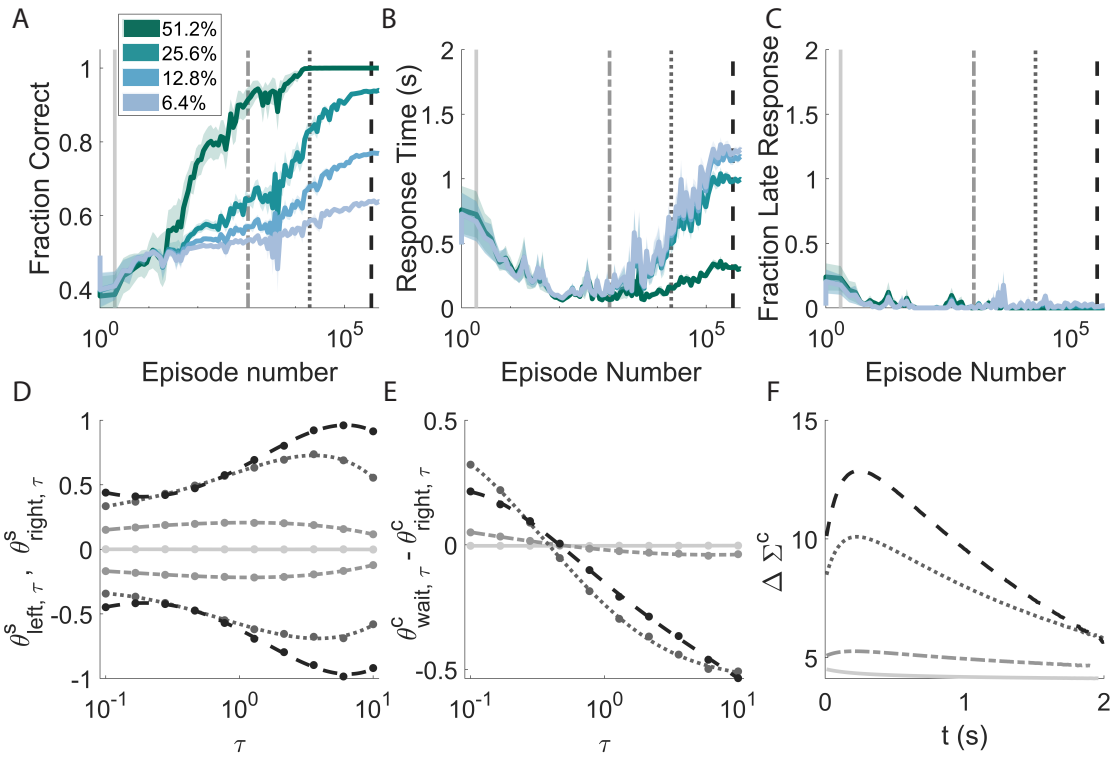

Figure 10. Learning is characterised by a non-monotonic adaptation of the average response time that is consequent to the necessity of finding a fine balance between integrating information and the cost of waiting to make decisions. **A**: Accuracy of the model for signals with different coherences across learning. **B**: Average response times. Trials with increasing level of coherences correspond to greater response times and greater probabilities of 'late' responses. The initial descending trend (around 100 episodes) of the response times common to all coherences is due to the initial ignorance of the agent about the nature of the task, on the tendency to avoid late decisions and to prefer immediate rewards. C: Probability of not making a decision before the end of the episode, i.e. after $T_{\max}$. **D-E-F**: Evolution of different parameters and of the collapsing boundaries for the training instances corresponding to the vertical grey lines of the top panels. **D**: Evolution of the weights corresponding to the integrators of the signal. The weights are positive (negative) for the 'right' ('left') action. **E-F**: Values of the parameters (**E**) defining the contribution of the internal clock to the decision making process and relative collapsing boundaries (**F**).

information and helps to improve the performance.

Fig 10D and 10E show the evolution of $\theta^s_{\text{right},\tau}$, $\theta^s_{\text{left},\tau}$, and of $\theta^c_{\text{wait},\tau} - \theta^c_{\text{right},\tau}$ respectively (the shape of $\theta^c_{\text{wait},\tau} - \theta^c_{\text{left},\tau}$ is similar). The different colours (grey scale) and line styles correspond to the training instances highlighted with the vertical lines of the above panels. Each set of weights is separately rescaled, for each instance, by its absolute maximum value. This has been done to emphasise the relative importance among the parameters

rather than their magnitude. At the beginning of training (light grey lines) and despite the rescaling, the weights are close to zero because we initialised the biases of the model at higher absolute values. We chose this initialisation so that the model could exhibit reasonable starting response times without weighting the contribution of the different timescales *a priori*.

As the simulation progresses, the weights for the signal integration toward the 'right' and 'left' actions become stronger while maintaining an approximately symmetric trend with respect to zero (panel D). Also the parameters reflecting the internal clock in panel E grow across training, but fast (slow) timescales become positively (negatively) weighted. Thus, the weights at a given instance compose an overall descending trend. To understand this, we need to make two simple observations. First, time integrators receive a constant positive signal of one as input. Thus, the sign of the contribution of a time integrator to the decision process corresponds to the sign of its relative weight. Second, such weights are the ones responsible for the implementation of the collapsing boundaries (Fig 10F shows the boundaries corresponding to the considered training instances). By giving positive importance to the fast integrators, which are dominant at the beginning of an episode, the agent is implementing the initial rise of the effective, moving threshold (panel F). In contrast, the negative contribution from the slow τs is responsible for the collapsing trends reported in panel F.

4 Discussion

Decision making and reinforcement learning are fields with overlapping contributions that attempt to demystify how humans and animals make decisions. Our work unifies the two approaches by using a reinforcement learning agent to solve a task reminiscent of a classical perceptual decision making setup, similar to [10, 49–51].

The reinforcement learning agent receives sensory information and information from various "clocks" integrated on multiple timescales. Timescales have been implicit in the reinforcement learning framework, in the context of propagating information about the success (or failure) of the task in cases where the reward is not immediate, see eligibility traces [43, 52]. However, this is not the same as the concept of timescales in this model, where the emphasis is on acquiring and retaining sensory information from the environment, not unlike what happens in the field of Reservoir Computing [53]. We argue that reward maximisation, multiple time constants and perception of time are the fundamental ingredients for faithfully reproducing (i) an optimal decision-making boundary, (ii) the scalar property, and (iii) signal neutrality.

Indeed, the agent learns to solve the task in a relatively small number of episodes, performing better than any single-timescale drift-diffusion integrator while fitting well the psychophysical data. The agent's policy is markedly different from the drift-diffusion model, where a decision happens when one of the integrating processes reach the decision threshold. The reinforcement learning agent makes decisions within short 'active' time windows when fleeting bursts in the probability of choosing an action make that action possible. These "bursts" result from the broad agreement on the decision of many integrators with different timescales, akin to the concept of majority voting. The behaviour of our agent is compatible with the analysis performed in [54] on single-neuron single-trial spike trains in LIP area to uncover sudden activity jumps and their informativeness about choice.

The multiple clock time constants lead to a decision boundary with a shape similar to the theoretical optimal for decisions with bounded time. We demonstrate in simulations

that such a complex boundary is not learnable with a single timescale in the clock or without using a clock. The initial increment and then collapse of the decision boundary happen due to the interplay of clock integrators with multiple time constants.

Another direct consequence of the clock with multiple timescales is the scalar property [42,44], i.e. the ratio of the standard deviation over the mean of the response times remains constant. The removal of the clock results in a fixed boundary over time and cannot exhibit the scalar property. Even a single timescale clock, which leads to a non-optimal decaying decision boundary, fails to capture the scalar property. Our results suggest that an optimal decision-making boundary may lead to the scalar property.

As a side note, and contrary to [42, 44], the experimental results reported in [55] seem to support a linear relationship between response means and standard deviations ($y = ax + b$) rather than an exact scalar property ($y = ax$). Yet the very low coefficients of variation for fast responses could result from ignoring the effect of non-decision times, which can be assumed to have low variability [20]. In principle, the drift-diffusion model is capable of exhibiting a linear relationship between the mean response time and its standard deviation [55]. Yet, it seems difficult to display the scalar property (see Eq 22).

The multiple timescale signal integrators offer a way to "learn" the integration time constant from the data instead of treating it as a free parameter. Their presence makes the agent robust, as it can perform well with various task difficulties (expressed as a signal to noise ratio or signal coherence). There are optimal integration time constants for specific task difficulties. Varying much the signal to noise ratio would inevitably reduce the performance of an integrator with a single time constant.

A consequence of signal integration with multiple time constants is signal neutrality, the stereotypical collapse of the decision-making signal just before the agent decides. This characteristic is noticeable in the activity of neurons in LIP area during a motion-discrimination task [5,41]. We can intuitively understand this characteristic in the following way. The agent has to find a policy that works across various coherences. A strategy independent of the specific coherences, if achievable, is an appropriate solution to the

problem. The simulations suggest the agent discovers such a policy. To some extent, also the single integrator agents find such a policy. However, if we vary the coherence much, signal neutrality progressively fails.

Fluctuations play a significant role in signal neutrality. And indeed, as far as we can discern, the observation of the phenomenon in [17], where no multiple timescales are present, is rooted in the presence of large fluctuations in the activity traces being averaged. These fluctuations are smoothed out by a first-order filter and a random post-decision time. Nonetheless, they contribute significantly to the observed collapse, as testified by peak values well above the decision threshold.

In summary, our agent learns solely by maximising its reward. There is no strategy *a priori* prescribed, similar to a biological agent during a perceptual decision-making experiment. And yet, our model provides little information about the corresponding mechanisms at the circuit level. Nevertheless, it offers insights into complex processes. We argue that it is a good trade-off between complexity and simplicity [56]: the agent learns when to take actions in an "optimal" way. The learning process suggests that the optimal decision boundary is a consequence of time perception in multiple timescales.

We underline how the proposed agent could be extended to tackle different perceptual decision tasks. For example, being probabilistic, the agent inherently computes an ongoing estimate of the confidence related to each of the possible options. Thus the agent could be presented with the possibility to opt-out from a trial when the choice appears too uncertain. Confidence has moreover been related, in the perceptual decision making literature, to optimal learning [47, 57]. It is interesting to note, in this respect, that the learning rate for the proposed agent is indeed strongly modulated by confidence: an easy correct decision would trigger little learning; on the other hand, a confident but wrong decision would engender large changes in the model's parameters. In [47], moreover, parameters' fluctuations due to the ongoing learning have been shown to account for differences in psychometric curves in an identification task, not unlike the one examined here, *versus* a categorisation task, to which the multi-τ agent could be adapted with minor modifications.

Since we have largely focused the attention on the post-learning phase, the role of such fluctuations in the agent remains an open, and stimulating, issue.

The building blocks of the present model, *i.e.* the signal accumulators, may have biological counterparts [58, 59]. It is possible to use pools of noisy attractors to implement integrators with wildly different timescales, as required by a multi-scale system. Attractor dynamics has been long one of the main staples of theoretical neuroscience [60]. Several winner-take-all spiking networks capable of implementing a probabilistic classification of the noisy signal have been described in the literature [61, 62]. Therefore we see no conceptual barriers to a more detailed, spiking model mimicking the workings of the agent.

Beyond specific interpretations in this work, we would like to advocate the consideration of multiple timescales in models handling non-stationary and noisy information. There is increasing evidence that performance improves or becomes more robust to changes in the environment when various elements are performing nearly the same task. Adapting to different conditions becomes possible by selectively choosing among those. We notice this general strategy, known as "degeneracy", is present in many biological systems [63–66]. Degeneracy permits rapid adaptation to novel conditions leading to robust performance, adaptability and survivability.

5 Supplementary Material

5.1 Learning

The learning algorithm adopted is a reinforcement learning actor-critic model with eligibility traces [43]. The goal of reinforcement learning is to maximise the cumulative reward:

$$G_t = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{t+N-1} R_{t+N} \tag{26}$$

where $0 \leq \gamma \leq 1$ is a discount factor that describes the tendency of the agent to invest in future rewards (here $\gamma \simeq 1 - 10^{-7}$). In our specific case, an episode ends when the agent

chooses 'left' or 'right', or the maximum allowed time $T_{\max}$ is reached without a decision. Rewards are given at the end of the episode only. The reward is 1 for the case of correct decision, and 0 otherwise. The policy (the actor) is embodied by the probabilities p_a ($a =$ 'right', 'left' or 'wait') defined in Eq. 17, Main Text. Instead, the parametrisation defining the critic, *i.e.* the value function $V(t) = \mathbf{E}_p\{G_t|s_t\}$, is:

$$V(t) = \sum_{\tau} \left\{ W^s_\tau |x^s_\tau(t)| + W^c_\tau x^c_\tau(t) \right\} + b_v \tag{27}$$

where the weights $\mathbf{W}$ are learned alongside $\boldsymbol{\theta}$ (with $\mathbf{W}$ and $\boldsymbol{\theta}$ we refer to all the parameters for the critic and actor respectively), and we used the absolute value of the integrators because, similarly to the definition of Σ_{wait}, positive and negative fluctuations of the signal should contribute in the same way to the expected reward. For each episode, the algorithm defines two sets of eligibility traces, $\mathbf{e}^W_t$ and $\mathbf{e}^\theta_t$, for the critic and the actor respectively:

$$\begin{aligned} \mathbf{e}^W_t &= \gamma\lambda^W \mathbf{e}^W_{t-\Delta t} + \gamma^t \nabla_{\mathbf{W}} V(t) \\ \mathbf{e}^\theta_t &= \gamma\lambda^\theta \mathbf{e}^\theta_{t-\Delta t} + \gamma^t \nabla_{\boldsymbol{\theta}} p_a(t) \end{aligned}$$

where t is the time inside an episode, and $0 \leq \lambda^\theta \leq 1$ and $0 \leq \lambda^W \leq 1$ are the traces decay parameters (here $\lambda^\theta = \lambda^W = 1 - 10^{-5}$). The parameters are then updated according to:

$$\begin{aligned} \delta &\equiv R_{t+\Delta t} + \gamma V(t+\Delta t) - V(t) \\ \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \eta^\theta \, \delta \, \mathbf{e}^\theta \\ \mathbf{W} &\leftarrow \mathbf{W} + \eta^W \, \delta \, \mathbf{e}^W. \end{aligned}$$

5.2 Interpretation as majority voting

Fig 11 shows the tendency of the agent to make decisions when there is a coherent alignment of the contributions of the different integrators of the signal.

characterised by a performance increase until $\tau_M \approx 10$, and then by a subsequent decrease. The reason is that for the usual setting considered ($\sigma_I = 0.02$ and $T_{max} = 2$ s) integrators with a timescale lower than 10 s can carry more relevant information than the slower ones. Thus, if we keep the number of features constant, the accuracy would be higher for the region with the most appropriate integrators. At the same time, variety is always preferred when increasing N_τ.

Fig 13 reports the performance of two agents, one with the exponential distribution of τ adopted in the paper (black line as usual), and the second with a linear distribution of τ over the same range (red lines). The performance of the two models remain comparable as T_{max} (panel A) and σ_I (panel B) vary. The result suggests how the specific distribution of timescales chosen does not impact considerably the results. Thus, it is more important than the τs are sufficiently spread over different magnitudes with enough density to cover the considered range. Moreover, Fig 13 shows the performance of the agent with a single integrator (see Section 2.4, Main Text). The behaviour of the different timescales as the parameters vary (x-axis of the panels) are analogous to the ones reported in Fig 13 for the single integrator with optimised thresholds. However, for the case of the agent reported here, the performance is usually higher than the single timescales of Fig 13, since this model can also exploit some temporal information.

References

[1] Burrhus F Skinner. Operant behavior. American psychologist, 18(8):503, 1963.

[2] Sashank Pisupati, Lital Chartarifsky-Lynn, Anup Khanal, and Anne K Churchland. Lapses in perceptual decisions reflect exploration. Elife, 10:e55490, 2021.

[3] SW Link and RA Heath. A sequential theory of psychological discrimination. Psychometrika, 40(1):77–105, 1975.

[4] Hauke R Heekeren, Sean Marrett, and Leslie G Ungerleider. The neural systems that mediate human perceptual decision making. Nature reviews neuroscience, 9(6):467–479, 2008.

[5] Jamie D Roitman and Michael N Shadlen. Response of neurons in the lateral intraparietal area during a combined visual discrimination reaction time task. Journal of neuroscience, 22(21):9475–9489, 2002.

[6] Joshua I Gold and Michael N Shadlen. The neural basis of decision making. Annual review of neuroscience, 30, 2007.

[7] Timothy D Hanks, Mark E Mazurek, Roozbeh Kiani, Elisabeth Hopp, and Michael N Shadlen. Elapsed decision time affects the weighting of prior probability in a perceptual decision task. Journal of Neuroscience, 31(17):6339–6352, 2011.

[8] Rafal Bogacz, Eric Brown, Jeff Moehlis, Philip Holmes, and Jonathan D Cohen. The physics of optimal decision making: a formal analysis of models of performance in two-alternative forced-choice tasks. Psychological review, 113(4):700, 2006.

[9] Jan Drugowitsch, Rubén Moreno-Bote, Anne K Churchland, Michael N Shadlen, and Alexandre Pouget. The cost of accumulating evidence in perceptual decision making. Journal of Neuroscience, 32(11):3612–3628, 2012.

[10] Rajesh PN Rao. Decision making under uncertainty: a neural model based on partially observable markov decision processes. Frontiers in computational neuroscience, 4:146, 2010.

[11] Tarryn Balsdon, Valentin Wyart, and Pascal Mamassian. Confidence controls perceptual evidence accumulation. Nature communications, 11(1):1–11, 2020.

[12] Joshua I Gold and Michael N Shadlen. Banburismus and the brain: decoding the relationship between sensory stimuli, decisions, and reward. Neuron, 36(2):299–308, 2002.

[13] Yong Liu and Steven D. Blostein. Optimality of the sequential probability ratio test for nonstationary observations. IEEE Transactions on Information Theory, 38(1):177–182, 1992.

[14] Rafal Bogacz. Optimal decision-making theories: linking neurobiology with behaviour. Trends in cognitive sciences, 11(3):118–125, 2007.

[15] Rani Moran. Optimal decision making in heterogeneous and biased environments. Psychonomic bulletin & review, 22(1):38–53, 2015.

[16] Dobromir Rahnev and Rachel N Denison. Suboptimality in perceptual decision making. Behavioral and Brain Sciences, 41, 2018.

[17] Mark E Mazurek, Jamie D Roitman, Jochen Ditterich, and Michael N Shadlen. A role for neural integrators in perceptual decision making. Cerebral cortex, 13(11):1257–1269, 2003.

[18] Scott D Brown and Andrew Heathcote. The simplest complete model of choice response time: Linear ballistic accumulation. Cognitive psychology, 57(3):153–178, 2008.

[19] Mads Lund Pedersen, Tor Endestad, and Guido Biele. Evidence accumulation and choice maintenance are dissociated in human perceptual decision making. PloS one, 10(10):e0140361, 2015.

[20] Gabriel M Stine, Ariel Zylberberg, Jochen Ditterich, and Michael N Shadlen. Differentiating between integration and non-integration strategies in perceptual decision making. Elife, 9:e55365, 2020.

[21] Roger Ratcliff. A theory of memory retrieval. Psychological review, 85(2):59, 1978.

[22] Abraham Wald and Jacob Wolfowitz. Optimum character of the sequential probability ratio test. The Annals of Mathematical Statistics, pages 326–339, 1948.

[23] Alex Roxin. Drift–diffusion models for multiple-alternative forced-choice decision making. The Journal of Mathematical Neuroscience, 9(1):5, 2019.

[24] Roger Ratcliff and Philip L Smith. A comparison of sequential sampling models for two-choice reaction time. Psychological review, 111(2):333, 2004.

[25] Philip L Smith and Roger Ratcliff. Psychology and neurobiology of simple decisions. Trends in neurosciences, 27(3):161–168, 2004.

[26] Jerome R Busemeyer and James T Townsend. Decision field theory: a dynamic-cognitive approach to decision making in an uncertain environment. Psychological review, 100(3):432, 1993.

[27] Marius Usher and James L McClelland. The time course of perceptual choice: the leaky, competing accumulator model. Psychological review, 108(3):550, 2001.

[28] Roger Ratcliff, Yukako T Hasegawa, Ryohei P Hasegawa, Philip L Smith, and Mark A Segraves. Dual diffusion model for single-cell recording data from the superior colliculus in a brightness-discrimination task. Journal of neurophysiology, 97(2):1756–1774, 2007.

[29] Ori Ossmy, Rani Moran, Thomas Pfeffer, Konstantinos Tsetsos, Marius Usher, and Tobias H Donner. The timescale of perceptual evidence integration can be adapted to the environment. Current Biology, 23(11):981–986, 2013.

[30] William T Newsome and Edmond B Pare. A selective impairment of motion perception following lesions of the middle temporal visual area (mt). Journal of Neuroscience, 8(6):2201–2211, 1988.

[31] Walter S Pritchard. The brain in fractal time: 1/f-like power spectrum scaling of the human electroencephalogram. International Journal of Neuroscience, 66(1-2):119–129, 1992.

[32] Stefan J Kiebel, Jean Daunizeau, and Karl J Friston. A hierarchy of time-scales and the brain. PLoS computational biology, 4(11), 2008.

[33] Catherine Wacongne, Etienne Labyt, Virginie van Wassenhove, Tristan Bekinschtein, Lionel Naccache, and Stanislas Dehaene. Evidence for a hierarchy of predictions and prediction errors in human cortex. Proceedings of the National Academy of Sciences, 108(51):20754–20759, 2011.

[34] Klaus Linkenkaer-Hansen, Vadim V Nikouline, J Matias Palva, and Risto J Ilmoniemi. Long-range temporal correlations and scaling behavior in human brain oscillations. Journal of Neuroscience, 21(4):1370–1377, 2001.

[35] Christopher J Honey, Rolf Kötter, Michael Breakspear, and Olaf Sporns. Network structure of cerebral cortex shapes functional connectivity on multiple time scales. Proceedings of the National Academy of Sciences, 104(24):10240–10245, 2007.

[36] Giancarlo La Camera, Alexander Rauch, David Thurbon, Hans-R Luscher, Walter Senn, and Stefano Fusi. Multiple time scales of temporal response in pyramidal and fast spiking cortical neurons. Journal of neurophysiology, 96(6):3448–3464, 2006.

[37] Dante Francisco Wasmuht, Eelke Spaak, Timothy J Buschman, Earl K Miller, and Mark G Stokes. Intrinsic neuronal dynamics predict distinct functional roles during working memory. Nature communications, 9(1):1–13, 2018.

[38] Sean E Cavanagh, John P Towers, Joni D Wallis, Laurence T Hunt, and Steven W Kennerley. Reconciling persistent and dynamic hypotheses of working memory coding in prefrontal cortex. Nature communications, 9(1):1–16, 2018.

[39] Alberto Bernacchia, Hyojung Seo, Daeyeol Lee, and Xiao-Jing Wang. A reservoir of time constants for memory traces in cortical neurons. Nature neuroscience, 14(3):366–372, 2011.

[40] Gustavo BM Mello, Sofia Soares, and Joseph J Paton. A scalable population code for time in the striatum. Current Biology, 25(9):1113–1122, 2015.

[41] Michael N Shadlen and William T Newsome. Neural basis of a perceptual decision in the parietal cortex (area lip) of the rhesus monkey. Journal of neurophysiology, 86(4):1916–1936, 2001.

[42] John Gibbon, Chara Malapani, Corby L Dale, and Charles R Gallistel. Toward a neurobiology of temporal cognition: advances and challenges. Current opinion in neurobiology, 7(2):170–184, 1997.

[43] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.

[44] Robin Cao, Alexander Pastukhov, Maurizio Mattia, and Jochen Braun. Collective activity of many bistable assemblies reproduces characteristic dynamics of multistable perception. Journal of Neuroscience, 36(26):6957–6972, 2016.

[45] David Roxbee Cox and Hilton David Miller. The theory of stochastic processes. Routledge, 2017.

[46] Herbert A Simon. Rational choice and the structure of the environment. Psychological review, 63(2):129, 1956.

[47] André G Mendonça, Jan Drugowitsch, M Inês Vicente, Eric EJ DeWitt, Alexandre Pouget, and Zachary F Mainen. The impact of learning on perceptual decisions and its implication for speed-accuracy tradeoffs. Nature communications, 11(1):1–15, 2020.

[48] Anne K Churchland, Roozbeh Kiani, and Michael N Shadlen. Decision-making with multiple alternatives. Nature neuroscience, 11(6):693–702, 2008.

[49] Chi-Tat Law and Joshua I Gold. Reinforcement learning can account for associative and perceptual learning on a visual-decision task. Nature neuroscience, 12(5):655, 2009.

[50] Nathan F Lepora. Threshold learning for optimal decision making. In Advances in Neural Information Processing Systems, pages 3763–3771, 2016.

[51] Mads Lund Pedersen, Michael J Frank, and Guido Biele. The drift diffusion model as the choice rule in reinforcement learning. Psychonomic bulletin & review, 24(4):1234–1251, 2017.

[52] Eleni Vasilaki, Nicolas Frémaux, Robert Urbanczik, Walter Senn, and Wulfram Gerstner. Spike-based reinforcement learning in continuous state and action space: when policy gradient methods fail. PLoS Comput Biol, 5(12):e1000586, 2009.

[53] Luca Manneschi, Matt OA Ellis, Guido Gigante, Andrew C Lin, Paolo Del Giudice, and Eleni Vasilaki. Exploiting multiple timescales in hierarchical echo state networks. arXiv preprint arXiv:2101.04223, 2021.

[54] Kenneth W Latimer, Jacob L Yates, Miriam LR Meister, Alexander C Huk, and Jonathan W Pillow. Single-trial spike trains in parietal cortex reveal discrete steps during decision-making. Science, 349(6244):184–187, 2015.

[55] Eric-Jan Wagenmakers and Scott Brown. On the linear relation between the mean and the standard deviation of a response time distribution. Psychological review, 114(3):830, 2007.

[56] Robert C Wilson and Anne GE Collins. Ten simple rules for the computational modeling of behavioral data. Elife, 8:e49547, 2019.

[57] Jan Drugowitsch, André G Mendonça, Zachary F Mainen, and Alexandre Pouget. Learning optimal decisions with confidence. Proceedings of the National Academy of Sciences, 116(49):24872–24880, 2019.

[58] Roozbeh Kiani, Timothy D Hanks, and Michael N Shadlen. Bounded integration in parietal cortex underlies decisions even when viewing duration is dictated by the environment. Journal of Neuroscience, 28(12):3017–3029, 2008.

[59] Miriam LR Meister, Jay A Hennig, and Alexander C Huk. Signal multiplexing and single-neuron computations in lateral intraparietal area during decision-making. Journal of Neuroscience, 33(6):2254–2267, 2013.

[60] Daniel J Amit and Nicolas Brunel. Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex. Cerebral cortex (New York, NY: 1991), 7(3):237–252, 1997.

[61] Xiao-Jing Wang. Probabilistic decision making by slow reverberation in cortical circuits. Neuron, 36(5):955–968, 2002.

[62] Xiao-Jing Wang. Decision making in recurrent neuronal circuits. Neuron, 60(2):215–234, 2008.

[63] Gerald M Edelman and Joseph A Gally. Degeneracy and complexity in biological systems. Proceedings of the National Academy of Sciences, 98(24):13763–13768, 2001.

[64] Ryan N Gutenkunst, Joshua J Waterfall, Fergal P Casey, Kevin S Brown, Christopher R Myers, and James P Sethna. Universally sloppy parameter sensitivities in systems biology models. PLoS Comput Biol, 3(10):e189, 2007.

[65] Bryan C Daniels, Yan-Jiun Chen, James P Sethna, Ryan N Gutenkunst, and Christopher R Myers. Sloppiness, robustness, and evolvability in systems biology. Current opinion in biotechnology, 19(4):389–395, 2008.

[66] Dagmara Panas, Hayder Amin, Alessandro Maccione, Oliver Muthmann, Mark van Rossum, Luca Berdondini, and Matthias H Hennig. Sloppiness in spontaneously active neuronal networks. Journal of Neuroscience, 35(22):8480–8492, 2015.

Chapter 3

Discussion

3.1 Contributions and limitations

What are the principles to define efficient representations for learning? Across all the thesis, we have tried to provide answers to this question, analysing learning from representations that are definable as the evolution of a fixed[1] dynamical system. This research for principles has been facilitated by analysing the question in the context of reservoir computing. In particular, we focused on ensembles of unconnected (Paper III) or connected (ESN, Paper I and II) leaky integrators, where the dynamical properties of the system are governed by clearly interpretable parameters. We found, without surprise, that the concepts of hierarchical connectivity structure, neuronal specialisation, and multiplicity of integration rates, can dramatically affect the ability of reservoirs, as of any neural networks, to learn. Thus, we developed techniques and models to further exploit such features. Following the order of publications reported across the thesis, the focus shifts from the concept of hierarchical connectivity (Paper I), to the introduction of sparsity and specialisation (Paper II) in reservoir responses, to the study of decision-making when a reservoir of integration times is available (Paper III). Timescales are the underlying theme of the thesis, intimately linked to the ability of a system to retain temporal information. Across all the works, the role of timescales is explicitly studied trough the values of the leakage terms adopted. In paper I and II, multiple timescales appear also through the different rates of parameters' adaptations. Indeed, learning on

[1] The only exception to this was in the case where the leakage terms were subjected to adaptation (Paper I), but such learning can be seen as a procedure to tune the hyperparameters to the tasks considered.

the leakage terms (Paper I) or on the thresholds (Paper II) needed to occur on a much lower time constant in comparison to the adaptation of the synaptic weights.

We want to provide a summary of the research works highlighting the novelties and the limitations of each paper. Particular emphasis will be given to a critical analysis of the different works.

Paper I

Novelty and Contributions

- Interpretation of the systems' behaviour and results from the point of view of timescales and connectivity structure. To the best of our knowledge, the work constitutes the first empirical study where the interplay between connectivity structure and time constants was studied and characterised for different tasks with interpretable results.
- Formulation of a learning rule on the leakage terms and application of the latter on multiple, connected or unconnected, reservoirs.
- Utilisation of benchmarks that have not been previously adopted in the field of reservoir computing and that can provide novel understanding of reservoirs' behaviour.

Critical Analysis

Online adaptation of the leakage terms can be computationally challenging in terms of convergence time. Learning has been successfully accomplished only through the adoption of a lower timescale of adaptation on α (in comparison to the output weights) and through a 'phasic' re-initialisation of the weights to avoid local minima. Thus, optimisation of the leakage terms is more computationally expensive in comparison to the case where the read-out is trained exclusively. However, the method proposed is also considerably more efficient than grid-search or evolutionary algorithms. As we will discuss in Section 3.2 *Future Works*, this difficulty in the training procedure could also be further alleviated thanks to normalisation algorithms that can prevent covariate shifts.

While the above limitation is computational and technical, it is also important to notice how the leakage term constitutes a global parameter of the network. Consequently, adaptation of parameters as α through gradient descent is not local and difficult to apply on physically defined reservoirs. Nevertheless, the approximation and the consequent learning rule proposed can be

still adopted in a mathematical model of the system and generalised to other hyperparameters, as for instance the input connectivity matrix.

Paper II

Novelty and Contributions

- Formulation of the model SpaRCe, which is composed by an initialisation and a learning phase of thresholds with 'hard' boundaries.
- Improved performance of ESNs on classification tasks and catastrophic forgetting, demonstrating the importance of specialisation and sparsity as features that can enhance the ability of the network to learn in an online fashion.

Critical Analysis

The introduction of specialised representations increases the ability of the network to memorise associations, as shown in the results on the biologically inspired task. This improved memory capacity is usually a desirable feature of the system, but can sometimes lead to overfitting. Indeed, if the memory capacity is high enough, the system has the possibility to memorise the associations on the training data without discovering relevant and transferable features. This can happen if learning is driven by nodes that are specialised because of noise in the data, and in particular when the dimensionality of the reservoir is high and disproportionate for the task considered. However, it should be possible to avoid this regime if the model is appropriately tuned in the vast majority of the cases, since the formulation of SpaRCe enables learning to rediscover the situation when thresholds are equal to zero at worst.
The algorithm can be efficiently and directly applied if the representation of the reservoir considered is continuous. In section 3.2 we will discuss possible extensions of this work, considering situations where sparse activities can lead to learning advantages.

Paper III

Novelty and Contributions

- Introduction of multiple timescales and of an explicit clock mechanism in the decision-making process.
- The model led to novel interpretations of established experimental results, for which the presence of multiple timescales appear to be necessary. Interestingly, learning is

accomplished exclusively through reward maximisation, which leads to remarkable agreements with experimental data when the reservoir of integrators has a wide range of characteristic times.

Critical Analysis

We consider the model to be elegant and abstract, but it can arguably lack of biological realism. While this does not impact the contribution of the paper, the hypothesis drawn could be tested by using a more realistic model, where multiple timescales could arise through the recurrency of a network of spiking neurons. This variation could clarify and better establish the connections among the observables studied in the model and the experimental results on the neuronal activities of LIP. In the way, the analogy between $\Delta\Sigma_{\text{right}}$ or $\Delta\Sigma_{\text{left}}$ and the LIP activities could be better established. Despite the extensive number of results provided in the paper, the model has been tested only on a two choice decision-making experiment, and further analysis could be performed for other environments.

3.2 Future Work

We believe that the works reported here have the potential of finding applications across different fields and give inspiration for future theoretical developments. This section will be divided in three parts: in the first (***Applications***), we will discuss applications of the models proposed for complex machine learning tasks and physical reservoirs; in the second (***Improvements to ESN techniques***), possible extensions of the works will be presented with the aim to overcome some limitations and generalise previous works; in the third (***Generalisation to other methods***), the conceptual results of Paper II will be exploited to define a future research project on sequential learning.

Applications. Our results suggested how sophisticated variations of ESNs can lead to good performance in problems that require multiple timescales while maintaining the high degree of generality and intepretability that distinguish them. The low computational cost of these systems makes them easily usable in architectures composed by other types of neural networks.
One promising example of application of the models developed is in Visual Place Recognition tasks (VPR) [19], where a model is asked to recognise previously visited locations from a sequence of im-

ages that have been captured during navigation. The difficulty of this problem lies on the fact that, during testing, locations are visited on different environmental conditions, and the model needs to discover features that are robust with respect to lighting, weather, and/or angular changes. Considering that datasets are acquired with a specific time frame during navigation, the signal is dynamic.

In the work [19] we proposed a network architecture in which ESNs were added after a pre-processed convolutional neural network to provide a form of working memory and capture temporal information about the navigation process. Thanks to such composed architecture, ESNs can focus on the temporal features of the problem. In [19], we demonstrated how SpaRCe and systems with multiple leakage terms can reach leading performance in a variety of VPR datasets .

Considering these results, we hypothesize that an architecture composed by a convolutional neural network and the more complex variants of ESNs proposed could also be applied in partially observable Markov decision processes. In particular, some of the methodology developed for the VPR benchmarks could also be used for Reinforcement Learning problems where an agent navigates an environment and needs to make decisions from visual input.

The understanding developed in the papers reported can also be used in the context of reservoir computing. One main challenge that such a research field is facing is the development of a methodology that can guide the tuning of the parameters that affect the properties of a specific device. This challenge is not only task-dependent, but also reservoir-dependent, and it can require a considerable amount of time to acquire an understanding on how to exploit the computation abilities of a specific reservoir. When a mathematical model of the system is available, a general solution to this problem is gradient descent optimisation. A generalisation of the learning rule proposed in paper I can be used to guide the search on the considered hyperparameters.

Improvements to ESN techniques An intriguing idea is to generalise the hierarchical architecture proposed in the first paper to a feedforward network, where each layer is composed by multiple ESNs. A specific layer would contain reservoir with different timescales, and the leakage terms of different networks can be drawn from some fat-tailed distribution as in paper III. From the analysis accomplished in the first paper, we already know that parameters as $\gamma^{(21)}$, defining the strength of interaction among two networks, and $\alpha^{(1,2)}$, are critical. While we can alleviate the

necessity of an extensive optimisation on αs by exploiting a high number of ESNs and a wide range of characteristic times, the connection strengths among two reservoirs $\gamma^{(ij)}$ needs to be carefully optimised. Optimisation of $\gamma^{(ij)}$ could provide insights on how to connect multiple reservoirs, and can be accomplished by adapting the learning rule that was used in the first paper on the leakage term. Thanks to the multiplicity of dynamical properties available at the starting condition, the system should exhibit good generalisation properties and be robust to different tasks. At the same time, instead of learning only the read-out as in a standard ESN or optimising all the connectivity structure as in another recurrent neural network [20], we would adapt the output weights, the coupling strengths $\gamma^{(ij)}$ and/or the leakage terms. Thus, the number of parameters would scale linearly with number of total nodes and quadratically with the number of ESNs considered at worst.

We want now to notice how learning of the parameters $\gamma^{(ij)}$ and/or the leakage terms of the system would probably require a lower learning rate in comparison to the output weights. Indeed, we know from the training of αs and of the thresholds θ (first and second work respectively) that different rates of adaptation can be necessary when there is a clear hierarchy among the parameters of the system. In both cases, the learning rules on the leakage terms or thresholds depended on the output weights. Consequently, the latter parameters needed to change at a faster rate to provide a clear signal in the gradients of α or θ. Moreover, in these situations initial conditions matter. In SpaRCe, the dependence on the starting conditions has been overcome by considering the starting percentile P_n as a hyperparameter. In training the αs, this problem has been alleviated through a phasic re-initialisation of the output weights. Thus, we can expect that learning of $\gamma^{(ij)}$ and/or $\alpha^{(i)}$ would be subjected to similar difficulties also in the feedforward architecture of ESNs proposed here. Nevertheless, the multiplicity of available values of αs should alleviate this problem at least in the adaptation of the latter, offering a dynamical repertoir as starting condition.

This learning difficulty is even emphasized for hierarchically connected systems, where deeper "layers" need to adapt to the statistical changes of previous ones [21]. For instance, a relatively small change of α can completely alter the statistical properties of the ESN considered, to which deeper layers or output weights need to adapt. For this reason, it could be beneficial to consider normalisation techniques that can stabilise some properties of the statistics of activities during training. As with the previous works in [21] [22], SpaRCe can be applied as a batch or "layer" normalisation technique to achieve this purpose. Indeed, after the application of SpaRCe (see definition of the

model in Paper II) the new distribution of activities $\mathbf{x}$ of the reservoir is zero at the percentile value P_n by construction, regardless of the changes in the dynamic of the ESN.

Generalisation to other methods. In Paper II, we have showed how the concept of specialised neurons can improve the ability of a model to face sequential learning problems. In that scenario, we introduced thresholds and sparse representations on the read-out level of an ESN. The algorithm proposed (SpaRCe) is simple and does not make explicit use of the task identity for learning, in contrast to other machine learning models as [23] [24] [25], defined to alleviate catastrophic forgetting. Thus, we believe that the introduction of a controllable level of sparsity can be beneficial not only for ESNs, and a possibility is to generalise SpaRCe to networks with multiple hierarchical dependencies as multi-layer perceptrons. In this case, sparsity can be controlled thanks to a normalisation mechanism (as the percentile operation defined in paper II) across the statistics of the activities of the nodes belonging to a layer. For instance, a node of the MLP can be defined as active if its value is greater than the n percent of the other neurons in the layer. In other words, Eq. 4 and 5 of paper II (Main Text) could be exploited by using the layers' statistics of a MLP. However, a possible variation of the model could be required, considering that the percentile operation used in Eq. 4 and 5 is not differentiable and that this could negatively affect backpropagation. To describe methodologies to practically perform this operation is outside the scope of the manuscript, but an idea could be to train a feedback loop with the aim of imposing the desired sparsity level in the considered layer. Once a controllable level of sparsity can be imposed, it would be possible to dynamically change the percentage of active nodes on the base of an estimate of the novelty of the data. If the last stimuli were "predictable", the sparsity level should increase to discover the minimal part of the network that can solve the task processed. If stimuli are novel and data belong to a previously unseen task, the sparsity level should decrease to recruit new neurons. In this way, the definition of sparse representations as a function of novelty, of the error function and of the layer depth could permit an indirect optimisation of the available resources in sequential learning paradigms. While the proposed methodology is based on neuronal representations, it is also beneficial to avoid adaptation of weights that were critical for the previously learnt task. This could be accomplished by computing an online estimate of Fisher information matrix [25]), which can provide a measure of the importance of the parameters for a given task.

In summary, this thesis has been a search for mathematically tractable models with intrinsic, desirable properties as multiple timescales and neuronal specialisation. The availability of such concepts, introduced through arguably simple definitions, led to improved performance, robustness with respect to different tasks, and to an increased ability to learn multiple tasks sequentially. Perhaps, these principles will be also important for the formulation of models that can exhibit more generalised forms of intelligence. In this sense, the concepts within this thesis constitute a personal starting point for future research works that focuses on defining robust models that can quickly adapt to different tasks while retaining the information learnt on previous ones.

Chapter 4

Appendix

I1. Proof of conditions for the Echo State Property

i) Let us consider two possible network activities $\mathbf{x}$ and $\mathbf{x}'$, a general input $\mathbf{s}$ and a sigmoid-like activation function as hyperbolic tangent, then the distance $d\Big(\mathcal{F}(\mathbf{x},\mathbf{s}),\mathcal{F}(\mathbf{x}',\mathbf{s})\Big)$ between the next states of the network driven by $\mathbf{s}$ from such activities can be bounded by noticing

$$\begin{aligned} d\Big(\mathcal{F}(\mathbf{x},\mathbf{s}),\mathcal{F}(\mathbf{x}',\mathbf{s})\Big) = d\Big\{(1-\alpha)\mathbf{x}+\alpha f\Big[\mathbf{W}_{\text{in}}\mathbf{s}+\mathbf{W}\mathbf{x}\Big],(1-\alpha)\mathbf{x}'+\alpha f\Big[\mathbf{W}_{\text{in}}\mathbf{s}+\mathbf{W}\mathbf{x}'\Big]\Big\} = \\ = ||(1-\alpha)(\mathbf{x}-\mathbf{x}')+\alpha f\Big[\mathbf{W}_{\text{in}}\mathbf{s}+\mathbf{W}\mathbf{x}\Big]-\alpha f\Big[\mathbf{W}_{\text{in}}\mathbf{s}+\mathbf{W}\mathbf{x}'\Big]|| \leq \\ \leq ||(1-\alpha)(\mathbf{x}-\mathbf{x}')||+||\alpha f\Big[\mathbf{W}_{\text{in}}\mathbf{s}+\mathbf{W}\mathbf{x}\Big]-\alpha f\Big[\mathbf{W}_{\text{in}}\mathbf{s}+\mathbf{W}\mathbf{x}'\Big]|| \leq \\ \leq ||(1-\alpha)(\mathbf{x}-\mathbf{x}')||+||\alpha\Big[\mathbf{W}_{\text{in}}\mathbf{s}+\mathbf{W}\mathbf{x}\Big]-\alpha\Big[\mathbf{W}_{\text{in}}\mathbf{s}+\mathbf{W}\mathbf{x}'\Big]|| = \\ = ||(1-\alpha)(\mathbf{x}-\mathbf{x}')||+||\alpha\mathbf{W}(\mathbf{x}-\mathbf{x}')|| \leq \\ \leq (1-\alpha)d(\mathbf{x},\mathbf{x}')+\alpha\Lambda d(\mathbf{x},\mathbf{x}') = (1-\alpha+\alpha\Lambda)d(\mathbf{x},\mathbf{x}') \end{aligned}$$

where, considering that $0 \leq \Lambda < 1$ and that the leakage term $0 < \alpha \leq 1$, implies $(1-\alpha+\alpha\Lambda) = \Lambda' < 1$. In the above, the activation function f is neglected (line three to four) because less or equal than its argument by construction, and the last passage exploits the definition of maximum singular value. Thus, the distance between the states $\mathbf{x}$ and $\mathbf{x}'$ shrinks by a factor $\Lambda' < 1$ at each time step, leading to the echo state property.

ii) For the null input sequence $\mathbf{s}(t) = \mathbf{0}\ \forall t$ the system is autonomous, that is its dynamics across

time can be described as $\frac{d\mathbf{x}}{dt} = g\big(\mathbf{x}(t)\big)$. In this case, it is possible to study the stability of the equilibrium point $\tilde{\mathbf{x}} = \mathbf{0}$ without loss of generality, since any other equilibrium point can be translated to $\mathbf{0}$ thanks to a simple change of variable.

The concept of asymptotic stability can be described as follows. A system is asymptotically stable if, give any $\epsilon > 0$, a $\delta(\epsilon)$ exists such that $||\mathbf{x}(0)|| < \delta \rightarrow ||\mathbf{x}(t)|| < \epsilon \; \forall t$, and $\lim_{t\to\infty} ||\mathbf{x}(t)|| = 0$. For an autonomous system, the equilibrium point $\tilde{\mathbf{x}} = \mathbf{0}$ is not asymptotically stable when the spectral radius of the system is above unity ([26]). In other words, the dynamic of the system can not be confined on an arbitrary small region centered in zero and will tend to go away from the equilibrium point. Considering this, it is now trivial to understand why the ESN property is not satisfied for the null sequence $\mathbf{s} = \mathbf{0}$ when the spectral radius is above unity. Of course, if $\mathbf{x}(0) = \mathbf{0}$ and $\mathbf{s}(0) = \mathbf{0}$, $\mathbf{x}(t) = \mathbf{0} \; \forall t$. Thus, a possible response of the system to zero external signal is the null response. However, if the initial condition is $\mathbf{x}(t) \neq \mathbf{0}$, the system will diverge from the null response. Consequently, there are two different possible responses to the null sequence given diverse initial conditions, and this violates the echo state network property.

I2. Hyperparameters in ESNs

In this section, we will consider the possible methodologies to set the hyperparameters of ESN and/or reservoirs.

i) Grid search on a subset of hyperparameters, which in computationally expensive tasks are considered independently, or applications of evolutionary algorithms on the hyperparameter space.

ii) Definition and adoption of task-independent measurements that help to interpret the reservoir behaviour and its dynamical properties. Example of these can be the memory capacity [27], quantifying the ability of the network to recall past temporal information, or the kernel generalisation, which quantifies the ability of the model to generalise to similar, but previously unseen, data [27].

iii) Utilisation of pseudo-analytical arguments with known task characteristics/requirements to limit the region of parameters to be explored, or to simply perform an educated guess. For instance, α and ρ are related to the range of timescales of the network. Their exploration

can therefore be limited in the space of values that reflect the temporal dependencies of the task (when these are known a priori).

iv) Direct gradient descent methods on the values of the hyperparameters. All the hyperparameters considered affect the dynamic of the system, thus minimisation of a cost function through exact gradient descent methods would require backpropagation through time (BPTT). BPTT unrolls all dependencies in the succession of mathematical operations that result in the definition of the output function. Given that the system is dynamically changing across time, such minimisation needs to unravel all temporal dependencies.

Learning Algorithms

Ridge Regression

Let us consider the cost function

$$\mathrm{E} = \sum_{\mathrm{i}=1}^{\mathrm{N_{data}}} \sum_{\mathrm{j}=1}^{\mathrm{K}} \left[\mathrm{W_{ij} x_j} - \tilde{\mathrm{y}} \right]^2 + \beta \sum_{\mathrm{i,j}} \mathrm{W_{ij}^2} \tag{4.1}$$

where with W we refer to the output weights $\mathrm{W_{out}}$, neglecting the subscript for convenience and β is the scaling factor of the L2 penalty term. As in the main text (Introduction), N is the number of reservoir nodes and K the output dimensionality, while $\mathrm{N_{data}}$ corresponds to the number of samples. The above error function can be written in matrix notation for the case with one output node as

$$\mathrm{E} = \left(\mathbf{W X} - \tilde{\mathbf{Y}} \right) \left(\mathbf{W X} - \tilde{\mathbf{Y}} \right)^{\mathrm{T}} + \beta^2 \mathbf{W W}^{\mathrm{T}} \tag{4.2}$$

where $\mathbf{X}$ and $\tilde{\mathbf{Y}}$ are $\mathrm{N} \times \mathrm{N_{data}}$ and $\mathrm{K} \times \mathrm{N_{data}}$ ($\mathrm{K} = 1$ for one output node) matrices obtained through concatenation (across columns) of the network activities and the desired outputs respectively. The

algorithm can be found by setting to zero the gradient of the cost function as follows

$$\begin{aligned}
\nabla_{\mathbf{W}}\Big(\mathbf{W}\mathbf{X}-\tilde{\mathbf{Y}}\Big)\Big(\mathbf{W}\mathbf{X}-\tilde{\mathbf{Y}}\Big)^{\mathrm{T}}+\nabla_{\mathbf{W}}\beta^2\mathbf{W}\mathbf{W}^{\mathrm{T}}=0\\
\nabla_{\mathbf{W}}\Big(\mathbf{W}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}-\mathbf{W}\mathbf{X}\tilde{\mathbf{Y}}^{\mathrm{T}}-\tilde{\mathbf{Y}}\mathbf{X}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}+\tilde{\mathbf{Y}}\tilde{\mathbf{Y}}^{\mathrm{T}}\Big)+2\beta\mathbf{W}=\\
=\mathbf{W}\mathbf{X}\mathbf{X}^{\mathrm{T}}+\big(\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\big)^{\mathrm{T}}-\big(\mathbf{X}\tilde{\mathbf{Y}}^{\mathrm{T}}\big)^{\mathrm{T}}-\tilde{\mathbf{Y}}\mathbf{X}^{\mathrm{T}}+2\beta\mathbf{W}=\\
=\mathbf{W}\mathbf{X}\mathbf{X}^{\mathrm{T}}+\mathbf{W}\mathbf{X}\mathbf{X}^{\mathrm{T}}-\tilde{\mathbf{Y}}\mathbf{X}^{\mathrm{T}}-\tilde{\mathbf{Y}}\mathbf{X}^{\mathrm{T}}+2\beta\mathbf{W}=0\\
\rightarrow\mathbf{W}\Big(\mathbf{X}\mathbf{X}^{\mathrm{T}}+\beta\mathbf{I}_{\mathrm{N}}\Big)=\tilde{\mathbf{Y}}\mathbf{X}^{\mathrm{T}}\\
\rightarrow\mathbf{W}=\tilde{\mathbf{Y}}\mathbf{X}^{\mathrm{T}}\Big(\mathbf{X}\mathbf{X}^{\mathrm{T}}+\beta\mathbf{I}_{\mathrm{N}}\Big)^{-1}
\end{aligned}\tag{4.3}$$

where $\mathbf{I}_{\mathrm{N}}$ is the identity matrix and Eq.4.3 corresponds to the ridge regression algorithm. We notice how Eq. 4.3 holds trivially also for the case where $\mathrm{K}>1$. The purpose of the penalty term is to avoid large values of the output weights $\mathbf{W}$ that would render the solution susceptible to small variations of the data. The consequence is that, in particular for $\beta\rightarrow 0$, $\mathbf{W}$ can overfit the training data and that the solution can not generalise to the test set.

Bibliography

[1] Michiel Hermans and Benjamin Schrauwen. Recurrent kernel machines: Computing with infinite echo state networks. *Neural Computation*, 24(1):104–133, 2012.

[2] Hoger Amin and Andrew C Lin. Neuronal mechanisms underlying innate and learned olfactory processing in drosophila. *Current opinion in insect science*, 36:9–17, 2019.

[3] Mehrab N Modi, Yichun Shuai, and Glenn C Turner. The drosophila mushroom body: from architecture to algorithm in a learning circuit. *Annual review of neuroscience*, 43:465–484, 2020.

[4] Andrew C Lin, Alexei M Bygrave, Alix De Calignon, Tzumin Lee, and Gero Miesenböck. Sparse, decorrelated odor coding in the mushroom body enhances learned odor discrimination. *Nature neuroscience*, 17(4):559, 2014.

[5] Herbert Jaeger. The echo state approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.

[6] Thomas Natschläger, Wolfgang Maass, and Henry Markram. The" liquid computer": A novel strategy for real-time computing on time series. *Telematik*, 8(ARTICLE):39–43, 2002.

[7] Christian Klos, Yaroslav Felipe Kalle Kossio, Sven Goedeke, Aditya Gilra, and Raoul-Martin Memmesheimer. Dynamical learning of dynamics. *Physical Review Letters*, 125(8):088103, 2020.

[8] Izzet B Yildiz, Herbert Jaeger, and Stefan J Kiebel. Re-visiting the echo state property. *Neural networks*, 35:1–9, 2012.

[9] Herbert Jaeger. *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*, volume 5. GMD-Forschungszentrum Informationstechnik Bonn, 2002.

[10] Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural networks*, 20(3):335–352, 2007.

[11] Mantas Lukoševičius. A practical guide to applying echo state networks. In *Neural networks: Tricks of the trade*, pages 659–686. Springer, 2012.

[12] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Jochen J Steil. Backpropagation-decorrelation: online recurrent learning with o (n) complexity. In *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, volume 2, pages 843–848. IEEE, 2004.

[15] David Sussillo and Larry F Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557, 2009.

[16] Eleftheria Vrontou, Lukas N Groschner, Susanne Szydlowski, Ruth Brain, Alina Krebbers, and Gero Miesenböck. Response competition between neurons and antineurons in the mushroom body. *Current Biology*, 2021.

[17] Roger Ratcliff and Philip L Smith. A comparison of sequential sampling models for two-choice reaction time. *Psychological review*, 111(2):333, 2004.

[18] Jan Drugowitsch, Rubén Moreno-Bote, Anne K Churchland, Michael N Shadlen, and Alexandre Pouget. The cost of accumulating evidence in perceptual decision making. *Journal of Neuroscience*, 32(11):3612–3628, 2012.

[19] Anil Ozdemir, Andrew B Barron, Andrew Philippides, Michael Mangan, Eleni Vasilaki, and Luca Manneschi. Echovpr: Echo state networks for visual place recognition. *arXiv preprint arXiv:2110.05572*, 2021.

[20] Sarath Chandar, Chinnadhurai Sankar, Eugene Vorontsov, Samira Ebrahimi Kahou, and Yoshua Bengio. Towards non-saturating recurrent units for modelling long-term dependencies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3280–3287, 2019.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[22] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[23] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *Proceedings of the 35th International Conference on Machine Learning*, 80:4548–4557, 10–15 Jul 2018.

[24] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019.

[25] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[26] H. K. Khalil. *Nonlinear systems*. Prentice Hall, 2000.

[27] Matthew Dale, Julian F Miller, Susan Stepney, and Martin A Trefzer. A substrate-independent framework to characterize reservoir computers. *Proceedings of the Royal Society A*, 475(2226):20180723, 2019.