Cognitive Computation (2020) 12:330–356
https://doi.org/10.1007/s12559-019-09634-2

Interpreting Recurrent Neural Networks Behaviour via Excitable Network Attractors

Andrea Ceni[1] · **Peter Ashwin**[2] · **Lorenzo Livi**[1,3]

Received: 3 August 2018 / Accepted: 5 March 2019 / Published online: 23 March 2019

Abstract

Machine learning provides fundamental tools both for scientific research and for the development of technologies with significant impact on society. It provides methods that facilitate the discovery of regularities in data and that give predictions without explicit knowledge of the rules governing a system. However, a price is paid for exploiting such flexibility: machine learning methods are typically black boxes where it is difficult to fully understand what the machine is doing or how it is operating. This poses constraints on the applicability and explainability of such methods. Our research aims to open the black box of recurrent neural networks, an important family of neural networks used for processing sequential data. We propose a novel methodology that provides a mechanistic interpretation of behaviour when solving a computational task. Our methodology uses mathematical constructs called excitable network attractors, which are invariant sets in phase space composed of stable attractors and excitable connections between them. As the behaviour of recurrent neural networks depends both on training and on inputs to the system, we introduce an algorithm to extract network attractors directly from the trajectory of a neural network while solving tasks. Simulations conducted on a controlled benchmark task confirm the relevance of these attractors for interpreting the behaviour of recurrent neural networks, at least for tasks that involve learning a finite number of stable states and transitions between them.

Keywords Recurrent neural networks · Dynamical systems · Network attractors · Bifurcations

Introduction

Artificial recurrent neural networks (RNNs) are widely used to solve tasks involving temporal data, e.g. speech [18] and handwriting recognition [44], audio classification [26, 52] or time series forecasting [8]. RNNs are characterised by the presence of feedback connections in a hidden layer, which allows generating a state–space representation that equips the network with short-term memory capability. RNNs are universal approximators of dynamical systems [14, 19], meaning that, given enough neurons in the hidden layer, it is possible to fine-tune the weights to achieve any desired level of accuracy. Nevertheless, training via back-propagation through time is difficult due to the vanishing/exploding gradient problem [23, 43]. This has led to the development of new and faster techniques for training RNNs, including a different paradigm known as reservoir computing [32, 33]. Echo state networks (ESNs) [21, 30] constitute an important example of reservoir computing, where a recurrent layer (called a reservoir) is composed of a large number of neurons with randomly initialised connections that are not fine-tuned via gradient-based optimisation mechanisms. The main idea behind ESNs is to exploit the rich dynamics generated by the reservoir with an output layer, the read-out that is optimised to solve a specific task.

✉ Lorenzo Livi
lorenz.livi@gmail.com

Andrea Ceni
ac860@exeter.ac.uk

Peter Ashwin
p.ashwin@exeter.ac.uk

[1] Department of Computer Science, University of Exeter, Exeter EX4 4QF, UK

[2] Department of Mathematics, University of Exeter, Exeter EX4 4QF, UK

[3] Departments of Computer Science and Mathematics, University of Manitoba, Winnipeg, MB R3T 2N2, Canada

Problem Statement and Research Hypothesis

The high-dimensional and non-linear nature of RNNs complicates interpretability of their internal dynamics, which are characterised by complex, input-dependent spatio-temporal patterns of activity [47, 55]. This poses constraints on understanding the behaviour of RNNs: they are usually viewed as black boxes from which it is hard to extract useful knowledge about their inner workings. As highlighted by recent research efforts [10, 24, 40], similar interpretability issues affect many other machine learning methods. Furthermore, an increasing societal need to develop accountability and explainability of decision making by AI [17] is driving the development of methodologies for explaining the behaviour of such methods.

Our aim in this paper is to develop effective models that capture the essential dynamical behaviour of RNNs on computational tasks as input-driven responses of a dynamical system, while neglecting microscopic details of the RNN dynamics in phase space (i.e. the space of all possible neuron activations). To this end, we hypothesise that RNNs can undertake computations by exploiting (i) transient dynamical regimes and (ii) excitable connections to switch between different stable attractors, depending on input and current state. The RNN behaviour depends on the task at hand: for example, long transients are mostly exploited in time series forecasting problems, while switching between attractors is mostly exploited in classification problems or tasks requiring to learn a finite set of memory states. These mechanisms are not mutually exclusive and can be exploited synergistically to realise more complex computations.

Contribution and Paper Organisation

In order to test our hypothesis, we develop a theoretical framework based on network attractors of dynamical systems. An attractor is a subset of the state space of a dynamical system (i.e., the phase space), where the state will asymptotically converge for "usual" choices of initial conditions. Network attractors are special kinds of attractor which can be thought of as directed graphs, where nodes correspond to local attractors and directed edges correspond to particular trajectories that connect (or almost connect) those local attractors. Since for this paper, the local attractors of interest are all stable fixed points, we will use the term local attractor synonymously with stable fixed point. A stable fixed point is the simplest possible attractors: this is a point in phase space where all variables of the dynamical system assume constant values, and all close enough initial conditions converge to this point. However, fixed points need not be stable: they can be partially stable (saddle points) or totally unstable (repellers). According to the nature of the trajectories connecting local attractors, it is possible to consider both heteroclinic network attractors composed of heteroclinic connections between saddles (i.e. partially stable fixed points) and excitable network attractors (ENAs) composed by connections that are excitable, i.e. that require a small initial perturbation to move between stable attractors [5, 60].

Heteroclinic network attractors have already been suggested as models to describe transient computation [45, 46]; here, conversely, we focus attention on ENAs which have the advantage of being more robust to perturbations. We focus particularly on ENAs between stable fixed points [5], although ENAs can be theoretically conceived between any kinds of local attractor: limit cycles, limit tori or strange attractors. More precisely, we show how to construct ENAs describing RNN behaviour for tasks that involve switching between a finite set of attractors. Interestingly, from a directed graph (representing the underlying network attractor of a dynamical system), it is possible to reverse engineer [4] a set of ordinary differential equations ruling, in our case, the RNN behaviour. This, in turn, opens the way to a more analytic description of how RNNs solve computational tasks.

In this paper, we focus on the flip-flop task with two bits [55], since it provides a controlled workbench to test our hypothesis. The input to discrete-time RNN is assumed to be a discrete temporal sequence with values from $\{+1, 0, -1\}$. In general, we consider discrete time k varying input, denoted as $\mathbf{u}[k]$, and output, denoted as $\mathbf{z}[k]$, of a system related by

$$(\mathbf{x}[k], \mathbf{z}[k]) = \mathbf{R}(\mathbf{u}[k], \mathbf{x}[k-1]), \tag{1}$$

where $\mathbf{R}$ represents an RNN with evolving internal state, namely $\mathbf{x}[k]$, and connection weights that are optimised during the training phase. We consider response to inputs where, independently for any k, $\mathbf{u}[k]$ is $(0, 0)$ with probability $1 - p$ or otherwise chosen to be one of the four inputs $(\pm 1, 0)$ or $(0, \pm 1)$ with equal probability. This generates a sequence of inputs that remain at $(0, 0)$ for an exponentially distributed period of time, but occasionally takes one of the four values $(0, \pm 1)$, $(\pm 1, 0)$. The target output to be learned by the RNN is a two-dimensional vector $\mathbf{z}[k]$ that can assume four possible configurations: $(1, 1)$, $(1, -1)$, $(-1, 1)$ and $(-1, -1)$. The system (1) is considered to successfully accomplish the task if the network is able to reliably and accurately reproduce all 20 possible actions described in Table 1.

We use RNNs that are ESNs subjected to supervised training with a perturbation matrix obtained by injecting the output into the ESN dynamics. This choice does not limit the validity of our hypothesis: we claim that our results are general and can be used to describe the behaviour

Table 1 The two-bit flip-flop task. Depending on the output $\mathbf{z}[k-1]$ and input $\mathbf{u}[k]$, we expect the output $\mathbf{z}[k]$ to become as given in the table, where N.C. indicates "no change" from the current output value

Outputs\ Inputs	(0, 0)	(1, 0)	(−1, 0)	(0, 1)	(0, −1)
(1, 1)	N.C.	N.C.	(−1, 1)	N.C.	(1, −1)
(1, −1)	N.C.	N.C.	(−1, −1)	(1, 1)	N.C.
(−1, 1)	N.C.	(1, 1)	N.C.	N.C.	(−1, −1)
(−1, −1)	N.C.	(1, −1)	N.C.	(−1, 1)	N.C.

of more general discrete-time, input-driven RNNs. The contributions of this paper can be summarised as follows:

- We provide a theoretical framework to describe the behaviour of RNNs by means of ENAs. The proposed theoretical framework is general and covers several types of computational tasks. We present a specific instance of such a framework applied to describe RNN behaviour on tasks requiring to learn a finite set of memory states;
- We use bifurcation analysis to manually design (low-dimensional) ESNs that give rise to ENAs able to solve the flip-flop task with any number of bits. This allows us to justify our choice of modelling framework and suggests its validity in the context of RNNs;
- As ESNs are driven by inputs and subject to training via perturbation matrices, bifurcation analysis alone is not sufficient to explain changes to their behaviour. In fact, inputs and perturbation matrices can affect ESN behaviour in a non-trivial way. To this end, we introduce an algorithm to extract the ENA describing the ESN behaviour for the task at hand. The algorithm analyses an ESN trajectory and constructs a directed graph encoding the underlying ENA on which the dynamics takes place. The vertices (nodes) of such a graph are associated with the fixed points of the dynamics and directed edges describe excitable connections between them. We apply this algorithm to an ESN that is trained to solve the flip-flop task and show that the resulting ENA is able to explain how ESNs perform computations in a detailed and mechanistic way;
- We define a notion of excitability threshold for this high-dimensional, non-linear dynamical system driven by inputs. We propose a method for computing this excitability threshold that accounts for inputs and can directly be applied to trajectories generated by a neural network while solving a task;
- Our simulations suggest three important findings. First, as already noted by recent research [1, 55], the dynamics of high-dimensional RNNs takes place in a much lower-dimensional phase space region that is determined by the structure introduced with training and inputs. Here, we observe that the dynamics is indeed low-dimensional, but highlight the fact that additional dimensions are used occasionally to switch between stable states according to control inputs. This suggests that, for example, a simplistic use of methods based on explained variance to reduce dimensions needs to be avoided. Second, we show how ENA models describing RNN behaviour can be exploited to provide a mechanistic interpretation of errors occurring during the computation undertaken by RNNs. Finally, we show how excitability thresholds of the extracted ENAs allow us to assess the robustness of RNNs to noise-induced perturbations.

The remainder of this paper is organised as follows. The section "Background" introduces the essential background material needed in this paper. The section "Designing Low-Dimensional ESNs to Solve Flip-Flop Tasks" shows how to design low-dimensional ESN models that give rise to ENAs able to solve the flip-flop task. In "Extracting ENAs from the ESN Trajectory", we propose an algorithm for automatically extracting ENAs directly from an ESN trajectory generated while solving a task. In "Simulations", we present and discuss results of the simulations. Finally, the section "Conclusions" draws conclusions and points to future research directions. We include three appendices: Appendix A reviews notions of linear stability used throughout the paper. Appendix B discuss bifurcations of fixed points for low-dimensional ESN maps. Appendix C provides details of the procedure used to determine fixed points.

Background

Echo State Networks

We consider a specific system of the form (1), corresponding to a discrete-time ESN state-update and related output:

$$\mathbf{x}[k] = \phi(\mathbf{W}_r\mathbf{x}[k-1] + \mathbf{W}_{in}\mathbf{u}[k]), \tag{2}$$

$$\mathbf{z}[k] = \mathbf{W}_o\mathbf{x}[k]. \tag{3}$$

$\mathbf{x}[k] \in \mathbb{R}^{N_r}$ is the state, $\mathbf{u}[k] \in \mathbb{R}^{N_i}$ and $\mathbf{z}[k] \in \mathbb{R}^{N_o}$ denote input and output, respectively. The activation function $\phi(\cdot)$ is applied component-wise; without loss of generality, we consider $\phi = \tanh : \mathbb{R} \longrightarrow (-1, 1)$. It is worth mentioning [43] that Eq. 2 is often written $\mathbf{x}[k] = \mathbf{W}_r\phi(\mathbf{x}[k-1]) + \mathbf{W}_{in}\mathbf{u}[k]$. However, [37] proved that the two formulations are equivalent up to a change of coordinates; they produce the same discrete-time dynamics. In this paper, we build on Eq. 2 and consider a network of discrete-time leaky-integrator neurons [22] of the form:

$$\mathbf{x}[k] = (1-\alpha)\mathbf{x}[k-1]+\alpha\phi(\mathbf{W}_r\mathbf{x}[k-1]+\mathbf{W}_{in}\mathbf{u}[k]+\boldsymbol{\epsilon}). \quad (4)$$

Here, $\alpha \in (0, 1]$ is called a leak rate and explicitly sets the timescale of the ESN [56]. The $\boldsymbol{\epsilon}$ term represents additive white Gaussian noise with spherical covariance matrix and unit standard deviation.

The reservoir $\mathbf{W}_r \in \mathbb{R}^{N_r \times N_r}$ and input $\mathbf{W}_{in} \in \mathbb{R}^{N_r \times N_i}$ matrices are usually random with i.i.d. entries drawn from uniform or Gaussian distributions [31, 32]. However, in the literature, it is possible to find reservoirs with different connection patterns, including deterministic topologies [50] and those exploiting the norm-preserving property of orthogonal matrices [36]. In our case, the read-out matrix $\mathbf{W}_o \in \mathbb{R}^{N_o \times N_r}$ is optimised for the task at hand. Relevant hyperparameters directly affecting ESN performance include the number of neurons and sparseness of their connections, the spectral radius of the reservoir matrix, and leak rate α [9, 29]. The so-called echo-state property (ESP) [15, 34, 62] guarantees the existence and uniqueness of a global attracting trajectory for any input sequence in a compact set. The ESP, although useful in some tasks like forecasting tasks, is in practice difficult to verify and it is usually formulated only for ESNs with state-update of the form shown in Eqs. 2 and 4. Therefore, it is not suitable to ESN models and tasks we discuss here (see the following subsection).

Training ESNs with Low-Rank Perturbation Matrices

Training of RNNs is typically implemented by means of stochastic gradient descent or variations of thereof [51]. Learning long-term dependencies in RNNs with gradient descent is known to be problematic, as a consequence of the so-called vanishing/exploding gradient problem [43]. To this end, different approaches have been proposed that can be summarised in two categories: (i) methods using gating mechanisms (such as Long Short-Term Memory [20] and Gated Recurrent Unit [12] networks) and (ii) those based on unitary matrices and constant-slope activations [3]. On the other hand, training of the recurrent layer in ESNs is typically realised by perturbing a randomly initialised reservoir with a low-rank, deterministic matrix. This is conventionally accomplished by feeding back the ESN output to the recurrent layer [27, 48, 49, 59, 61] or, as [35] recently proposed, by designing the reservoir directly as $\mathbf{W}_r = \mathbf{X} + \mathbf{D}$, where $\mathbf{X}$ is a random matrix and $\mathbf{D}$ is a deterministic, low-rank matrix encoding the task of interest.

In this paper, we use a supervised learning algorithm which exploits the feedback of the ESN output as a mechanism for training the recurrent layer. The state-update (4) takes the following form:

$$\mathbf{x}[k] = (1-\alpha)\mathbf{x}[k-1]+\alpha\phi(\mathbf{W}_r\mathbf{x}[k-1]+\mathbf{W}_{in}\mathbf{u}[k]+\mathbf{W}_{fb}\mathbf{y}[k-1]+\boldsymbol{\epsilon}), \quad (5)$$

where $\mathbf{W}_{fb} \in \mathbb{R}^{N_r \times N_o}$ is a matrix with i.i.d. random coefficients usually drawn from a uniform or Gaussian distribution. Depending on whether training is performed batch or online $\mathbf{y}[k-1]$ in Eq. 5 takes the form of either the target signal or the output produced by the ESN (3), respectively. In batch mode, it is possible to distinguish two main phases (see Fig. 1 for an illustration). First, the reservoir is fed with an auxiliary input, i.e. the target signal $\mathbf{y}$. We construct a matrix $\mathbf{X} \in \mathbb{R}^{N \times N_r}$ containing the states $\mathbf{x}[k]$ of the ESN generated in response to target and input signals. Finally, the weights $\mathbf{W}_o$ of the read-out are determined by solving a regularised least-squares problem, $\mathbf{W}_o = \left(\left(\mathbf{X}^\top\mathbf{X} + \lambda^2\mathbf{I}\right)^{-1}\mathbf{X}^\top\mathbf{y}\right)^\top$, where $\mathbf{I}$ is an $N_r \times N_r$ identity matrix and $\lambda \geq 0$ is a regularisation parameter. Successively, the target signal is replaced by the generated output $\mathbf{z}$; this "closed-loop phase" corresponds to the test phase of the trained ESN. An analysis of the stability of the transition from open- to closed-loop can be found in [49].

Definition 1 We call the *trained reservoir* the following matrix

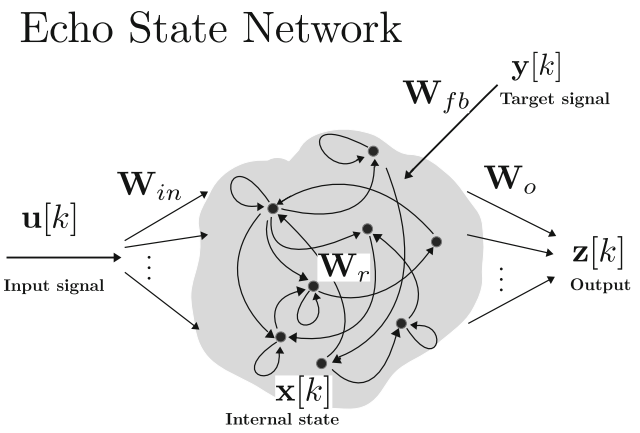
$$\mathbf{M} := \mathbf{W}_r + \mathbf{W}_{fb}\mathbf{W}_o. \quad (6)$$

Once the read-out matrix $\mathbf{W}_o$ is optimised, by imposing $\mathbf{y}[k] = \mathbf{z}[k]$ and expanding in Eq. 5 with Eq. 3, we obtain:

$$\begin{aligned}\mathbf{x}[k] = {} & (1-\alpha)\mathbf{x}[k-1] + \alpha\phi(\mathbf{W}_r\mathbf{x}[k-1] + \mathbf{W}_{in}\mathbf{u}[k] \\ & +\mathbf{W}_{fb}\mathbf{W}_o\mathbf{x}[k-1] + \boldsymbol{\epsilon})(1-\alpha)\mathbf{x}[k-1] \\ & +\alpha\phi(\mathbf{M}\mathbf{x}[k-1] + \mathbf{W}_{in}\mathbf{u}[k] + \boldsymbol{\epsilon}). \end{aligned} \quad (7)$$

Remark The trained reservoir (6) is obtained by adding a matrix $\mathbf{W}_{fb}\mathbf{W}_o \in \mathbb{R}^{N_r \times N_r}$, which is low-rank $N_o \ll N_r$ relative to the randomly initialised reservoir matrix $\mathbf{W}_r$. Therefore, the reservoir is in some sense trained using output feedback connections.

Inputs $\mathbf{u}[k]$ in Eq. 7 play the role of control inputs and are typically constant or impulsive signals. The ESN read-out matrix $\mathbf{W}_o$ is conventionally determined by solving



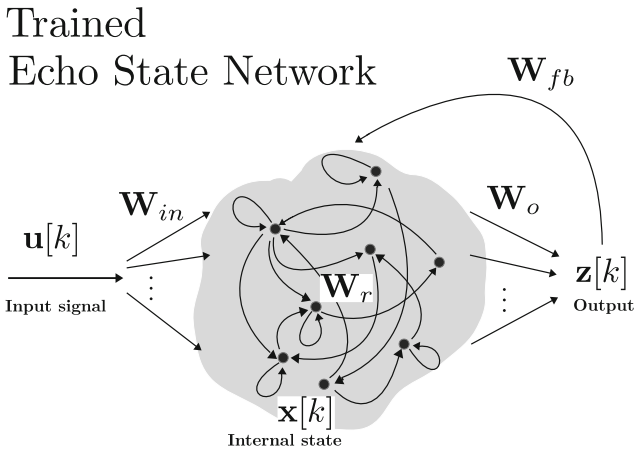


Fig. 1 **Left** Illustration of a ESN in the open-loop (training) phase (5). Training is supervised by means of the target signal **y**. **Right** Illustration of an ESN (7) after the optimisation of the read-out matrix. Feeding back the output signal into the reservoir gives the closed-loop (trained) system which self-sustains the dynamics driven by the input signal

a regularised least-squares problem. Nonetheless, we note that also online training schemes have been developed, e.g. the FORCE learning algorithm originally introduced in [54] and further extended by [13]. During the test phase, regardless of the adopted training mechanism, the state-update of ESNs is described by Eq. 7. In this paper, we consider batch training via ridge regression and analyse the trajectory generated by Eq. 7 during the test phase.

It is worth noting that our theoretical framework does not rely on a particular training method or a particular RNN architecture. We note that complicated (trained) RNN models may be described using a noisy nonautonomous dynamical system, which in our system is represented by Eq. 7. We focus on ESNs as they are the simplest RNN models to test our hypothesis. For this reason, the terms ESN and RNN are used interchangeably.

Network Attractors

Many common dynamical systems encountered in nature are dissipative [11, 53]. In such systems, the absence of any conservation law means that typically the system evolves towards an attracting set of dimension strictly less than the original phase space dimension; such a set (or a particular subset of it) is commonly called an *attractor*: formal definitions are discussed for example in [39]. The *basin of attraction* of an attractor is the set of all initial conditions from which the system evolves toward the attractor. Attractors convey crucial information about the behaviour of the dynamical systems which have generated them.

Here, we consider the following noise-free discrete-time dynamical system with inputs:

$$\mathbf{x}[k] = \mathbf{G}(\mathbf{x}[k-1], \mathbf{u}[k]), \tag{8}$$

where $\mathbf{G} : \mathbb{R}^{N_r} \times \mathbb{R}^{N_i} \longrightarrow \mathbb{R}^{N_r}$ is related to Eq. 7 as follows:

$$\mathbf{G}(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} (1-\alpha)x_1 + \alpha\phi\left(\mathbf{M}_{(1)} \cdot \mathbf{x} + (\mathbf{W}_{in})_{(1)} \cdot \mathbf{u}\right) \\ (1-\alpha)x_2 + \alpha\phi\left(\mathbf{M}_{(2)} \cdot \mathbf{x} + (\mathbf{W}_{in})_{(2)} \cdot \mathbf{u}\right) \\ \vdots \\ (1-\alpha)x_{N_r} + \alpha\phi\left(\mathbf{M}_{(N_r)} \cdot \mathbf{x} + (\mathbf{W}_{in})_{(N_r)} \cdot \mathbf{u}\right) \end{pmatrix}, \tag{9}$$

where **M** is the trained reservoir matrix (6), the subscript (i) denotes the ith rows of a matrix and $\cdot$ the usual dot product. As mentioned before, the input signal for the flip-flop task is null most of the time, at which point it is governed by the autonomous dynamics of

$$\mathbf{x}[k] = \mathbf{F}(\mathbf{x}[k-1]) = \mathbf{G}(\mathbf{x}[k-1], \mathbf{0}). \tag{10}$$

Fixed points $\mathbf{p} \in \mathbb{R}^{N_r}$ are solutions of $\mathbf{F}(\mathbf{p}) = \mathbf{p}$. Related to the notion of attractor is the notion of *limit set* [11], thus we introduce the following

Definition 2 The *ω-limit set* of a point $\mathbf{x}_0$ under the iterated map (10) is defined by

$$\omega_{\mathbf{F}}(\mathbf{x}_0) := \bigcap_{n \in \mathbb{N}} \overline{\{\mathbf{F}^h(\mathbf{x}_0) \mid \quad h > n\}}. \tag{11}$$

Remark The ω-limit set of a point $\mathbf{x}_0$ is the set of limit points of the forward trajectory $\{\mathbf{F}^h(\mathbf{x}_0)\}_{h \in \mathbb{N}}$. For a given fixed point **p**, its basin of attraction is formed by all points $\mathbf{x} \in \mathbb{R}^{N_r}$ such that $\omega_{\mathbf{F}}(\mathbf{x}) = \{\mathbf{p}\}$. If such a fixed point is stable, then there exists a neighbourhood of **p** whose ω-limit set corresponds to this fixed point.

More complex attractors consisting of networks of invariant sets in phase space have been proposed in the literature [41, 60]. Such models found renewed interest in neuroscience [38, 58] and other fields of research, as they

provide a fundamental tool to describe dynamic processes occurring on transients that explore excitable connections. More relevant to our paper, we focus on networks of stable fixed points that are connected by excitable connections. Following [5, 6], we say that there exists an *excitable connection* for amplitude $\delta > 0$ from stable fixed point $\mathbf{p}_i$ to $\mathbf{p}_j$ whenever

$$B_\delta(\mathbf{p}_i) \cap W^s(\mathbf{p}_j) \neq \emptyset, \tag{12}$$

where $B_\delta(\mathbf{p}_i)$ stands for the closed ball centred on $\mathbf{p}_i$ with radius $\delta > 0$ and $W^s(\mathbf{p}_j) = \left\{\mathbf{x} \in \mathbb{R}^{N_r} \mid \omega_{\mathbf{F}}(\mathbf{x}) = \{\mathbf{p}_j\}\right\}$ denotes the basin of attraction[1] of the fixed point $\mathbf{p}_j$.

Definition 3 We define *excitability threshold* [5] (or just *threshold*) of the excitable connection from $\mathbf{p}_i$ to $\mathbf{p}_j$, and denote it as $\delta_{th}(\mathbf{p}_i, \mathbf{p}_j)$, the following nonnegative real number:

$$\delta_{\text{th}}(\mathbf{p}_i, \mathbf{p}_j) := \inf\{\delta > 0 \ : \ B_\delta(\mathbf{p}_i) \cap W^s(\mathbf{p}_j) \neq \emptyset\}. \tag{13}$$

Remark The quantity (13) can be informally interpreted as the fact that the fixed point $\mathbf{p}_i$ is $\delta_{\text{th}}(\mathbf{p}_i, \mathbf{p}_j)$ away from the basin of $\mathbf{p}_j$ (see Fig. 2 for a visual explanation).

Definition 4 A set $X_{\text{exc}} \subset \mathbb{R}^{N_r}$ is called an *excitable network attractor* (ENA) for amplitude $\delta > 0$ if there exists a collection of fixed points $\{\mathbf{p}_i\}_{i=1}^M$, such that

$$X_{\text{exc}}(\{\mathbf{p}_i\}_{i=1}^M, \delta) := \bigcup_{\substack{i,j=1 \\ i\neq j}}^{M} \left\{\mathbf{F}^h(B_\delta(\mathbf{p}_i))\right\}_{h\geq 0} \cap W^s(\mathbf{p}_j), \tag{14}$$

where $\mathbf{F}(B_\delta(\mathbf{p}_i)) := \{\mathbf{F}(\mathbf{x}) \mid \mathbf{x} \in B_\delta(\mathbf{p}_i)\}$ is based on Eq. 10 (see [5, 6]).

The autonomous dynamics on such a set converges to one of the stable fixed points; hence, external inputs are necessary to get interesting dynamics. If the external input is large enough, then the state will escape from the current basin of attraction and switch to a different one, until another sufficiently large input will lead to another change of basin.

The excitability threshold (13) is defined as the (Euclidean) distance between a given stable fixed point, $\mathbf{p}_i$, and the basin of attraction of another fixed point, say $\mathbf{p}_j$. Such a quantity measures the minimum distance in phase space necessary to escape from the basin of $\mathbf{p}_i$ and converge towards $\mathbf{p}_j$. Nevertheless, if the dynamical system is high dimensional, then there will be a large number of possible escaping directions that could be exploited by inputs. Therefore, excitability thresholds (13) alone may not be representative of nonautonomous systems driven by inputs. For this purpose, in order to take into account the action of inputs on the dynamics, we introduce the notion of *input-driven excitability threshold* of an excitable connection. Considering K as a compact subspace of $\mathbb{R}^{N_i}$, we define $\mathcal{G}(\mathbf{x}_0; K) := \bigcup_{\mathbf{u}\in K} \mathbf{G}(\mathbf{x}_0, \mathbf{u})$, where $\mathbf{G}$ is the function in Eq. 9 defining the nonautonomous dynamical system which describes the trained neural network. The set $\mathcal{G}(\mathbf{x}_0; K)$ contains all states reachable from $\mathbf{x}_0$ under the action of input values $\mathbf{u} \in K$. Importantly, the presence of noise has the effect to perturb away the internal state from the exact location of the stable point in the deterministic counterpart of the dynamics. Therefore, instead of $\mathcal{G}(\mathbf{p}_i; K)$, we rather observe the following set.

Definition 5 We call *local input response set* of the stable point $\mathbf{p}_i$ the subset of phase space defined by:

$$\mathcal{G}(B_r(\mathbf{p}_i); K) := \bigcup_{\mathbf{x}\in B_r(\mathbf{p}_i)} \mathcal{G}(\mathbf{x}; K), \tag{15}$$

where the radius r can be shrunk according to the amplitude of the noise and K is a subset of admissible input values for the task at hand.

Remark Continuity of $\mathbf{G}$ implies that, if $r \to 0^+$, then the monotonically decreasing sequence of sets $\{\mathcal{G}(B_r(\mathbf{p}_i); K)\}_{r\geq 0}$ converges to $\mathcal{G}(\mathbf{p}_i; K)$ regardless of $K \subset \mathbb{R}^{N_i}$. Furthermore, we note that $\mathcal{G}(B_r(\mathbf{p}_i); K)$, as a subspace of $\mathbb{R}^{N_r}$, is compact if $K \subset \mathbb{R}^{N_i}$ is compact.

If K represents all possible inputs of a particular task[2], then we can drop it and denote (15) simply as $\mathcal{G}(B_r(\mathbf{p}_i))$. Therefore, the subregion of the phase space represented by the local input response set $\mathcal{G}(B_r(\mathbf{p}_i))$ encodes the action exercised by the input when the internal state of the RNN is nearby the stable point $\mathbf{p}_i$.

Definition 6 We define *input-driven excitability threshold* of the excitable connection from $\mathbf{p}_i$ to $\mathbf{p}_j$, and denote it as $\delta_{\text{inp}}(\mathbf{p}_i, \mathbf{p}_j)$, the following nonnegative real number:

$$\delta_{\text{inp}}(\mathbf{p}_i, \mathbf{p}_j) := \inf\{\delta > 0 \ : \ \mathcal{G}(\mathbf{p}_i) \cap B_\delta(\mathbf{p}_i) \cap W^s(\mathbf{p}_j) \neq \emptyset\}. \tag{16}$$

Remark The excitability threshold in Eq. 16 has a similar meaning to the one defined in Eq. 13, although it considers only the subspace exploited by the action of inputs nearby $\mathbf{p}_i$, where the excitable connection starts from (see Fig. 2).

[1] The basin of attraction corresponds to the stable manifold whenever $\mathbf{p}_j$ is hyperbolic. This has interior if $\mathbf{p}_j$ is an attractor.

[2] For example, in the flip-flop task, we have $K = \{(0, 0), (1, 0), (-1, 0), (0, 1), (0, -1)\}$.

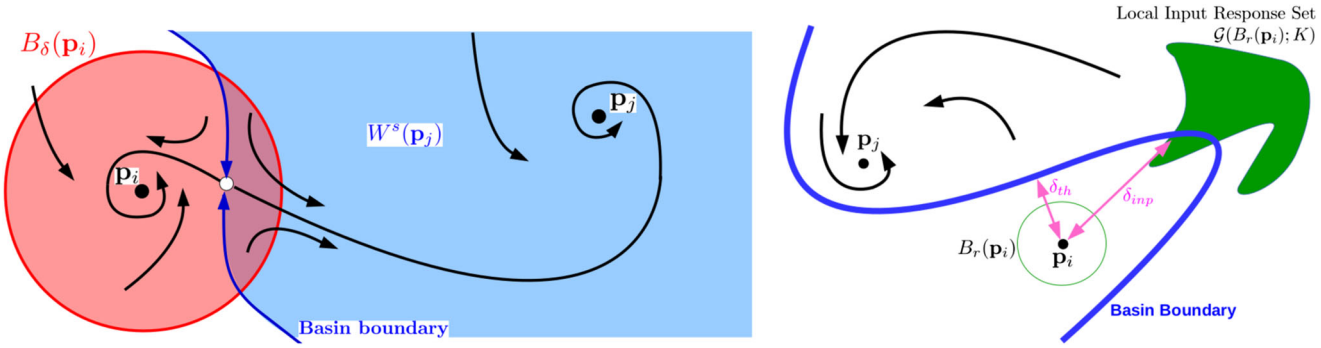


Fig. 2 **Left** Depiction of an excitable connection from $\mathbf{p}_i$ to $\mathbf{p}_j$. The red area is a closed ball centred on $\mathbf{p}_i$ of radius δ. The blue area represents the stable manifold of $\mathbf{p}_j$, i.e. its basin of attraction. The open circle represents a saddle whose stable manifold (blue curves) denotes the boundary of the basin of attraction of $\mathbf{p}_j$. Some points of $B_\delta(\mathbf{p}_i)$ converge to $\mathbf{p}_i$ itself, while those points beyond the basin boundary converge towards $\mathbf{p}_j$, as suggested by the black arrows. **Right** Representation of the activation of an excitable connection through the action of the input which allows accomplishing the switch from the stable point $\mathbf{p}_i$ to the stable point $\mathbf{p}_j$. Firstly, the internal state of the RNN stands nearby the stable point $\mathbf{p}_i$, in the neighbourhood $B_r(\mathbf{p}_i)$. Then, a control input $\mathbf{u}[k+1] \in K$ drives the current internal state $\mathbf{x}[k]$ to the next state $\mathbf{x}[k+1]$ inside the local input response set, represented as the green subregion. Finally, if the state falls beyond the basin boundary, then the internal state converges to the stable point $\mathbf{p}_j$. Excitability threshold $\delta_{\text{th}}(\mathbf{p}_i, \mathbf{p}_j)$, in Eq. 13, is computed along the direction where the distance is shortest in order to escape from the basin of attraction of $\mathbf{p}_i$ and converge to $\mathbf{p}_j$. Nevertheless, input can potentially drive the dynamics towards alternative dimensions for the purpose of achieving the switch from $\mathbf{p}_i$ to $\mathbf{p}_j$. The input-driven excitability threshold $\delta_{\text{inp}}(\mathbf{p}_i, \mathbf{p}_j)$, in Eq. 16, is computed considering the subspace exploited by the input to solve the task

Finally, from the fact that $\mathcal{G}(\mathbf{p}_i) \cap B_\delta(\mathbf{p}_i) \cap W^s(\mathbf{p}_j) \subseteq B_\delta(\mathbf{p}_i) \cap W^s(\mathbf{p}_j)$, it follows that $\delta_{\text{th}}(\mathbf{p}_i, \mathbf{p}_j) \leq \delta_{\text{inp}}(\mathbf{p}_i, \mathbf{p}_j)$ holds for all pairs of fixed points.

Designing Low-Dimensional ESNs to Solve Flip-Flop Tasks

In this section, starting from ESN models, we show how to manually design ENAs that realise computations needed for the flip-flop task. This step allows us to justify the choice of the modelling framework adopted here and its validity in the context of RNNs. For this purpose, we rely on the bifurcation theory (see Appendix B for technical notions). A bifurcation [28] is a qualitative change in the solution set (phase space topology) on changing a parameter of a dynamical system. On varying the parameters of the model, if such qualitative changes appear, then we say the system has undergone a bifurcation. For this reason, the notion of bifurcation plays an important role in training RNNs and in describing their behaviour. Training the recurrent layer of RNNs corresponds to shaping their phase space topology, possibly inducing bifurcations that lead to a qualitative change in behaviour. We argue that adding a low-rank perturbation matrix to the reservoir (6) can induce bifurcations that lead to the creation of attracting regions in ESN phase space useful to solve the task at hand. Clearly, this can also happen with more sophisticated training algorithms. Let us assume the origin as the only global attractor for the autonomous system $\mathbf{x}[k] = (1-\alpha)\mathbf{x}[k-1]+\alpha\phi(\mathbf{W}_r\mathbf{x}[k-1])$ associated with Eq. 5. Then, the bifurcation induced by Eq. 6 leads to a transition from such a trivial dynamic to one where the origin becomes unstable[3] and repels trajectories towards other attracting regions in phase space, where the nonautonomous dynamics actually takes place.

In "A Minimal-Dimension Example to Solve the Two-Bit Flip-Flop Task", we provide a low-dimensional example of an ESN that is able to solve the flip-flop task; the reservoir of such ESN is formed by two neurons. In "A $2k$-Dimensional Model for k-Bit Flip-Flop Tasks", instead, we design an ESN model with a reservoir formed by $2k$ neurons which is able to solve the flip-flop task with k bits. For both examples, we show there is an underlying ENA that explains their dynamics.

A Minimal-Dimension Example to Solve the Two-Bit Flip-Flop Task

In order to solve the k-bit flip-flop task, one needs to learn 2^k memory states (stable fixed points) and the related switching patterns dictated by control inputs. Here, we show how to design an ESN with two neurons in the recurrent layer, giving rise to an ENA able to solve the flip-flop task with $k = 2$ bits. According to the analysis of Appendix B, we know that it is possible to obtain, with k neurons, up to 3^k fixed points, 2^k of which are stable, $3^k - 2^k - 1$ are saddles and 1 (the origin) is a repeller. Therefore, we set the trained reservoir weights (6) according to condition (43). In in particular,

$$\mathbf{M}(b) := \omega_r \begin{bmatrix} 1 & b \\ b & 1 \end{bmatrix} \tag{17}$$

[3] The origin could in principle remain stable and other attractors appear elsewhere: an example can be found in [62].

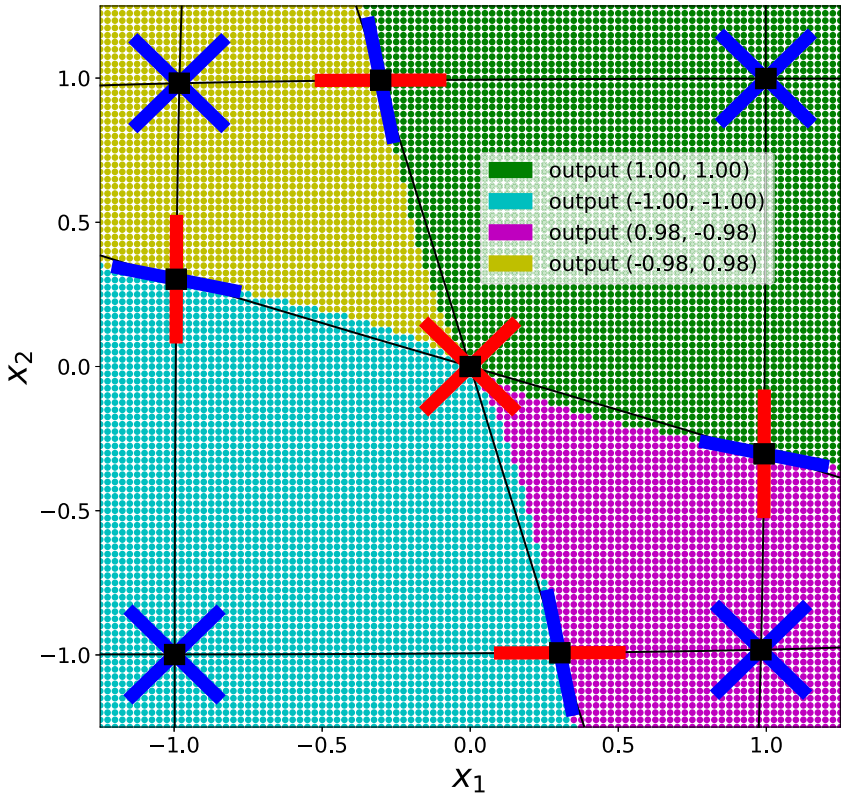


Fig. 3 Basins of attraction, nullclines, fixed points and corresponding eigenvectors of the linearised two-dimensional system (17), with $b = 0.2$. Black curves represent the nullclines (see Appendix B) whose reciprocal intersections determine fixed points (black squares). Red segments indicate eigenvectors of the linearised system for real eigenvalues larger than 1. Blue line segments represent eigenvectors of real eigenvalues in (0, 1). Particularly important are blue lines of saddles, which represent local linear approximations of the boundary of the basins of attraction. The whole phase space is divided into four basins of attraction associated to the four stable points and their boundaries. These boundaries between these basins coincide with the stable manifolds of the saddles. The legend shows output values produced at every attractor, which, in this specific example, correspond with the internal state, i.e. the phase space coordinates of the attractors. Points on the plane have been coloured according to the attractor to which they apparently converged to after 100 steps

with $\omega_r = 3$ scaling the reservoir weights, and b acting as tuning parameter. As shown in Fig. 3, for every $0 \leq b < 0.47$, the autonomous system $\mathbf{x}[k] = \tanh(\mathbf{M}(b)\mathbf{x}[k-1])$ has four stable attractors located near the vertices of the invariant square $[-1, 1]^2$.

The input is injected into the ESN via the following weight matrix, $\mathbf{W}_{in} = \omega_{in}\mathbf{I}_2$, i.e. a scaled 2×2 identity matrix, setting the scaling factor (called a hyperparameter in machine learning) to $\omega_{in} = 6$. Let us set the remaining parameters in Eq. 7 as $\boldsymbol{\epsilon} = 0$ and $\alpha = 1$, and let the ESN output (3) be the identity mapping. Let us assume that, at time k, the system is in state $\mathbf{p} = (p_1, p_2)$, which is close to one of the four attractors, i.e. $\mathbf{p}_1 \approx (1, 1)$, $\mathbf{p}_2 \approx (-1, 1)$, $\mathbf{p}_3 \approx (-1, -1)$, $\mathbf{p}_4 \approx (1, -1)$. The possible, non-null inputs at time-step k are $\mathbf{u}[k] \in \{(1, 0), (0, 1), (-1, 0), (0, -1)\}$. The action of the input pulse is encoded in a vector $\Delta \in [-1, 1]^2$, representing the difference between the state before and after the occurrence of such a pulse, whose components are

$$\Delta_j \approx \tanh\left(3p_j + 6u_j\right) - \tanh\left(3p_j\right), \quad j = 1, 2, \tag{18}$$

considering $\omega_r = 3$, $\omega_{in} = 6$ and, for the sake of simplicity, $b = 0$, which corresponds to the case of two independent neurons. Hence, the switching mechanism between attractors is ruled by the signs of both p_j and u_j; the former encoding the position and the latter the direction where to move. Therefore, the state changes if and only if p_j and u_j assume different signs for some $j \in \{1, 2\}$. In fact, if they have the same sign, then Eq. 18 is null because $\tanh\left(\text{sgn}(p_j)[3 + 6]\right) \approx \tanh\left(\text{sgn}(p_j)3\right)$ due to saturating activation functions. While, if they have different signs, then Eq. 18 becomes close to -1 or 1, according to $\text{sgn}(p_j)$, because $\tanh\left(\text{sgn}(p_j)[3 - 6]\right) = -\tanh\left(\text{sgn}(p_j)3\right)$. Clearly, it is not necessary that $|\omega_r - \omega_{in}| = \omega_r$ holds, as in our set up. Although we assumed $b = 0$ for clarity of explanation, the conclusion is the same for each $b \in [0, 0.47)$. However, when the parameter approaches the bifurcation value $b \approx 0.47$, the escaping dynamics from certain fixed points become significantly slower.

Given a set of stable fixed points, a $\delta > 0$ gives rise to a specific ENA (see definition of ENA in Eq. 14). For

instance, a large δ will potentially activate all excitable connections between the attractors. However, in order to properly solve the flip-flop task, some of the connections need not to be active, namely the diagonal connections between $\mathbf{p}_1$ and $\mathbf{p}_3$, and between $\mathbf{p}_2$ and $\mathbf{p}_4$. Due to symmetry, there are only three different excitability thresholds that an excitable connection can have, that is $\delta_{th}(\mathbf{p}_2, \mathbf{p}_1), \delta_{th}(\mathbf{p}_1, \mathbf{p}_2), \delta_{th}(\mathbf{p}_1, \mathbf{p}_3)$. In the particular configuration shown in Fig. 3, it holds that $\delta_{th}(\mathbf{p}_2, \mathbf{p}_1) < \delta_{th}(\mathbf{p}_1, \mathbf{p}_2) < \delta_{th}(\mathbf{p}_1, \mathbf{p}_3)$. Therefore, the underlying ENA supporting the nonautonomous ESN dynamics is defined as $X_{\text{exc}}(\{\mathbf{p}_i\}_{i=1}^4, \delta)$, for every $\delta_{th}(\mathbf{p}_1, \mathbf{p}_2) < \delta < \delta_{th}(\mathbf{p}_1, \mathbf{p}_3)$. We note that $\delta_{th}(\mathbf{p}_1, \mathbf{p}_3) \approx \sqrt{2}$, regardless of $b \in [0,\ 0.47)$. We can reduce the size of the basin of attraction of $\mathbf{p}_2$ and $\mathbf{p}_4$ by increasing the value of b, which in turn reduces the excitability threshold of the corresponding connections, until at $b \approx 0.47$ a bifurcation occurs making them disappear[4]. However, the basins of attraction of the other two stable points, $\mathbf{p}_1$ and $\mathbf{p}_3$, become bigger ($\delta_{th}(\mathbf{p}_1, \mathbf{p}_2) \approx 2 - \delta_{th}(\mathbf{p}_2, \mathbf{p}_1)$), so increasing their excitability thresholds. Therefore, when the ESN state is close to $\mathbf{p}_1$ (or $\mathbf{p}_3$), the input needs to be very precise to throw back the state to the narrow basins of $\mathbf{p}_2$ and $\mathbf{p}_4$. Moreover, it must have a large amplitude compared to what is needed to escape from such narrow basins. Nevertheless, setting ω_{in} large enough and depending on b (until a value of $\omega_{in} = 6$, which is enough for every $0 < b < 0.47$), the system is able to reproduce the flip-flop dynamics without errors.

Unfortunately, it does not seem to be possible to design a two-dimensional ESN for the two-bit flip-flop where the excitability of each attractor can be tuned by changing the location of the nearby saddles. Nevertheless, as we will show in the next section, this becomes possible by including two additional dimensions in the model.

A 2*k*-Dimensional Model for *k*-Bit Flip-Flop Tasks

In this section, we propose a $2k$-dimensional model able to solve k-bit flip-flop tasks. For the sake of clarity, but without loss of generality, we show the $k = 2$ bit case. The key insight is to model the switching dynamics between two fixed points in a two-dimensional space and then build a model formed by two independent systems with two-dimensional switching dynamics.

Unlike the previous example, here we want to tune the excitability of all connections. This can be obtained by imposing the condition in Eq. 42 and tuning the reservoir weights to change the position of the saddles. Therefore, we define

$$\mathbf{B} := \begin{bmatrix} 1.1 & 4 \\ -s & 4 \end{bmatrix}, \tag{19}$$

where s is a real parameter. Hence, for $0 \leq s < 2.15$, the autonomous dynamical system defined by $\mathbf{x}[k] = \tanh(\mathbf{B}\mathbf{x}[k-1])$ has one repeller (the origin), two attractors, namely $\mathbf{p}_1 \approx (1, 1)$, $\mathbf{p}_2 \approx (-1, -1)$, and two saddles. By increasing s within the $[0, 2.15)$ interval, we can make the saddles closer to the respective stable points (see Fig. 4), hence decreasing the excitability thresholds of these attractors, until a saddle-node bifurcation occurs approximately at $s = 2.15$, annihilating attractors with saddles.

Let us define the input matrix as follows:

$$\mathbf{W}_{in} := \omega_{in} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix},$$

where ω_{in} is a positive real parameter. For instance, setting $s = 2$ and $\omega_{in} = 1$, the two-dimensional dynamical system defined by $\mathbf{x}[k] = \tanh(\mathbf{B}\mathbf{x}[k-1] + \mathbf{W}_{in}\mathbf{u}[k])$ is able to accomplish the switching mechanism between $\mathbf{p}_1$ and $\mathbf{p}_2$ according to inputs $(1, 0)$, $(-1, 0)$, and ignore other inputs, i.e. $(0, 1)$, $(0, -1)$. Of course, replacing zeros in $\mathbf{W}_{in}$ with relatively small values (compared to ω_{in}) does not change the results.

The complete system able to solve the two-bit flip-flop dynamics can be obtained by defining the reservoir as the following four-dimensional block diagonal matrix,

$$\mathbf{M} := \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{bmatrix}, \tag{20}$$

where $\mathbf{0}$ denotes a 2×2 matrix with all zeros. Starting from a given initial condition, for every $0 \leq s < 2.15$, the state of the four-dimensional autonomous dynamical system $\mathbf{x}[k] = \tanh(\mathbf{M}\mathbf{x}[k-1])$ converges to one of these four attractors $(\mathbf{p}_1, \mathbf{p}_1)$, $(\mathbf{p}_1, \mathbf{p}_2)$, $(\mathbf{p}_2, \mathbf{p}_1)$, $(\mathbf{p}_2, \mathbf{p}_2)$. In order to produce a two-dimensional output (3) suitable for the task at hand, we define $\mathbf{W}_o := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$, which basically corresponds to a projection onto the first and third component, i.e. $(x_1, x_2, x_3, x_4) \longmapsto (x_1, x_3)$. Finally, we define the input matrix as:

$$\mathbf{W}_{in} := \omega_{in} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

[4] A bifurcation where both saddles collide with the corresponding stable points $\mathbf{p}_2$ and $\mathbf{p}_4$, simultaneously annihilating all six fixed points.

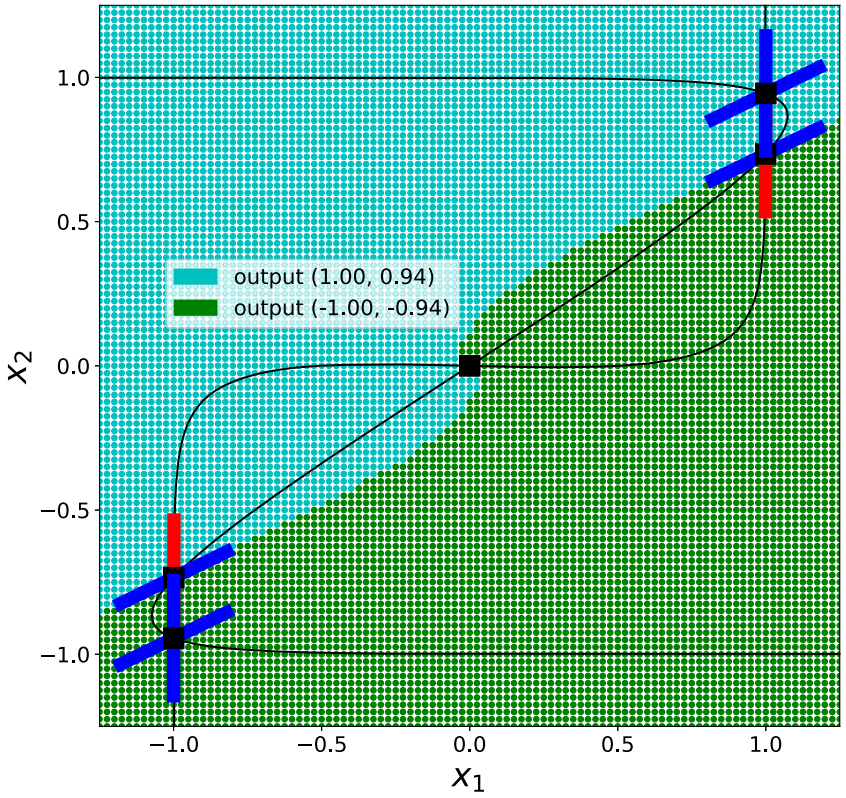


Fig. 4 Basins of attraction, nullclines, fixed points and eigenvectors of the linearised two-dimensional autonomous system having (19) as reservoir matrix, with $s = 2$. Black curves represent the nullclines (see Appendix B) whose reciprocal intersections determine fixed points (black squares). Red segments indicate real eigenvectors associated with real eigenvalues larger than 1. Blue line segments represent real eigenvectors of real eigenvalues in (0, 1). Points of the plane have been coloured according with the attractor on which they converged after 100 steps

We considered the case of a four-dimensional system composed by two independent two-dimensional systems. Nevertheless, the dynamics remains qualitatively the same even if the coupling between these two-dimensional systems is weak, i.e. if the zero matrices in Eq. 20 are replaced by matrices with relatively small (absolute) values. With those definitions in mind and, as before, $\epsilon = 0$ and $\alpha = 1$, the ESN ruled by Eqs. 7 and 3 correctly implements the flip-flop task with two bits.

As the dynamics of systems (x_1, x_2) and (x_3, x_4) are independent from each other, the set of fixed points of the overall dynamics turns out to be the Cartesian product of the set of fixed points of (x_1, x_2) and the fixed points of (x_3, x_4) system. This gives rise to a large number of fixed points: there are 4 stable points, 8 saddles with 1 unstable directions, 8 saddles with 2 unstable directions, 4 saddles with 3 unstable directions and 1 repeller. However, the set of fixed points where the nonautonomous flip-flop dynamics takes place is formed by 4 stable fixed points and 8 saddles with only one unstable direction (see "Evaluation on Manually Designed Low-Dimensional ESNs" for a detailed example). Every stable fixed point is close to a pair of saddles and, due to symmetry, they are all at the same distance $\delta_{th}(\mathbf{p}_1, \mathbf{p}_2)$. This quantity defines the excitability thresholds of the connections needed in the flip-flop task, and therefore implicitly defines the ENA ruling the behaviour of the four-dimensional ESN.

Extracting ENAs from the ESN Trajectory

In this section, we describe the proposed algorithm to extract an ENA from an ESN trajectory. The proposed algorithm, which is schematically illustrated in Fig. 5, takes the trained reservoir matrix $\mathbf{M}$ (6) and a trajectory of the nonautonomous ESN (7) as input and produces a weighted directed graph, representing the ENA describing the ESN behaviour, as output. The algorithm is composed of two main steps. In "Finding Fixed Points of the Dynamics", we describe the procedure to find fixed points of the ESN dynamics (corresponding to the vertices of the graph). Successively, in "Determining Excitable Connections Between Attractors", we show how to determine the excitable connections and related thresholds between stable fixed points (corresponding to the weighted directed edges).

Finding Fixed Points of the Dynamics

The optimisation algorithm we have used to find fixed points is based on [55] (see [16] for an open-source Tensorflow toolbox for finding fixed points in arbitrary RNN architectures and [25] for an alternative method to identify fixed points). The key idea is to define a scalar function whose minima correspond to fixed points of the ESN dynamics.

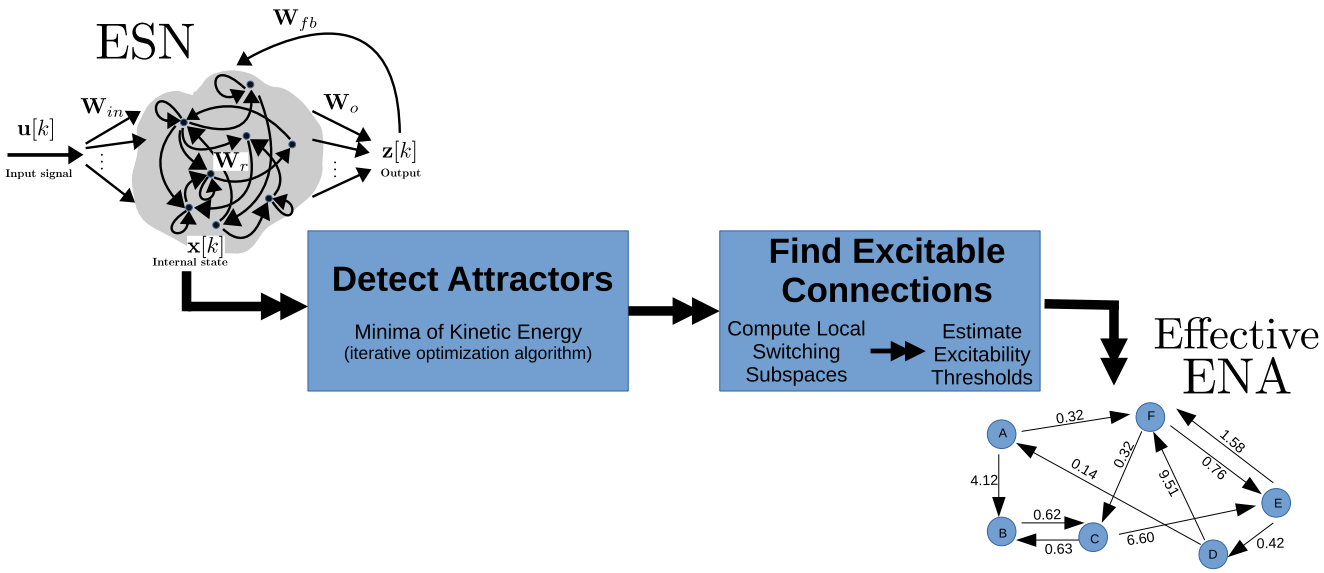


Fig. 5 The proposed method to extract ENAs from trajectories

Definition 7 We call *velocity field* of the autonomous system (10) the vector field $\mathbf{Q} : \mathbb{R}^{N_r} \longrightarrow \mathbb{R}^{N_r}$, defined as

$$\mathbf{Q}(\mathbf{x}) := \mathbf{F}(\mathbf{x}) - \mathbf{x}, \tag{21}$$

with $\mathbf{F}$ being the map in Eq. 10.

Therefore, the velocity field takes the following form:

$$\mathbf{Q}(\mathbf{x}) = \alpha \left[\tanh(\mathbf{M} \cdot \mathbf{x}) - \mathbf{x}\right], \tag{22}$$

where $\mathbf{Q}(\mathbf{x}[k])$ is the vector that needs to be added to the current state $\mathbf{x}[k]$ to obtain the next one. In fact,

$$\begin{aligned} \mathbf{x}[k+1] = \mathbf{F}(\mathbf{x}[k]) &\iff \mathbf{x}[k+1] - \mathbf{x}[k] = \mathbf{F}(\mathbf{x}[k]) - \mathbf{x}[k] \\ &\iff \mathbf{x}[k+1] = \mathbf{x}[k] + \mathbf{Q}(\mathbf{x}[k]). \end{aligned} \tag{23}$$

Definition 8 We define *kinetic energy* of the autonomous system (10) to be the following scalar function:

$$q(\mathbf{x}) := \frac{1}{2} \|\mathbf{Q}(\mathbf{x})\|^2. \tag{24}$$

Note that fixed points $\mathbf{x}^* \in \mathbb{R}^{N_r}$ satisfy $\mathbf{Q}(\mathbf{x}^*) = \mathbf{0}$ if and only if $q(\mathbf{x}^*) = 0$. Fixed points are hence identified as the global minima of Eq. 24. We use the quasi-Newton algorithm BFGS [42] to minimise (24). In order to speed up the optimisation by several orders of magnitude, we explicitly provided the gradient of Eq. 24 to BFGS, which reads:

$$\nabla q(\mathbf{x}_0) = \mathbf{J}_{\mathbf{Q}}(\mathbf{x}_0)^T \mathbf{Q}(\mathbf{x}_0) = \alpha \left(\mathbf{D}(\mathbf{x}_0)\mathbf{M} - \mathbf{I}_{N_r}\right)^T \mathbf{Q}(\mathbf{x}_0), \tag{25}$$

where $\mathbf{I}_{N_r}$ is an $N_r \times N_r$ identity matrix, $\mathbf{J}_{\mathbf{Q}}(\mathbf{x}_0)$ denotes the Jacobian matrix of the velocity field (22) and $\mathbf{D}(\mathbf{x}_0)$ is a diagonal matrix defined in Eq. 31 of Appendix A, both evaluated on $\mathbf{x}_0$.

The initial conditions for minimising (24) are determined by randomly sampling states from a trajectory of the nonautonomous ESN (7). The convergence of the BFGS algorithm depends on a tolerance. In fact, the algorithm may converge to similar solutions that, depending on chosen tolerance, are numerically different. As these solutions represent fixed points of the dynamics, we aggregate them in a meaningful way and return a nonredundant set of fixed points. For this purpose, as a post-processing step, we run a clustering algorithm on the set of all solutions and return only cluster representatives (details provided in Appendix C).

Determining Excitable Connections Between Attractors

Once stable fixed points have been identified, we determine the excitable connections between them. As discussed before (and in more detail in Appendix B), the ESN training (6) induces bifurcations that generate new fixed points, and possibly also other attracting regions in phase space that, however, are not explicitly modelled in this work. The ESN is driven by control inputs that allow us to correctly perform switches between stable states. As a consequence, we first need to understand how such inputs affect the dynamics from a geometric point of view. This is done in "Local Switching Subspaces" by introducing the notion of local switching subspace (LSS), which is strictly related to the notion of local input response set, Eq. 15, introduced in "Network Attractors". Then, in "Estimation of Excitability Thresholds", we describe how we determine

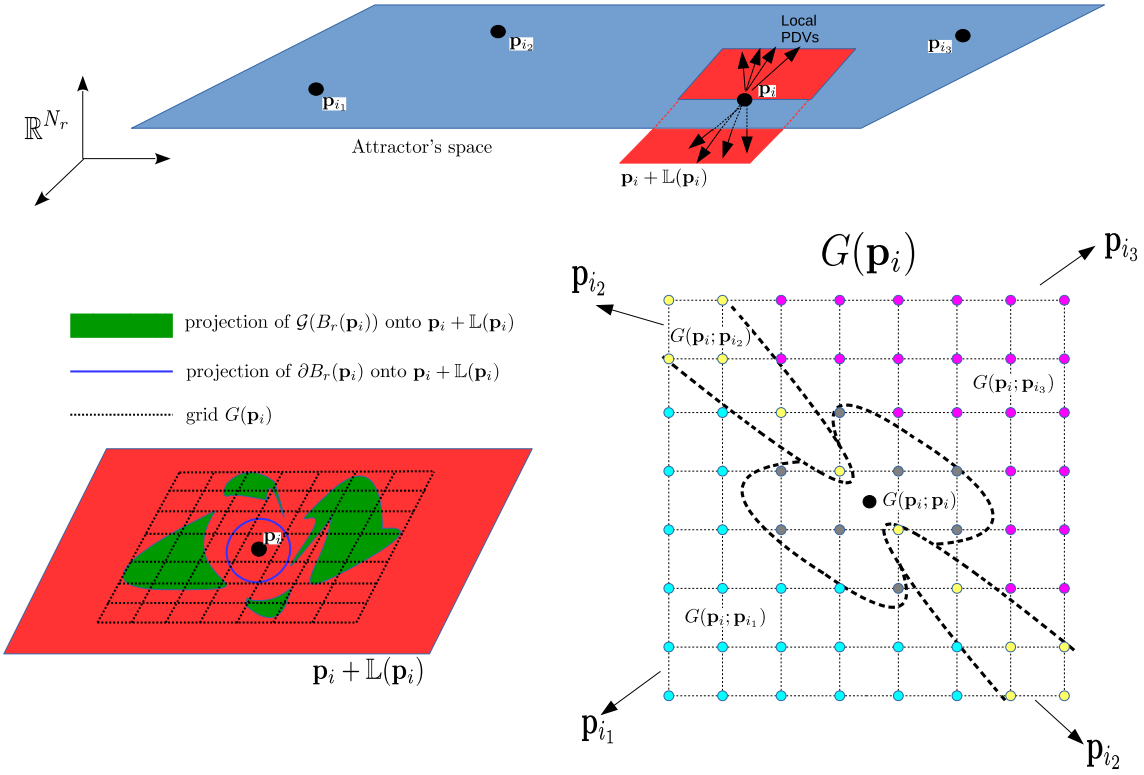


Fig. 6 **Top** In blue, the low-dimensional space containing the attractors; black arrows represent local PDVs (26) originating from the attractor $\mathbf{p}_i$ which in turn define the LSS of $\mathbf{p}_i$, sketched in red. **Bottom left** Representation of the local input response set $\mathcal{G}(B_r(\mathbf{p}_i))$, in green, and the grid $G(\mathbf{p}_i)$, dashed line, in the LSS of $\mathbf{p}_i$, in red. **Bottom right** The partition of a two-dimensional grid $G(\mathbf{p}_i)$. Every colour represents a subset (27). Dashed black lines track the boundary of the basins corresponding to those attractors reachable from $\mathbf{p}_i$ through excitable connections, which can be enabled by inputs allowing exploration of the hypercube centred on $\mathbf{p}_i$

all excitable connections that are relevant for the task under consideration and compute their excitability thresholds. The method we propose is based on a grid of points lying in the LSS, accounting for the input action on the dynamics. By simulating the autonomous dynamics with initial conditions taken from such a grid, we are able to approximate input-driven excitability thresholds (16) and also to quantify how likely it is that the RNN uses such connections while solving the task.

Local Switching Subspaces

Definition 9 Let us consider an ESN trajectory (7), $\mathbf{x}[0], \mathbf{x}[1], \mathbf{x}[2], \ldots, \mathbf{x}[k], \ldots$, obtained with inputs $\mathbf{u}[1], \mathbf{u}[2], \ldots, \mathbf{u}[k+1], \ldots$. Moreover, let us denote with $\mathcal{K} := \{k_i\}_{i \in \mathbb{N}}$ the set of indices for which $\mathbf{u}[k_i] \neq \mathbf{0}$. We define *pulse difference vector* (PDV) a vector containing the difference between pre- and post-input states, namely

$$\mathbf{x}[k_i] - \mathbf{x}[k_i - 1], \quad k_i \in \mathcal{K}. \tag{26}$$

The PDVs (26) convey relevant information about the action of inputs on ESN state and we exploit such information to determine the excitable connections and related thresholds.

Remark We remind the reader that the number of nonzero inputs is controlled by the parameter p of the exponential distribution (see (1) and related discussion in the text), which in turn determines the (average) number of PDVs available for the following analysis.

In order to compute only those connections that are actually used by the ESN while solving the task, we need to focus on the action of inputs in the neighbourhood of each attractor. To this end, we consider an Euclidean ball $B_r(\mathbf{p})$ of radius $r \geq 0$ centred on an attractor $\mathbf{p}$, and call *local* PDVs of $\mathbf{p}$ the finite set of PDVs that originate inside $B_r(\mathbf{p})$. Therefore, referring to Eq. 15, the local PDVs of $\mathbf{p}$ is a collection of vectors originating in $B_r(\mathbf{p})$ and ending in $\mathcal{G}(B_r(\mathbf{p}))$.

Definition 10 Let $\mathbb{L}(\mathbf{p})$ be the vector space obtained by means of principal components analysis of the local PDVs (26) of the attractor $\mathbf{p}$, retaining only $l \ll N_r$ principal components. We define the *local switching subspace* (LSS) of $\mathbf{p}$, and we denote it as $\mathbf{p}+\mathbb{L}(\mathbf{p})$, the affine space composed by attaching the vector space $\mathbb{L}(\mathbf{p})$ over $\mathbf{p}$ (see Fig. 6, top panel).

Estimation of Excitability Thresholds

The idea is to sample the LSS of a stable point $\mathbf{p}_i$ and describe it as a grid of points $G(\mathbf{p}_i)$. Then, we consider those points on the grid as initial conditions for the autonomous system (10), which is then iterated for a sufficiently large number of steps to ensure convergence to nearby attractors. Finally, tracing the evolution of these initial conditions allows us to estimate excitability thresholds and other relevant quantities.

The proposed algorithm for finding excitability thresholds is graphically illustrated in Fig. 6. The algorithm is based on a hypercube $H(\mathbf{p}_i)$ centred on the attractor $\mathbf{p}_i$ that is contained within the LSS $\mathbf{p}_i + \mathbb{L}(\mathbf{p}_i)$. The length of the hypercube is such that $H(\mathbf{p}_i)$ contains the projection of $\mathcal{G}(B_r(\mathbf{p}_i))$ on $\mathbf{p}_i + \mathbb{L}(\mathbf{p}_i)$. In order to estimate excitability thresholds, we consider a mesh with a pre-defined density of points on the hypercube $H(\mathbf{p}_i)$, thus obtaining a grid of points, $G(\mathbf{p}_i) = \{\mathbf{g}_j^i\}$. To simplify the notation, we use a single index j to enumerate points of the grid. Through the ω-limit set (11) of the grid $G(\mathbf{p}_i)$, that is:

$$\omega_{\mathbf{F}}(G(\mathbf{p}_i)) = \bigcup_j \omega_{\mathbf{F}}(\mathbf{g}_j^i),$$

we can compute the following nonnegative integer $c(\mathbf{p}_i) := |\omega_{\mathbf{F}}(G(\mathbf{p}_i))| - 1$, which counts the number of excitable connections originating from $\mathbf{p}_i$ which are allowed to be activated through input. Indeed, $\omega_{\mathbf{F}}(G(\mathbf{p}_i))$ is composed of a set of stable fixed points, $\{\mathbf{p}_{i_0}, \mathbf{p}_{i_1}, \mathbf{p}_{i_2}, \ldots, \mathbf{p}_{i_{c(\mathbf{p}_i)}}\}$, determining the endpoints of the $c(\mathbf{p}_i)$ different excitable connections.

Remark Note that if the grid is sufficiently dense, then the attractor itself is always included in such a set of fixed points. However, we do not count this as an excitable connection. In what follows, we assume that the attractor is indexed by i_0, i.e. $\mathbf{p}_{i_0} = \mathbf{p}_i$.

With these definitions in mind, we are ready to compute thresholds of excitable connections. Let us denote with

$$\sigma_i : \{1, \ldots, |G(\mathbf{p}_i)|\} \longrightarrow \{i_0, i_1, \ldots, i_{c(\mathbf{p}_i)}\},$$

an indexing function such that $\mathbf{p}_{\sigma_i(j)} = \omega_{\mathbf{F}}(\mathbf{g}_j^i)$. Through the pre-image $\sigma_i^{-1}(\cdot)$, we obtain a partition of points of the grid into the following subsets:

$$G(\mathbf{p}_i; \mathbf{p}_{i_t}) := \{\mathbf{g}_j^i \in G(\mathbf{p}_i) \mid j \in \sigma_i^{-1}(i_t)\},\ t = 0, 1, \ldots, c(\mathbf{p}_i). \tag{27}$$

The points of the grid belonging to the subset defined by Eq. 27 are all destined to converge to $\mathbf{p}_{i_t}$. Therefore, we estimate the input-driven excitability threshold (16) of the connection from $\mathbf{p}_i$ to $\mathbf{p}_{i_t}$ as follows:

$$\widetilde{\delta}_{\text{inp}}(\mathbf{p}_i, \mathbf{p}_{i_t}) = \min \left\{ \|\mathbf{g}_j^i - \mathbf{p}_i\| \ : \ \mathbf{g}_j^i \in G(\mathbf{p}_i; \mathbf{p}_{i_t}) \right\}, \quad t = 1, \ldots, c(\mathbf{p}_i). \tag{28}$$

The excitability thresholds (28) represent geometric properties of the attractors and related basins in phase space learned through training. In order to determine the effective excitability (accounting for inputs) of each connection outgoing from $\mathbf{p}_i$, we exploit the local topology of the LSS of $\mathbf{p}_i$ by means of the grid partition (27) induced by $\sigma_i(\cdot)$. To this end, we define the ratio of initial conditions taken from the grid that converged to attractor $\mathbf{p}_{i_t}$ as follows:

$$\nu_{i,i_t} := \frac{|G(\mathbf{p}_i; \mathbf{p}_{i_t})|}{|G(\mathbf{p}_i)| - |G(\mathbf{p}_i; \mathbf{p}_i)|} \in [0, 1]. \tag{29}$$

In the limit of infinite number of points in the grid, the quantity (29) converges to the ratio of volumes between the portion of the hypercube belonging to the basin of $\mathbf{p}_{i_t}$ and the portion of the hypercube that does not belong to the basin of $\mathbf{p}_i$. Therefore, dense grids give ratios (29) providing accurate information about how the LSS is distributed between basins of attraction of stable fixed points. Finally, merging both Eqs. 28 and 29 into a single expression, we define *effective excitability* of the connection from $\mathbf{p}_i$ to $\mathbf{p}_{i_t}$ as follows:

$$\beta_{i,i_t} := \frac{\nu_{i,i_t}}{\widetilde{\delta}_{\text{inp}}(\mathbf{p}_i, \mathbf{p}_{i_t})}. \tag{30}$$

Note that this quantity takes into account both the distance between the attractor $\mathbf{p}_i$ and the basin's boundary, and the volume of the basin itself. A low value for β_{i,i_t} indicates that it is difficult to activate the connection from $\mathbf{p}_i$ to $\mathbf{p}_{i_t}$ during the task. This may be due (i) to the small volume occupied by the basin of the attractor $\mathbf{p}_{i_t}$ in the LSS of $\mathbf{p}_i$ or (ii) to a very high excitability threshold associated with such a connection. On the other hand, $\beta_{i,i_t} \gg 1$ necessarily implies that such a connection has a low excitability threshold, since $\nu_{i,i_t} \in [0, 1]$. As a consequence, the distance between the basin of attraction of $\mathbf{p}_{i_t}$ and $\mathbf{p}_i$ is small, thus the connection can be easily activated during the task.

Remarks on Computational Complexity There are three parameters controlling the complexity and, accordingly, the accuracy of the search in the grid: the dimension ζ_1 of the hypercube, the length ζ_2 of the hypercube's edge, and the number of points ζ_3 on each edge determining the density of the grid. ζ_3 and ζ_2 have a linear and polynomial impact on the computational complexity of the algorithm, respectively. However, ζ_1 is more critical as it increases exponentially the number of points in the grid and, accordingly, the overall complexity. In the simulations, we always set $\zeta_1 = l$, that is, the dimension of the hypercube is equal to the dimension

of the LSS which is low in our case. More generally, the dimension of LSSs depends on the complexity of the inputs and their impact on the dynamics, and this will need to be assessed on a case-by-case basis.

Simulations

In this section, we discuss the results of our simulations and relate this to our theoretical framework. In "Evaluation on Manually Designed Low-Dimensional ESNs", in order to evaluate the correctness of the algorithm proposed in "Extracting ENAs from the ESN Trajectory", we apply it to the manually designed, low-dimensional ESN maps discussed in "A Minimal-Dimension Example to Solve the Two-Bit Flip-Flop Task" and "A $2k$-Dimensional Model for k-Bit Flip-Flop Tasks". The section "Application of the Proposed Method to High-Dimensional Trained ESNs" applies our method to high-dimensional trained ESNs. We show that, even though the ESN is high dimensional, the dynamics generated by the trained reservoir (6) is effectively low dimensional. We also show the usefulness of ENAs for giving a mechanistic interpretation of prediction errors occurring during task execution. Finally, in "Noise Tolerance and Effective Excitability of ENAs", we show how ENA models can be used to assess the robustness to noise of trained ESNs. For all simulations, we use $p = 0.1$ as a parameter of the exponential distribution governing the occurrence of input pulses and set the tolerance of the kinetic energy (24) for detecting minima to 10^{-6}.

Evaluation on Manually Designed Low-Dimensional ESNs

For the grid search algorithm, we used $\zeta_1 = 2, \zeta_2 = 4$ and $\zeta_3 = 223$.

Minimal-dimension model The LSS determined as described in "Local Switching Subspaces" corresponds to the whole 2D phase space. As a consequence, excitability thresholds (13) match the corresponding input-dependent counterparts (16). For each attractor, the Euclidean distances from the two closest saddles are consistent with the estimation of excitability thresholds provided by our method. As discussed in "A Minimal-Dimension Example to Solve the Two-Bit Flip-Flop Task" and graphically represented in Fig. 7 bottom centre panel, we find three excitability thresholds in the computed ENA: 0.49, 1.39 and 1.42. In Fig. 7, bottom-right panel, we show that undesired connections (i.e. connections corresponding to state transitions that are not defined in the flip-flop task) between fixed points $(0.97, -0.97)$ and $(-0.97, 0.97)$ have very low effective excitability values (30), indicating that it is unlikely to use such connections during the task execution. The low effective excitability is due to the small volume ratio (29) of the basins associated with the two attractors. On the other hand, larger thresholds characterise the undesired connections from $(-1, -1)$ to $(1, 1)$, reflecting a lack of symmetry between the basin volumes of $(1, 1)$, $(-1, -1)$ and $(0.97, -0.97)$, $(-0.97, 0.97)$.

Four-Dimensional Model The first two principal components obtained via principal component analysis are related to all identified fixed points and produce a cumulative variance ratio larger than 0.96; Fig. 8 shows a trajectory of the map described in "A $2k$-Dimensional Model for k-Bit Flip-Flop Tasks" and related fixed points projected on the plane spanned by the two principal components. For every attractor, the computed LSS is a two-dimensional plane; this is expected since it is actually the plane depicted in Fig. 4. Furthermore, we note that none of these planes is aligned with the one where the attractors lie, stressing the importance of defining reference frameworks local to each attractor that take the action of inputs into account. Figure 8, middle-right panel, shows how the input moves the states out of the plane, using additional dimensions for the switches. The computed ENA, weighted with excitability thresholds (28) and effective excitability (30), is shown in Fig. 8, bottom-left and bottom-right panels, respectively. Symmetries of the dynamical system are clearly present in the resulting directed graphs. All desired connections are characterised by an excitability threshold of $\simeq 0.83$, while undesired ones have higher thresholds equal to $\simeq 1.19$. We note that the presence of undesired connections does not imply that the ESN actually uses such connections during the task execution. To this end, the effective excitability thresholds (30), shown in the bottom-right panel of Fig. 8, provide us with a more realistic picture of the behaviour under the action of inputs. Note that the effective excitability thresholds are very low for the undesired connections, implying that the LSS used by inputs is mostly occupied by basins corresponding to attractors adjacent to the end-point of desired connections.

Application of the Proposed Method to High-Dimensional Trained ESNs

We now consider implementation using ESNs (7) with a reservoir composed of 500 neurons. Experimenting with ESNs with different reservoir sizes confirms that one can get ESNs that can be successfully trained independent of the precise number of neurons, as long as there are enough. We choose 500 neurons for hardware and computing time considerations. For the grid search algorithm, we use $\zeta_1 = 3, \zeta_2 = 18$ and $\zeta_3 = 12$. The state-update (7) is configured without leakage, $\alpha = 1$ and standard deviation of noise ϵ

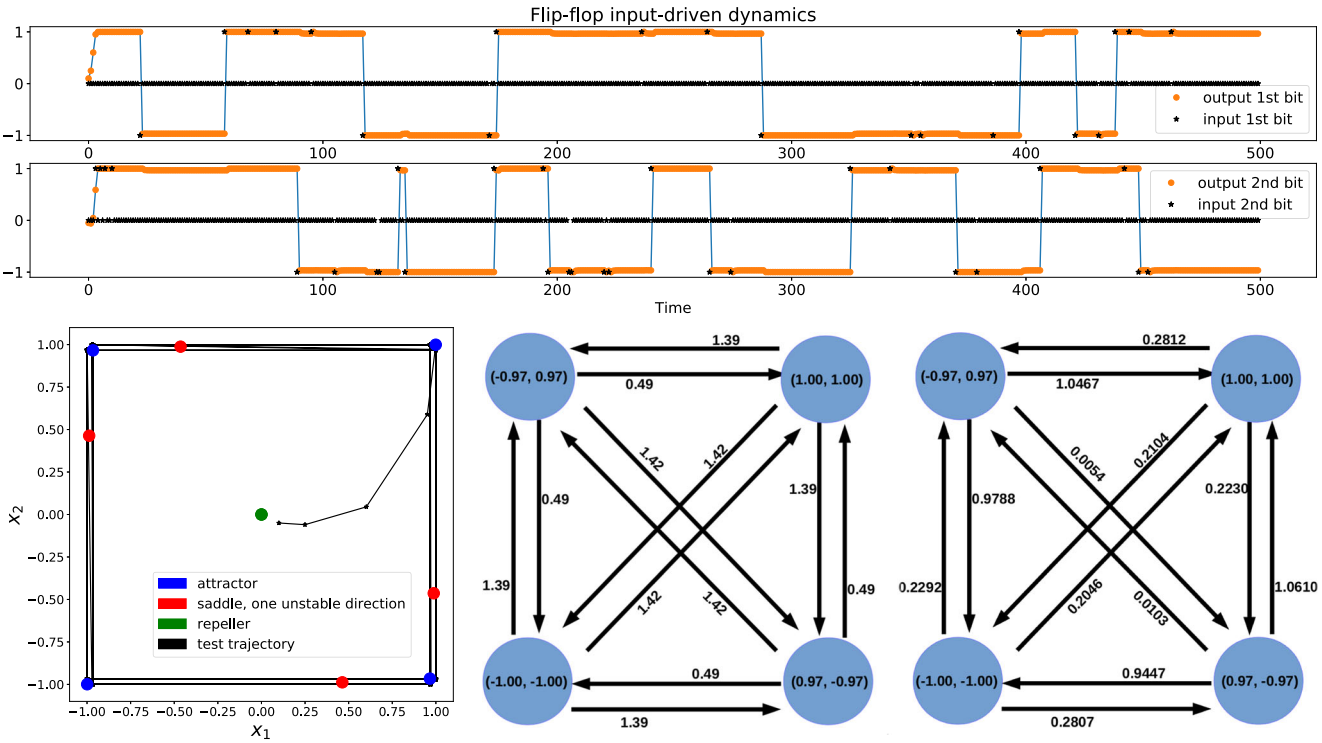


Fig. 7 **Top** Output produced by the two-dimensional ESN defined in "A Minimal-Dimension Example to Solve the Two-Bit Flip-Flop Task". **Bottom left** Fixed points found by the optimisation algorithm with 100 initial conditions. Length of the test trajectory was 1000 steps. **Bottom centre** Extracted ENA with edges weighted by excitability thresholds (28). **Bottom right** Extracted ENA with edges weighted according to Eq. 30. Node labels represent output values (3) produced on the attractors

is set to 10^{-4} during training. The entries of matrices $\mathbf{W}_{in}$, $\mathbf{W}_{fb}$ and $\mathbf{W}_r$ are i.i.d. drawn from a uniform distribution in $[-1, 1]$; the sparseness of $\mathbf{W}_r$ is 95%. Moreover, the reservoir matrix was rescaled to obtain a spectral radius equal to 0.9. Finally, the training set length is always 50,000 time-steps.

Low-Dimensional Dynamics Figure 9, top panel, shows the output produced by an ESN achieving high prediction performance. The extracted ENA, shown in the bottom-right panel, reveals that undesired connections have excitability thresholds (16) significantly higher than those of desired connections. This means that, in the LSS of every stable point, basins corresponding to attractors adjacent to the end-point of undesired connections stand relatively far away from the stable point compared to basins of attractors adjacent to the end-point of desired connections. The fixed points of the dynamics lie in a two-dimensional subspace of the 500-dimensional phase space (the cumulative variance ratio is close 1). It is observed that the trajectory spends most of the time close to such a plane. Hence, based on a principal component analysis of the trajectory, we can claim that the dynamics is two-dimensional. Nevertheless, during the task execution, the trajectory is occasionally driven away from such a 2D plane by the inputs in order to achieve the switches between attractors, and this feature is crucial to understand how the trained neural network behaves while solving the task. The cumulative variance ratio for the identification of the LSS of every attractor, as described in "Local Switching Subspaces", revealed that the switching between stable fixed points takes place in a three-dimensional subspace of the phase space, highlighting that the overall dynamics of ESNs is effectively low-dimensional after training. It is worth stressing that such LSSs are usually not aligned with the standard coordinate system of the original phase space and the subspaces where attractors lie, suggesting that inputs operate in phase space regions that are disjoint with respect to the low-dimensional linear subspace of the attractors; this is consistent with results reported in [55].

Computation Accuracy and Spurious Attractors Various measures of accuracy exist for quantifying the performance on prediction tasks. For instance, the mean-squared-error (MSE) is typically adopted in tasks involving continuous targets. The MSE is a real-valued scalar that informs us about how close the computed output is to the target one. However, it is not possible to infer additional insights by looking only at the MSE; for instance, it is not possible to provide a mechanistic description of errors in the

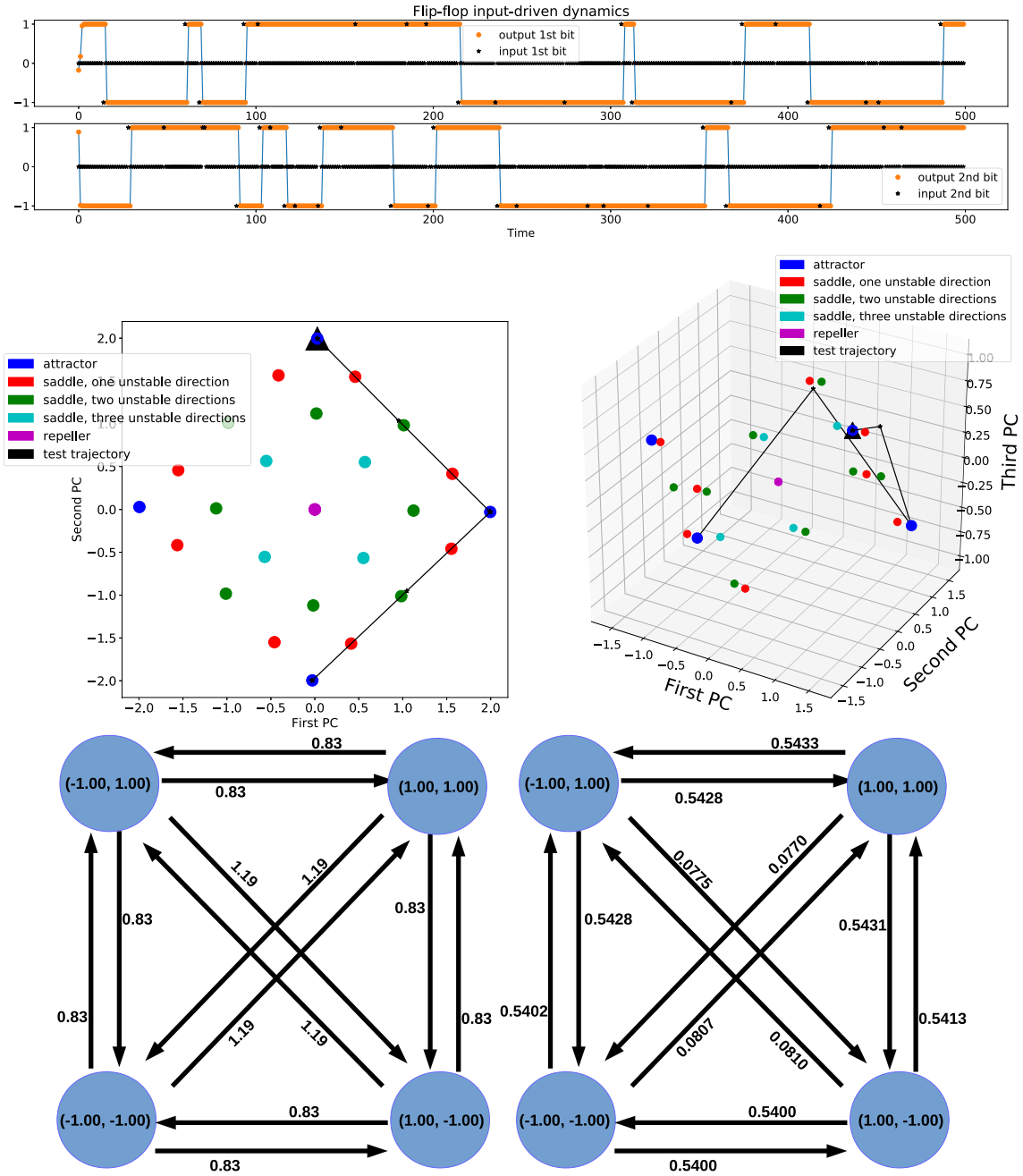


Fig. 8 **Top** Output of the four-dimensional ESN defined in "A Minimal-Dimension Example to Solve the Two-Bit Flip-Flop Task". **Centre left** Fixed points found by the optimisation algorithm with 200 initial conditions, depicted on the space spanned by the first two principal components. In black, it is shown a trajectory starting on the attractor (marked by a black triangle); to improve readability, the trajectory is limited to two switches only. **Centre right** Fixed points and trajectory depicted in the centre-left picture are embedded in the space spanned by the three principal components. **Bottom left** ENA with edge weights representing excitability thresholds (28). **Bottom right** ENA with edge weights representing the effective excitability of connections (30). Node labels represent output values produced by the ESN on the attractors

computation, i.e. why and how they occur. Here, we show how the effective ENA extracted from the ESN trajectory can be used to describe how the computation takes place in phase space and, in particular, how to diagnose the nature of prediction errors.

The top panel in Fig. 10 shows the output produced by an ESN achieving a low MSE of the order of 10^{-3}. The small errors observable around time-step 13,200 can be explained by looking at the ENA model depicted in the bottom panels of Fig. 10, which is represented with excitability thresholds

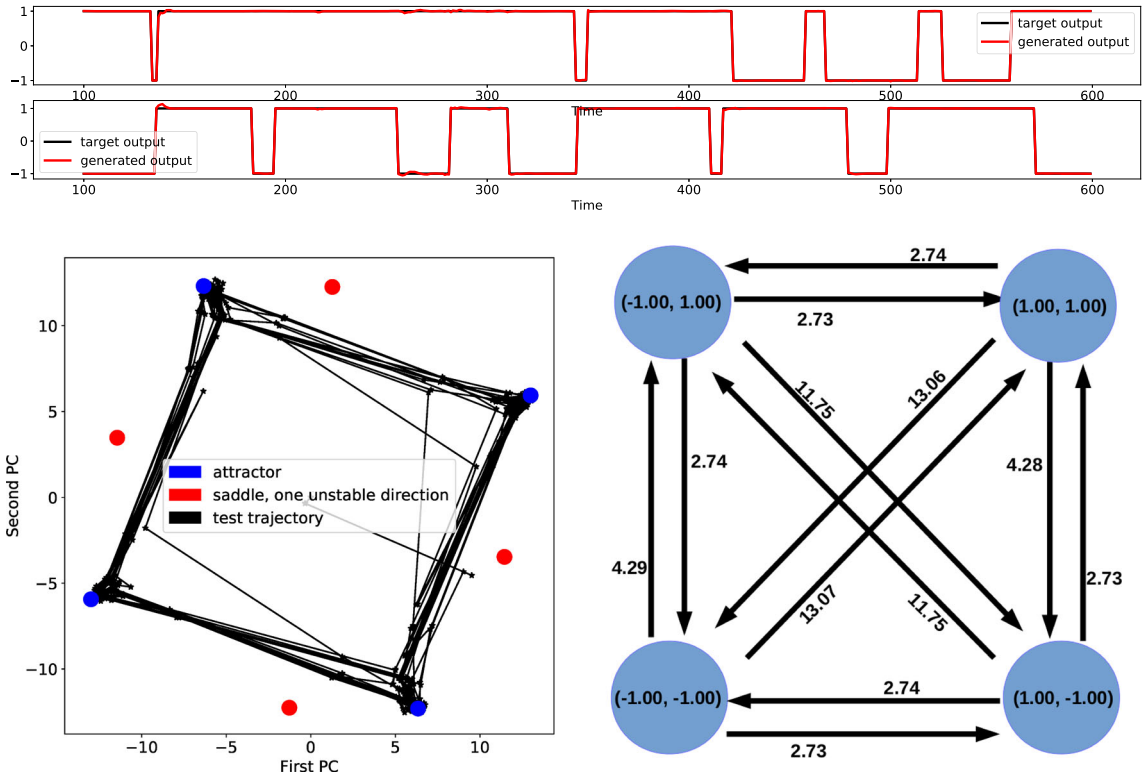


Fig. 9 **Top** Output produced by a trained ESN and target output. **Bottom left** Fixed point found by means of the optimisation algorithm with 500 initial conditions. The plane in the picture represents the space spanned by the first two principal components determined over all 500 solutions returned by the optimisation algorithm. **Bottom right** Extracted ENA with edge weights representing excitability thresholds (28). Node labels represent output values produced by the ESN (3) on the attractors

(28), left panel, and with effective excitability values (30), right panel. The directed graphs (whose topology is clearly identical) reveal the presence of two extra stable fixed points in the ESN phase space, which are generated during training. Nodes of these graphs are coloured according to output values produced by the ESN. The activation of the related excitable connections brings the ESN to operate in the proximity of such superfluous states, producing inaccurate output values and hence explaining the origin of such errors. Notably, the directed edge—in the graph on the right-hand side—connecting the cyan with the yellow node has a relatively high value of effective excitability. This means that, in the LSS of the related attractor (cyan), whose output is $(-1.03, -1.07)$, it is relatively easy to transition to the basin of the other attractor (yellow). This, in turn, produces an output value equal to $(-0.84, -0.40)$, which is significantly different from the target output, i.e. $(-1, -1)$. The prediction error, visible to the naked eye in the top panel around time-step 13,200, is indeed due to the activation of that excitable connection.

Both of these spurious attractors act as a sort of surrogates for the correct ones (i.e., those producing lower prediction errors). In fact, once the internal state switches to a spurious attractor, the ESN still behaves consistently. More precisely, spurious attractors are associated with higher effective excitability values on connections ending up in the correct attractors (see the bottom-right panel of Fig. 10). The existence of these spurious attractors and the unravelling of their roles in the ESN computation could not easily be inferred by looking only at the MSE or plotting the outputs produced by the ESN. This demonstrates the importance of ENA models for describing the ESN behaviour.

Noise Tolerance and Effective Excitability of ENAs

Here, we aim to provide a further example of the importance of ENAs for characterising the computation of ESNs. In particular, we ask the following question: given a set of ESNs trained on the same task and achieving the same performance during training, which one will be more robust to noise during test? Typical performance measures, such as MSE, cannot be used to answer such a question and provide useful insights. We show that an ENA model of the computation, weighted with effective excitability values (30), allows us to assess robustness to noise of ESNs.

As a perturbation, we consider the usual white Gaussian noise term $\boldsymbol{\epsilon}$ in the state-update (7), corresponding to

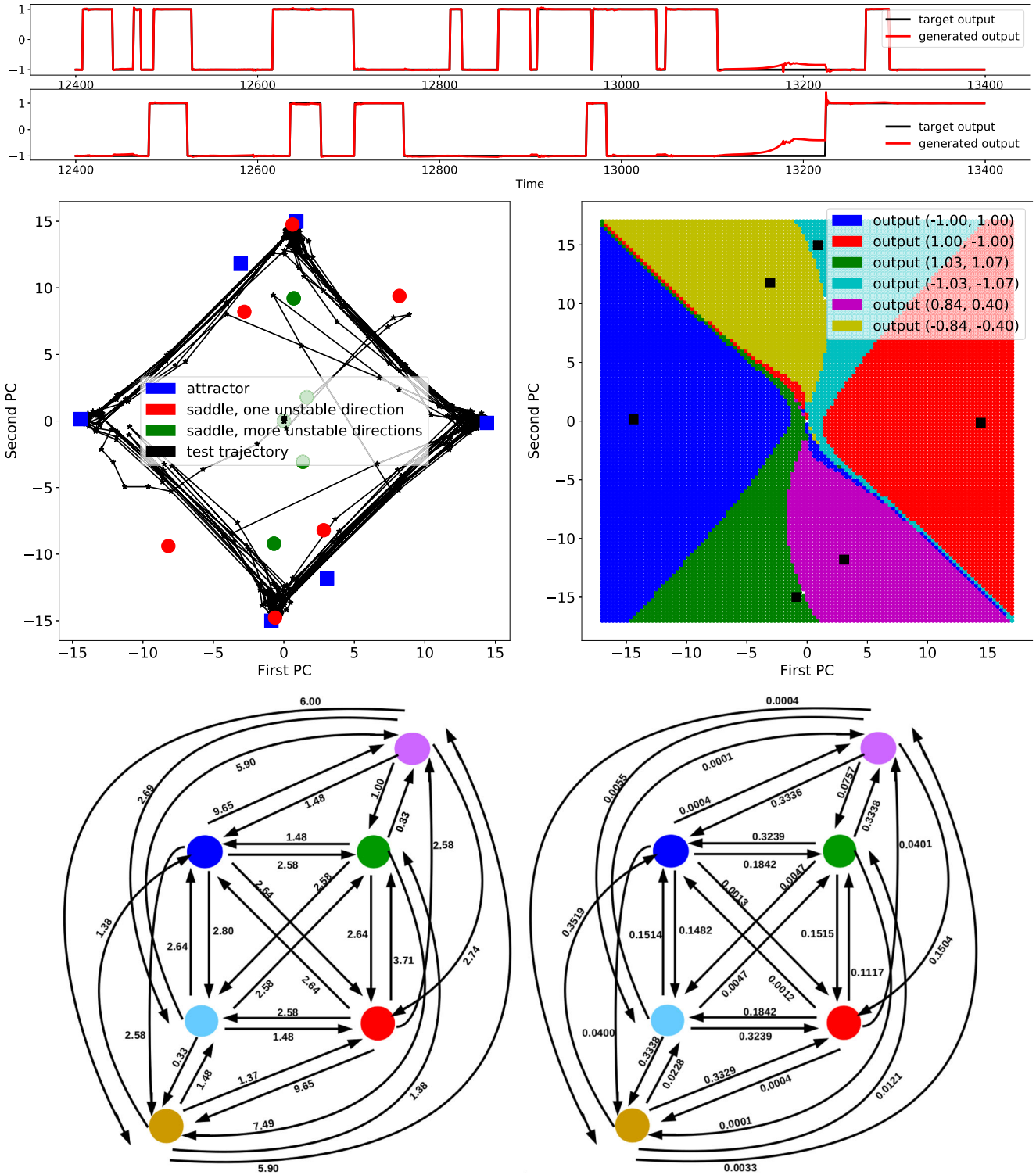


Fig. 10 **Top** Output of the trained ESN showing an incorrect switch around step 13, 200. **Centre left** Fixed points found by the optimisation algorithm plotted in their two-dimensional subspace together with the trajectory. **Centre right** Attractor plane divided by basins of attraction of every stable fixed point. The figure is drawn assuming a transient of 300 time-steps. Every basin is coloured depending on the attractor where the ESN converges to without applying any input. **Bottom left** Extracted ENA weighted with estimation of input-driven excitability thresholds (28). **Bottom right** Extracted ENA weighted with effective excitability value (30). Nodes are coloured according to the colours used in the centre-right figure

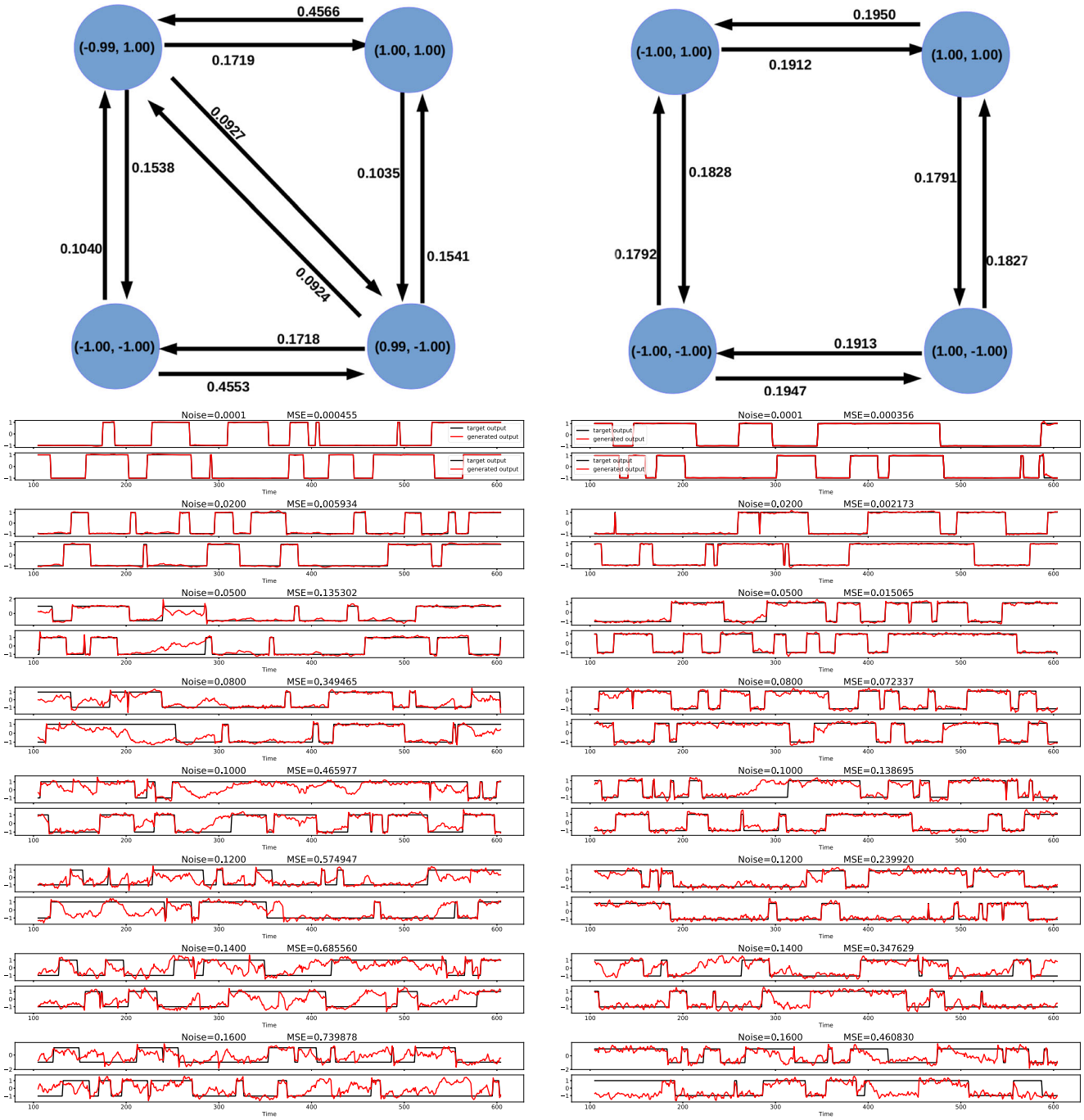


Fig. 11 **Left** Results corresponding to a trained ESN giving rise to an ENA with undesired connections and unbalanced outgoing weights (30). **Right** Results obtained with a trained ESN giving rise to an ENA with balanced outgoing weights for all excitable connections. From top to bottom, in both columns, ESN outputs and related MSEs obtained with increasing noise standard deviation: 10^{-4}, $2 \cdot 10^{-2}$, $5 \cdot 10^{-2}$, $8 \cdot 10^{-2}$, 10^{-1}, $1.2 \cdot 10^{-1}$, $1.4 \cdot 10^{-1}$ and $1.6 \cdot 10^{-1}$

perturbations directly applied to all neuron pre-activation values. We stress that such perturbations are applied only during the test phase; training is implemented as described in the previous section. By increasing the noise standard deviation, the ESN trajectory gets increasingly perturbed neglecting the possibility to reach the proximity of attractors, hence affecting the accuracy of the resulting output values.

We note that (i) ESNs yielding ENAs with higher effective excitability values are less robust to noise perturbations than ESNs giving rise to ENAs with low effective excitability values (see [6]), and (ii) ESNs producing ENAs with

balanced edge weights on desired outgoing connections are more robust to noise perturbations than ENAs with unbalanced outgoing weights. Regarding (i), high excitability values imply the existence of connections with low excitability thresholds. That is, the basin of the attractor corresponding to the end-point of such a connection is very close to the attractor associated with the origin of the connection, resulting in unnecessary switches that induce errors. Concerning (ii), for a given attractor, unbalanced outgoing connection weights could be a symptom of unbalanced distribution of space between basins of attraction in the LSS or the existence of some basins significantly closer to the attractor than other basins. In the latter case, a reasoning similar to (i) applies. On the other hand, if there is indeed an asymmetric distribution of volumes between basins then basins of attractors corresponding to connections possessing large volume ratios will fill most of the LSS, leaving little space for basins of those attractors related to connections with small volume ratios. Therefore, if the ESN state is close to an attractor with unbalanced outgoing connection weights, but with similar excitability thresholds, then under noise perturbations it is more likely to switch towards an attractor reachable through a connection with high effective excitability even when such a connection should not be activated. For these reasons, an ENA, where each node is characterised by balanced weights on outgoing connections and very low effective excitability values on undesired connections, provides a prototypical example of ESNs whose behaviour is robust to noise.

To quantify the robustness of this behaviour to noise, we selected two ESNs that solve the two-bit flip-flop task with very high and comparable accuracy—MSE during training is $\simeq 10^{-4}$. We test them on the same input series composed of 10^5 time-steps and recorded their MSEs by considering different noise instances with increasing standard deviation. Directed graphs representing the extracted ENAs are shown in Fig. 11. Results for increasing noise standard deviation are divided in two columns: on the left column, we report results obtained by the ESN that is least tolerant to noise. The directed graph on the left-hand side of Fig. 11 shows the presence of two undesired connections. The edge weights of desired connections are not balanced, especially those outgoing from attractors with output values $(1, 1)$ and $(-1, -1)$. Conversely, the directed graph on the right-hand side does not contain undesired connections and possess balanced outgoing weights. The absence of undesired connections indicates that, in the LSS of every attractor, the basins of the attractors corresponding to undesired connections do not exist or, alternatively, they are very small and hence not detectable with the grid density used in our simulations; therefore, they are not relevant for describing the ESN behaviour according with the numerical precision we considered in our simulations. Finally, we highlight that a performance breakdown for the ESN on the left-hand side panel is observed starting from noise standard deviation of 8×10^{-2}. On the other hand, the ESN on the right-hand side is significantly more robust to noise, denoting a performance breakdown for noise standard deviation of 1.4×10^{-1}.

Conclusions

In this paper, we present a novel methodology for modelling and interpreting the behaviour of RNNs driven by inputs. In order to obtain a mechanistic model describing how RNNs solve tasks, we exploit the theoretical framework offered by excitable network attractors [5], which are defined as networks of stable fixed points connected by excitable connections. We introduce a procedure to extract excitable network attractors directly from a trajectory generated by a trained RNN. Such a procedure is composed of two main steps: first, fixed points are computed by solving a non-linear optimisation problem [55] and successively excitable connections, with related thresholds, are determined by simulating the dynamics of the autonomous system.

We validate our theoretical developments by considering ESNs trained on the flip-flop task, a simple yet relevant benchmark that consists of learning a prescribed number of stable states and related switching patterns guided by control inputs. We cannot see any particular theoretical limitations in the application of our framework to more complex RNN architectures, although for more complicated dynamical tasks a detailed computation of bifurcation behaviour may become unfeasible. Simulation results provide several interesting insights on how RNNs solve tasks and highlight the usefulness of excitable network attractors in describing how RNNs undertake computations. We train echo state networks by means of ridge regression. Our results (not shown here) suggest that the regularisation parameter has a direct impact on the number of attracting regions in phase space generated through training: using too low values produces under-regularised models with a large number of attractors. An interesting future perspective consists of studying the impact of different training mechanisms (e.g. via FORCE learning or similar online approaches) on the resulting network attractor.

We believe that the proposed modelling framework based on excitable network attractors will be suitable to describe the RNN behaviour for many, if not all, tasks requiring the learning of a number of attractors (which need not be stable fixed points) and related switching patterns. To this end, in the future, we will also examine classification tasks. A related next step includes the possibility to handle inputs that are not instantaneous pulses, opening the way

to more interesting case studies of practical relevance. As mentioned in the introductory sections, network attractors can in principle be constructed between any type of invariant sets, including limit cycles and strange attractors. Our focus on fixed points was mainly dictated by the fact that computing fixed points from a trajectory is significantly easier than computing limit cycles, for instance.

Clearly, fixed points are not powerful enough to accurately model all possible behaviours in phase space. Therefore, an interesting future perspective consists in extending our modeling framework to handle networks composed of heterogeneous attractors, such as a network in phase space connecting fixed points and limit cycles. In turn, this will allow modeling more complex RNN behaviour. Finally, future directions include embedding the directed graph representing the excitable network attractor extracted from the trajectory in a new phase space, thus producing a set of ordinary differential equations [4] describing the RNN behaviour for the task under consideration.

Acknowledgements We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

Funding Information LL received partial support from the Canada Research Chairs program. PA received partial support of the EPSRC Centre for Predictive Modelling in Healthcare via grant EP/N014391/1.

Compliance with Ethical Standards

Conflict of interest Andrea Ceni, Peter Ashwin and Lorenzo Livi declare that they have no conflict of interest.

Human and Animal Rights This article does not contain any studies with human participants or animals performed by any of the authors.

Appendix A: Linear Stability Analysis

By considering the case where the neural network is autonomous, we follow (10) and write the related map as $\mathbf{x}[k] = \mathbf{F}(\mathbf{x}[k-1]) = \mathbf{G}(\mathbf{x}[k-1], \mathbf{0})$. The ith component of the ESN state, $x_i, i = 1, \ldots, N_r$, evolves according to the scalar map $F_i(\mathbf{x}) = (1-\alpha)x_i + \alpha \tanh(\mathbf{M}_{(i)} \cdot \mathbf{x})$. By noting that:

$$\frac{\partial F_i}{\partial x_j}(\mathbf{x}_0) = (1-\alpha)\delta_{ij} + \alpha \left(1 - \tanh^2(\mathbf{M}_{(i)} \cdot \mathbf{x}_0)\right) m_{ij},$$

where δ_{ij} is the Kronecker delta, it is possible to write the Jacobian matrix evaluated onto $\mathbf{x}_0$ as $\mathbf{J_F}(\mathbf{x}_0) = (1-\alpha)\mathbf{I}_{N_r} + \alpha \mathbf{D}(\mathbf{x}_0)\mathbf{M}$, where

$$\mathbf{D}(\mathbf{x}_0) = \begin{bmatrix} 1 - \tanh^2(\mathbf{M}_{(1)} \cdot \mathbf{x}_0) & 0 & \ldots & 0 \\ 0 & 1 - \tanh^2(\mathbf{M}_{(2)} \cdot \mathbf{x}_0) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 - \tanh^2(\mathbf{M}_{(N_r)} \cdot \mathbf{x}_0) \end{bmatrix} \tag{31}$$

is an $N_r \times N_r$ diagonal matrix representing the squashing action of $\tanh(\cdot)$ along the saturating components of $\mathbf{x}_0$. By linearising the network state-update (7) around a given fixed point $\mathbf{x}^*$, we obtain the linear system $\delta\mathbf{x}[k+1] = \mathbf{J_F}(\mathbf{x}^*)\delta\mathbf{x}[k]$, where $\delta\mathbf{x}[k] = \mathbf{x}[k] - \mathbf{x}^*$. It is known [53] that if a fixed point $\mathbf{x}^*$ is hyperbolic (i.e. $\mathbf{J_F}(\mathbf{x}^*)$ has no eigenvalues on the unit circle in the complex plane), then the linear approximation provides a bona fide characterisation of the nonlinear behaviour around that fixed point. Therefore, the (linear) stability of $\mathbf{x}^*$ is completely determined by the spectral radius of $\mathbf{J_F}(\mathbf{x}^*)$. If all eigenvalues of $\mathbf{J_F}(\mathbf{x}^*)$ are inside the unit circle, then $\mathbf{x}^*$ is a stable fixed point; on the other hand, if even just one eigenvalue has norm larger than 1, then the linearised map is expanding along the corresponding eigenvectors and $\mathbf{x}^*$ is called a saddle[5]. We conclude observing that holds $x_i^* = \tanh(\mathbf{M}_{(i)} \cdot \mathbf{x}^*) \iff \alpha x_i^* = \alpha \tanh(\mathbf{M}_{(i)} \cdot \mathbf{x}^*) \iff x_i^* + \alpha x_i^* = x_i^* + \alpha \tanh(\mathbf{M}_{(i)} \cdot \mathbf{x}^*) \iff x_i^* = (1-\alpha)x_i^* + \alpha \tanh(\mathbf{M}_{(i)} \cdot \mathbf{x}^*)$; hence, the number of fixed points and their positions in phase space do not change on varying $\alpha \in (0, 1]$. However, their linear stability properties are directly affected by α.

Appendix B: Bifurcation of Fixed Points in ESNs

In what follows, we perform a bifurcation analysis of one-dimensional ESN map and provide some sufficient conditions to design two-dimensional ESNs with a desired number of fixed points. In particular, we focus on *fold bifurcations*[6] of low-dimensional ESN maps, which

[5] If every eigenvalue has a norm greater than 1, then the fixed point is a repeller.

[6] We refer to [28] for the terminology; the fold bifurcation is also known as *saddle-node*, *limit point* or *turning point* bifurcation.

constitute the only codimension-1 bifurcation that generate new fixed points; as discussed by [57], it is the usual mechanism for creating new attractive fixed point in RNNs. Moreover, as shown by [7], some fold bifurcations are responsible for reducing the dimensions where the actual dynamics takes place, dividing the RNN parameter space in different regions of effective dimensionality.

B.1 ESN Maps in One Dimension

Here, we consider the following one-dimensional map,

$$x[k] = \tanh(mx[k-1] + w), \tag{32}$$

where $(m, w) \in \mathbb{R}^2$ are the bifurcation parameters; we simplify the state-update in Eq. 7 and set leak rate $\alpha = 1$, $\boldsymbol{\epsilon} = 0$, and remove explicit reference to inputs. Nevertheless, studying the map (32) is still useful to obtain insights about high-dimensional input-driven ESN dynamics. In fact, in the high-dimensional case (7), the activation function of the jth neuron is determined by:

$$x_j[k] = \tanh\left(m_{jj}x_j[k-1] + \sum_{s\neq j} m_{js}x_s[k-1] + (\mathbf{W}_{in})_{(j)} \cdot \mathbf{u}[k] + \varepsilon_j\right). \tag{33}$$

Hence, the parameter w in Eq. 32 can be interpreted as the weighted sum of all incoming neurons, plus input and noise terms. Fixed points of the map $F(x) = \tanh(mx + w)$ are the solutions x^* of the equation $Q(x^*) = 0$, where:

$$Q(x) := F(x) - x = \tanh(mx + w) - x. \tag{34}$$

It is not possible to find a closed-form expression for the fixed points. However, it is possible to state that (i) for every $(m, w) \in \mathbb{R}^2$, there exists at least one fixed point; (ii) there can be one, two or three fixed points. The proof of these statements follows straightforwardly from the fact that $\lim_{x\to\pm\infty} Q(x) = \mp\infty$ and because if $m < 1$, then Q is monotonic. Otherwise, there exist two critical points, $x_l < x_r$, namely

$$x_{l,r} = \frac{1}{m}\left[\pm\tanh^{-1}\left(\sqrt{\frac{m-1}{m}}\right) - w\right], \tag{35}$$

such that the function $Q(x)$ folds. Moreover, since $Q(x^*) = 0 \iff x^* = \tanh(mx^* + w) \in (-1, 1)$ for every $(m, w) \in \mathbb{R}^2$, all fixed points lie in the open interval $(-1, 1)$. Critical points in Eq. 35 are solutions to $Q'(x) = 0$, or equivalently to $F'(x) = 1$.

Let us assume $m > 0$. We observe a fold bifurcation whenever a critical point assumes the zero value. The condition $Q(x_{l,r}) = 0$ gives rise to the following parametrisation of the fold bifurcation curve,

$$w_\pm(m) := \pm\left[m\sqrt{\frac{m-1}{m}} - \tanh^{-1}\left(\sqrt{\frac{m-1}{m}}\right)\right], \quad m \in [1, +\infty), \tag{36}$$

which possesses two symmetric branches ending on a cusp (see Fig. 12 for an illustration). Crossing that curve in parameter space towards the region containing the semi-axis of $m > 1$, a new fixed point is formed (x_l or x_r, depending on the branch) and splits in a pair of fixed points, one stable and one unstable[7]. This curve delimits the boundary between a dynamic regime where two stable points coexist with an unstable one, and a regime where only one fixed point exists. Due to symmetry, in the $m < 0$ case, there are two bifurcation branches identifying a *flip* or *period doubling bifurcation*, which is analytically described by the curve $w_\pm(-m)$ with $m \leq -1$. Crossing that curve towards the region containing the semi-axis of $m < -1$, a stable fixed point loses stability and gives rise to a 2-periodic attracting trajectory surrounding it. We note that the flip bifurcation is detrimental for the flip-flop task considered here, as it gives period rather than fixed point attractors.

B.2 ESN Maps in Two Dimensions

Here, we consider a two-neuron reservoir. We denote the activation functions of these neurons as x and y, so that the ESN state evolution defines a trajectory $(x[k], y[k]) \in [-1, 1]^2$, $k = 1, 2, ...$, ruled by:

$$\begin{aligned} x[k] &= \tanh(ax[k-1] + by[k-1]), \\ y[k] &= \tanh(cx[k-1] + dy[k-1]). \end{aligned} \tag{37}$$

It is known that a fixed point can undergo only fold or flip bifurcations in one-dimensional, discrete-time dynamical systems [28]. Nevertheless, considering two or more dimensions, also *Neimark-Sacker* bifurcations could occur, where a stable point loses stability and an invariant curve surrounding it is created. These are the only possible codimension-1 bifurcations of a fixed point for a discrete-time dynamical system. Among them, only the fold bifurcation can generate new fixed points. The origin is always a fixed point of Eq. 10. Considering (37), the Jacobian matrix evaluated on the origin reads:

$$\mathbf{W}_r = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

Therefore, training ESNs via (6) implies a qualitative change of the dynamics around the origin if some eigenvalue crosses the unit circle, i.e. fixed point $(0, 0)$ bifurcates.

[7] The particular case of crossing that curve through the cusp gives rise to a pitchfork bifurcation of the origin.

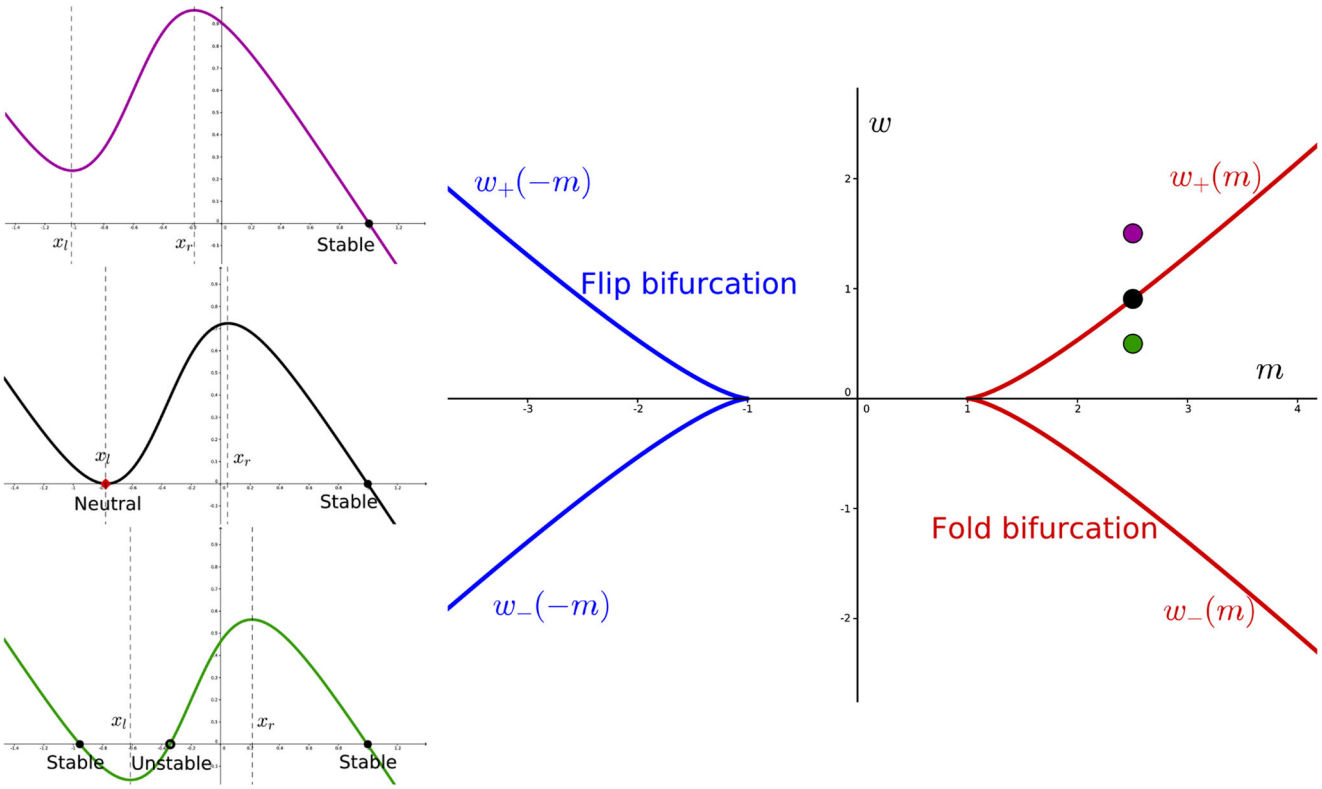


Fig. 12 **Left** Three different folding configurations of the function $Q(x)$ in Eq. 34 with $m = 2.5$; purple $w = 1.5$, black $w \approx 0.9$, green $w = 0.5$. **Right** Bifurcation diagram of fixed points in one-dimensional ESN

However, adding a low-rank matrix to the reservoir can induce global bifurcations far away from the origin as well; hence, we cannot rely on the local bifurcation of the origin to deduce the global attractor structure after training. As a counterexample, suppose $\lambda_1, \lambda_2 > 1$. The diagonal matrix $\text{diag}(\lambda_1, \lambda_2)$ and the upper triangular matrix $\begin{pmatrix} \lambda_1 & \gamma \\ 0 & \lambda_2 \end{pmatrix}$ share the same spectrum $\{\lambda_1, \lambda_2\}$. Nevertheless, if the coupling is strong enough, that is $|\gamma| > \dfrac{w_+(\lambda_1)}{y^*}$, where $w_+(\lambda_1)$ is the function (36) evaluated on λ_1 and y^* is the positive stable solution of $y[k] = \tanh(\lambda_2 y[k-1])$, then the fold bifurcation curve is crossed, making the dynamics for x variable trivial. As a consequence of this, the upper triangular matrix induces dynamics with just two attractors (plus two saddles and the origin is a repeller), while the diagonal matrix induces four attractors (plus four saddles and the repeller).

In order to count the number of fixed points of the map (37), we can draw its nullclines. Defining function $N_{\alpha,\beta}(\eta) := \dfrac{1}{\beta}[-\alpha\eta + \tanh^{-1}(\eta)]$, the nullclines are given by:

$$\begin{aligned} y &= N_{a,b}(x), \\ x &= N_{d,c}(y), \end{aligned} \tag{38}$$

and they represent the locus of points where x-dynamic/y-dynamic is stationary. As a consequence, the solutions of the algebraic nonlinear system (38) coincide with the set of fixed points. In the last part of this subsection, we show some sufficient conditions to control the number of fixed points assuming $a, d > 1$, i.e. when both nullclines are folding. We refer to Fig. 13 for a graphical illustration.

- Case $bc \geq 0$.

$$|N'_{a,b}(0)| < |N'_{d,c}(0)|^{-1}, \text{ i.e. } (1-a)(1-d) < bc \\ \implies \text{there are exactly 3 fixed points.} \tag{39}$$

On the other hand, when $(1-a)(1-d) \geq bc$, there could exist 5, 7 or 9 fixed points. Critical points of the function $N_{\alpha,\beta}(\eta)$ are $\eta_\pm = \pm\sqrt{\frac{\alpha-1}{\alpha}}$. Therefore, if $(1-a)(1-d) \geq bc$ holds then

$$\left|N_{a,b}\left(\sqrt{\frac{a-1}{a}}\right)\right| < \sqrt{\frac{d-1}{d}} \text{ or } \left|N_{d,c}\left(\sqrt{\frac{d-1}{d}}\right)\right| < \sqrt{\frac{a-1}{a}} \implies \text{there are exactly 5 fixed points.} \tag{40}$$

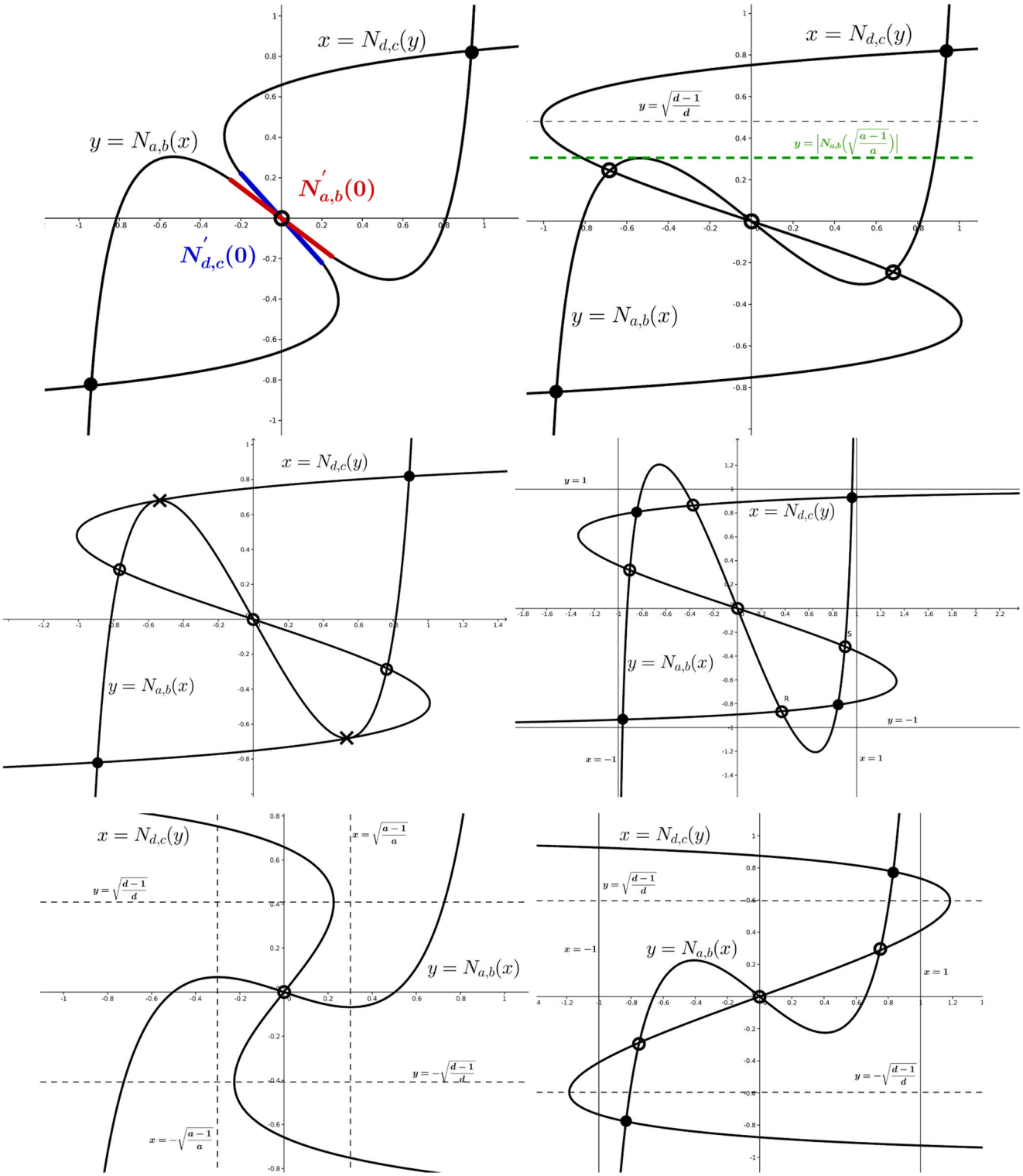


Fig. 13 In all panels, filled points denote stable attractors, while circles denote saddles or repellers. **Top left** Geometric representation of the sufficient condition (39); nullcline slopes on the origin are highlighted with different colours. **Top right** The condition (40); the black dashed line acts as an upper bound ensuring that the maximum height of the peak of the x-nullcline (represented by the green dashed line) does not cross further the y-nullcline. **Centre left** Special case of 7 fixed points. Crosses indicate neutral fixed points where fold bifurcations take place. **Centre right** An example of nullcline configuration holding the condition (43) in the case of $bc > 0$. Both curves come out of the invariant square $[-1, 1]^2$ with their humps, that guarantees 9 intersections, i.e. 9 fixed points. **Bottom left** Depiction of the sufficient condition (41); the origin is the only fixed point and it is unstable. **Bottom right** The sufficient condition (42)

Figure 13 depicts a nullcline configuration where there are 5 intersections[8]. From that configuration, we can make the humps of $y = N_{a,b}(x)$ more pronounced by increasing the $\frac{a}{b}$ ratio, until a fold bifurcation occurs, giving rise to a new couple of pair of fixed points. The case of 7 fixed points is obtained after the fold bifurcation.

- Case $bc < 0$.

$$\left|N_{a,b}\left(\sqrt{\frac{a-1}{a}}\right)\right| < \sqrt{\frac{d-1}{d}} \text{ and } \left|N_{d,c}\left(\sqrt{\frac{d-1}{d}}\right)\right| < \sqrt{\frac{a-1}{a}} \implies \text{there is exactly 1 fixed point.} \quad (41)$$

From the above configuration, increasing the d/c ratio makes stretch the humps of $N_{d,c}$ until a fold bifurcation occurs producing exactly three fixed points. Beyond the fold bifurcation, a new pair of fixed points is generated[9].

$$\left|N_{a,b}\left(\sqrt{\frac{a-1}{a}}\right)\right| < \sqrt{\frac{d-1}{d}} \text{ and } \left|N_{d,c}\left(\sqrt{\frac{d-1}{d}}\right)\right| > 1 \implies \text{there are exactly 5 fixed points.} \quad (42)$$

In both cases, a simple sufficient condition to ensure the existence of 9 fixed points (see Fig. 13) is

$$\left|N_{a,b}\left(\sqrt{\frac{a-1}{a}}\right)\right| > 1 \text{ and } \left|N_{d,c}\left(\sqrt{\frac{d-1}{d}}\right)\right| > 1 \implies \text{there are exactly 9 fixed points.} \quad (43)$$

The maximum number of nullcline intersections is 9, setting the maximum number of fixed points that can be generated in a two-dimensional ESN map.

Appendix C: Aggregation of Fixed Points via Clustering

The optimisation procedure described in "Finding Fixed Points of the Dynamics" provides a number of solutions equal to the number of initial conditions taken into account (assuming all initial conditions converge); however, such solutions might be similar up to a prescribed numerical precision. In order to reduce the set of all solutions to a few effective fixed points, we run the k-means clustering algorithm and retain only the elements belonging to the final clusters minimising the distance w.r.t. the cluster centroids. Instead of aggregating all solutions at once, we first group such solutions according to their linear stability properties as indicated by the spectrum of the Jacobian matrix of the autonomous map (10). In particular, we form the group of linear stable fixed points, the group with one unstable direction and so on. Finally, for each group, we run k-means with parameter k identified according to the minimum of the Davies-Bouldin index [2].

[8]Note that it holds $N_{\alpha,\beta}\left(\pm\sqrt{\frac{\alpha-1}{\alpha}}\right) = -\frac{1}{\beta}w_{\pm}(\alpha)$.

[9]Due to symmetry, this condition holds also when inverting the role of $N_{d,c}$, $N_{a,b}$ and consequently (a, b), (d, c).

References

1. Aljadeff J, Renfrew D, Vegué M, Sharpee TO. Low-dimensional dynamics of structured random networks. Phys Rev E. 2016;93(2):022302. https://doi.org/10.1103/PhysRevE.93.022302.
2. Arbelaitz O, Gurrutxaga I, Muguerza J, Pérez JM, Perona I. An extensive comparative study of cluster validity indices. Pattern Recogn. 2013;46(1):243–56. https://doi.org/10.1016/j.patcog.2012.07.021.
3. Arjovsky M, Shah A, Bengio Y. Unitary evolution recurrent neural networks. In: International conference on machine learning, pp 1120–1128, New York, USA; 2016.
4. Ashwin P, Postlethwaite C. On designing heteroclinic networks from graphs. Physica D: Nonlinear Phenomena. 2013;265:26–39. https://doi.org/10.1016/j.physd.2013.09.006.
5. Ashwin P, Postlethwaite C. Designing heteroclinic and excitable networks in phase space using two populations of coupled cells. Journal of Nonlinear Science. 2016;26(2):345–64. https://doi.org/10.1007/s00332-015-9277-2.
6. Ashwin P, Postlethwaite C. Sensitive finite state computations using a distributed network with a noisy network attractor. IEEE Transactions on Neural Networks and Learning Systems, pp 1–12. 2018. https://doi.org/10.1109/TNNLS.2018.2813404.
7. Beer RD. Parameter space structure of continuous-time recurrent neural networks. Neural Comput. 2006;18(12):3009–51. https://doi.org/10.1162/neco.2006.18.12.3009.
8. Bianchi FM, Scardapane S, Uncini A, Rizzi A, Sadeghian A. Prediction of telephone calls load using echo state network with exogenous variables. Neural Netw. 2015;71:204–13. https://doi.org/10.1016/j.neunet.2015.08.010.
9. Bianchi FM, Livi L, Alippi C. Investigating echo state networks dynamics by means of recurrence analysis. IEEE Transactions on Neural Networks and Learning Systems. 2018;29(2):427–39. https://doi.org/10.1109/TNNLS.2016.2630802.
10. Castelvecchi D. Can we open the black box of AI? Nat News. 2016;538(7623):20.
11. Cencini M, Cecconi F, Vulpiani A. Chaos: from simple models to complex systems. Singapore: World Scientific; 2010.
12. Chung J, Gulcehre C, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555. 2014.
13. De Pasquale B, Cueva CJ, Rajan K, Escola GS, Abbott LF. full-FORCE: a target-based method for training recurrent networks. PLoS ONE. 2018;13(2):1–18,2. https://doi.org/10.1371/journal.pone.0191527.
14. Funahashi K, Nakamura Y. Approximation of dynamical systems by continuous time recurrent neural networks. Neural Netw. 1993;6(6):801–806.
15. Gallicchio C, Micheli A. Echo state property of deep reservoir computing networks. Cogn Comput. 2017;9(3):337–50. ISSN 1866-9964. https://doi.org/10.1007/s12559-017-9461-9.
16. Golub MD, Sussillo D. Fixedpointfinder: a tensorflow toolbox for identifying and characterizing fixed points in recurrent neural networks. The Journal of Open Source Software. 2018;3:1003. https://doi.org/10.21105/joss.01003.

17. Goodman B, Flaxman S. European union regulations on algorithmic decision-making and a right to explanation. arXiv:1606.08813. 2016.
18. Graves A, Mohamed A-R, Hinton G. Speech recognition with deep recurrent neural networks. In: Proceedings of IEEE international conference on acoustics, speech and signal processing, pp 6645–6649, Vancouver, BC, Canada. IEEE; 2013.
19. Hammer B. On the approximation capability of recurrent neural networks. Neurocomputing. 2000;31(1):107–23. https://doi.org/10.1016/S0925-2312(99)00174-5.
20. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80.
21. Jaeger H, Haas H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science. 2004;304(5667):78–80. https://doi.org/10.1126/science.1091277.
22. Jaeger H, Lukoševičius M, Popovici D, Siewert U. Optimization and applications of echo state networks with leaky-integrator neurons. Neural Netw. 2007;20(3):335–52. https://doi.org/10.1016/j.neunet.2007.04.016.
23. Kanai S, Fujiwara Y, Iwamura S. Preventing gradient explosions in gated recurrent units. In: Advances in neural information processing systems, pp 435–444; 2017.
24. Karpatne A, Atluri G, Faghmous JH, Steinbach M, Banerjee A, Ganguly A, Shekhar S, Samatova N, Kumar V. Theory-guided data science: a new paradigm for scientific discovery from data. IEEE Trans Knowl Data Eng. 2017;29(10):2318–31. ISSN 1041-4347. https://doi.org/10.1109/TKDE.2017.2720168.
25. Katz GE, Reggia JA. Using directional fibers to locate fixed points of recurrent neural networks. IEEE Trans Neural Netw Learn Syst. 2018;29(8):3636–46. https://doi.org/10.1109/TNNLS.2017.2733544.
26. Keuninckx L, Danckaert J, Van der Sande G. Real-time audio processing with a cascade of discrete-time delay line-based reservoir computers. Cogn Comput. 2017;9(3):315–26. https://doi.org/10.1007/s12559-017-9457-5.
27. Koryakin D, Lohmann J, Butz MV. Balanced echo state networks. Neural Netw. 2012;36:35–45. https://doi.org/10.1016/j.neunet.2012.08.008.
28. Kuznetsov YA. Elements of applied bifurcation theory, vol 112. Berlin: Springer; 2013.
29. Livi L, Bianchi FM, Alippi C. Determination of the edge of criticality in echo state networks through Fisher information maximization. IEEE Trans Neural Netw Learn Syst. 2018;29(3):706–17. https://doi.org/10.1109/TNNLS.2016.2644268.
30. Løkse S, Bianchi FM, Jenssen R. Training echo state networks with regularization through dimensionality reduction. Cogn Comput. 2017;9(3):364–378. https://doi.org/10.1007/s12559-017-9450-z.
31. Lukoševičius M. A practical guide to applying echo state networks. Berlin: Springer; 2012, pp. 659–686. https://doi.org/10.1007/978-3-642-35289-8_36.
32. Lukoševičius M, Jaeger H. Reservoir computing approaches to recurrent neural network training. Cogn Comput. 2009;3(3):127–49. https://doi.org/10.1016/j.cosrev.2009.03.005.
33. Maass W, Joshi P, Sontag ED. Computational aspects of feedback in neural circuits. PLoS Comput Biol. 2007;3(1):e165. https://doi.org/10.1371/journal.pcbi.0020165.eor.
34. Manjunath G, Jaeger H. Echo state property linked to an input: exploring a fundamental characteristic of recurrent neural networks. Neural Comput. 2013;25(3):671–96. https://doi.org/10.1162/NECO_a_00411.
35. Mastrogiuseppe F, Ostojic S. Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. Neuron. 2018. ISSN 0896-6273. https://doi.org/10.1016/j.neuron.2018.07.003.
36. Mayer NM, Yu Y-H. Orthogonal echo state networks and stochastic evaluations of likelihoods. Cogn Comput. 2017;9(3):379–90. https://doi.org/10.1007/s12559-017-9466-4.
37. Miller KD, Fumarola F. Mathematical equivalence of two common forms of firing rate models of neural networks. Neural Comput. 2012;24(1):25–31.
38. Miller P. Itinerancy between attractor states in neural systems. Curr Opin Neurobiol. 2016;40:14–22. https://doi.org/10.1016/j.conb.2016.05.005.
39. Milnor J. On the concept of attractor. 1985.
40. Montavon G, Samek W, Müller K-R. Methods for interpreting and understanding deep neural networks. Digital Signal Process. 2017;73:1–15. https://doi.org/10.1016/j.dsp.2017.10.011.
41. Neves FS, Voit M, Timme M. Noise-constrained switching times for heteroclinic computing. Chaos: An Interdisciplinary Journal of Nonlinear Science. 2017;27(3):033107. https://doi.org/10.1063/1.4977552.
42. Nocedal J, Wright SJ. Numerical optimization. 2006: Springer, Berlin.
43. Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks. In: Proceedings of the 30th international conference on machine learning, vol 28, pp 1310–1318, Atlanta, Georgia, USA; 2013.
44. Pham V, Bluche T, Kermorvant C, Louradour J. Dropout improves recurrent neural networks for handwriting recognition. In: 14th international conference on frontiers in handwriting recognition, pp 285–290, Crete Island, Greece; 2014, https://doi.org/10.1109/ICFHR.2014.55.
45. Rabinovich M, Volkovskii A, Lecanda P, Huerta R, Abarbanel H, Laurent G. Dynamical encoding by networks of competing neuron groups: winnerless competition. Phys Rev Lett. 2001;87(6):068102.
46. Rabinovich M, Huerta R, Laurent G. Transient dynamics for neural processing. Science. 2008;321:48–50. https://doi.org/10.1126/science.1155564.
47. Rajan K, Abbott LF, Sompolinsky H. Stimulus-dependent suppression of chaos in recurrent neural networks. Phys Rev E. 2010;82(1):011903. https://doi.org/10.1103/PhysRevE.82.011903.
48. Reinhart RF, Steil JJ. Regularization and stability in reservoir networks with output feedback. Neurocomputing. 2012;90:96–105. https://doi.org/10.1016/j.neucom.2012.01.032.
49. Rivkind A, Barak O. Local dynamics in trained recurrent neural networks. Phys Rev Lett. 2017;118:258101. https://doi.org/10.1103/PhysRevLett.118.258101.
50. Rodan A, Tiňo P. Simple deterministically constructed cycle reservoirs with regular jumps. Neural Comput. 2012;24(7):1822–52. https://doi.org/10.1162/NECO_a_00297.
51. Ruder S. An overview of gradient descent optimization algorithms. arXiv:1609.04747. 2016.
52. Scardapane S, Uncini A. Semi-supervised echo state networks for audio classification. Cogn Comput. 2017;9(1):125–35. https://doi.org/10.1007/s12559-016-9439-z.
53. Strogatz SH. Nonlinear dynamics and chaos. UK: Hachette; 2014.
54. Sussillo D, Abbott LF. Generating coherent patterns of activity from chaotic neural networks. Neuron. 2009;63(4):544–57. https://doi.org/10.1016/j.neuron.2009.07.018.
55. Sussillo D, Barak O. Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. Neural Comput. 2013;25(3):626–49. https://doi.org/10.1162/NECO_a_00409.
56. Tallec C, Ollivier Y. Can recurrent neural networks warp time? In: International conference on learning representations; 2018. https://openreview.net/forum?id=SJcKhk-Ab.

57. Tiňo P, Horne BG, Giles CL. Attractive periodic sets in discrete-time recurrent networks (with emphasis on fixed-point stability and bifurcations in two-neuron networks). Neural Comput. 2001;13(6):1379–1414.
58. Tsuda I. Chaotic itinerancy and its roles in cognitive neurodynamics. Curr Opin Neurol. 2015;31:67–71. https://doi.org/10.1016/j.conb.2014.08.011.
59. Vincent-Lamarre P, Lajoie G, Thivierge J-P. Driving reservoir models with oscillations: a solution to the extreme structural sensitivity of chaotic networks. J Comp Neurol, pp 1–18. 2016.
60. Weinberger O, Ashwin P. From coupled networks of systems to networks of states in phase space. Discrete & Continuous Dynamical Systems - B. 2018;23:2043. ISSN 1531-3492. https://doi.org/10.3934/dcdsb.2018193.
61. wyffels F, Li J, Waegeman T, Schrauwen B, Jaeger H. Frequency modulation of large oscillatory neural networks. Biol Cybern. 2014;108(2):145–57. https://doi.org/10.1007/s00422-013-0584-0.
62. Yildiz IB, Jaeger H, Kiebel SJ. Re-visiting the echo state property. Neural Netw. 2012;35:1–9. https://doi.org/10.1016/j.neunet.2012.07.005.