

LICENTIATE THESIS

Using Lyapunov Exponents to Explain the Dynamics of  
Complex Systems

LUDVIG STORM

---

Department of Physics  
University of Gothenburg  
Göteborg, Sweden 2024

## ABSTRACT

Complex systems often display chaotic dynamics, characterised by being exponentially sensitive to changes in initial conditions. Such systems are in general difficult to analyse, due to the large number of nonlinearly interacting degrees of freedom. Dynamical-systems theory provides a framework for analysing such systems. One of the tools from this theory is the Lyapunov exponent, which quantifies the rate at which initially nearby trajectories converge or diverge over time. The exponent can be used to study how the stability of a complex system depends on different system parameters. The finite-time Lyapunov exponent can be used to reveal organising structures in the phase space of the system that separate it into different characteristic regions. These structures are referred to as Lagrangian coherent structures.

In this thesis, the Lyapunov exponent and Lagrangian coherent structures are used to explore the properties of complex systems. In the two presented papers, artificial neural networks are analysed, which are machine-learning algorithms with a large number of interconnected nonlinear computational nodes. We show that these systems can be analysed as complex dynamical systems, and show, among other things, how this perspective helps shedding light on how the neural networks learn to perform classification tasks. Additionally, a project on how microswimmers can escape through transport barriers in flows using orientational diffusion is presented, where the transport barriers are Lagrangian coherent structures.

## LIST OF PAPERS

This thesis consists of an introductory text about the two appended papers and an on-going unpublished project:

### **Paper A**

STORM, L., LINANDER, H., BEC, J., GUSTAVSSON, K., & MEHLIG, B. 2024 Finite-time Lyapunov exponents of deep neural networks. *Phys. Rev. Lett.* **132**, 057301.

### **Paper B**

STORM, L., GUSTAVSSON, K., & MEHLIG, B. 2022 Constraints on parameter choices for successful time-series prediction with echo-state networks. *Mach. Learn.: Sci, Technol.* **3**, 045021.

### **Paper C**

STORM, L., MOUSAVI, N., GUSTAVSSON, K., & MEHLIG, B. 2024 Transport barriers for microswimmers with orientational diffusion. *Unpublished*.

## MY CONTRIBUTIONS

My contributions to the appended publications and on-going project are:

### **Papers A**

I performed all numerical simulations except for the adversarial attack and the training of the convolutional network on the CIFAR10 dataset. I contributed to the design of the experiments and the interpretation of the results, especially in the large- $N$  limit. I wrote the paper together with BM with input from KG, HL, and JB.

### **Paper B**

I performed all the numerical simulations and analytical computations. I contributed to the design of the experiments and the interpretation of the results. I wrote the paper with input from BM and KG.

### **Paper C**

I performed all the numerical simulations and participated in the interpretation of the results.

## ACKNOWLEDGEMENTS

I firstly express gratitude to my supervisor, Bernhard Mehlig, for the opportunity to learn from him. I am also very grateful to my co-supervisor Kristian Gustavsson for the insightful discussions and helpful commentaries. I owe my thanks to Anshuman Dubey, Jan Meibohm and Hampus Linander for their advice and to my colleagues Navid, Linus, Enrique, and Mattias for our friendly exchanges. Lastly, a heartfelt thank you to my family for their unwavering support, and particularly to my brother for our discussions.



# CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>I</b>	<b>Background</b>	<b>3</b>
<b>2</b>	<b>Dynamical systems, Lyapunov exponents, and time-delay embedding</b>	<b>4</b>
2.1	Dynamical systems . . . . .	4
2.2	Lyapunov exponents . . . . .	5
2.3	Numerical computation of Lyapunov exponents . . . . .	10
2.4	Time-delay embedding . . . . .	11
<b>3</b>	<b>Lagrangian coherent structures</b>	<b>12</b>
3.1	Definition . . . . .	12
3.2	Finding the structures . . . . .	14
<b>4</b>	<b>Artificial neural networks</b>	<b>14</b>
4.1	Feed-forward neural networks . . . . .	16
4.2	Echo-state networks . . . . .	21
<b>5</b>	<b>Microswimmers and transport barriers</b>	<b>24</b>
5.1	The model . . . . .	25
5.2	The flow . . . . .	26
<b>II</b>	<b>Present work</b>	<b>27</b>
<b>6</b>	<b>Finite-time Lyapunov exponents of deep neural networks</b>	<b>28</b>
6.1	Learning regimes . . . . .	29
6.2	Lyapunov exponent for randomly initialised neural networks . . . . .	33
<b>7</b>	<b>Constraints on parameter choices for successful time-series prediction with echo-state networks</b>	<b>36</b>
7.1	Lyapunov exponent of the reservoir dynamics . . . . .	37
7.2	Full and partial information . . . . .	39
<b>8</b>	<b>Transport barriers for microswimmers with orientational diffusion</b>	<b>44</b>
8.1	Deformation of transport barriers . . . . .	44
<b>9</b>	<b>Conclusions and Outlook</b>	<b>47</b>
9.1	Finite-time Lyapunov exponents of deep neural networks . . . . .	47

9.2	Constraints on parameter choices for successful time-series prediction with echo-state networks . . . . .	48
9.3	Transport barriers for microswimmers with orientational diffusion . . . .	49
<b>III</b>	<b>Appendix</b>	<b>63</b>
A	Maximal Lyapunov exponent of a randomly initialised neural network	63
<b>IV</b>	<b>Research papers</b>	<b>67</b>



# 1 Introduction

Chaotic systems are ubiquitous in nature and engineering. Characterised by an exponentially sensitive dependence on initial conditions, chaotic systems are generally difficult to study. Complex systems – systems with a large number of nonlinearly interacting components – often exhibit chaotic behaviour. In physics, the weather [1], the motion of celestial bodies [2], and the dynamics of particles in turbulent flows [3, 4, 5] are examples of a chaotic system. In biology, the brain has been suggested to operate at the edge of chaos [6], and population dynamics often display chaotic behaviour [7]. In chemistry, the Belousov–Zhabotinsky reaction is an example of chaotic dynamics in chemical reactions [8].

While the sensitivity to initial conditions makes predicting chaotic dynamics difficult, the sensitivity can also yield insight into the structure of the system. For instance, artificial neural networks, machine-learning models used in classification tasks, can display transient chaos [9], and as is shown in Paper A, the networks exploit this property to become sensitive to differences in input signals which help them separate data into different classes.

Lyapunov exponents, a tool from dynamical-systems theory [10, 11], quantify this sensitivity. More precisely, the exponents quantify the average exponential convergence or divergence of nearby trajectories of a dynamical system, and a positive exponent is an indication of chaotic dynamics. The Lyapunov exponents carry a lot of information about the dynamical system. For example, if the sum of Lyapunov exponents is negative, the system is dissipative and information about the initial condition is lost over time. A zero-valued Lyapunov exponent tends to indicate that a symmetry exists in the system [12]. The Lyapunov exponents are asymptotic quantities, calculated in the limit of large times, which, if they exist, are independent of initial conditions [10, 13]. Their finite-time counterpart, the finite-time Lyapunov exponents (FTLE), are position dependent. The FTLEs have been used extensively in studying complex dynamics, such as the alignment of particles in a fluid [14, 15], the detection of transport barriers [16, 17, 18], and transport and mixing processes [19, 20, 21].

Another tool to explore complex dynamics are Lagrangian coherent structures [22, 23], which are closely related to FTLEs. These structures organise the dynamics into different characteristic regions, much like invariant man-

ifolds in dynamical systems, with the benefit of being easily extendable to finite-time settings and time-varying dynamics [24].

In this thesis, Lyapunov exponents and Lagrangian coherent structures are used to understand the structure of complex systems. The work includes two papers where artificial neural networks are analysed, and an on-going project on transport barriers for microswimmers in fluid flows. This text is meant as an introduction to the theory required to understand the results of the work, and as a summary of results and conclusions. To this end, the thesis will begin with a theory section where dynamical-systems theory, the Lyapunov exponent and related topics are presented, followed by theory on Lagrangian coherent structures. Next, theory on artificial neural networks is presented, since much of the work is dedicated to their analysis. Finally, a section on microswimmers is presented. With the theory out of the way, a summary of the results and conclusions of the two published papers and the on-going project is presented, together with a section with conclusions, open questions.

PART I  
BACKGROUND

## 2 Dynamical systems, Lyapunov exponents, and time-delay embedding

In this section, some concepts from dynamical systems theory are introduced that are used in the presented work. Dynamical-systems theory is a framework used for studying the dynamics of complex systems. Complex systems, like the weather or the brain, are nonlinear, and closed form solutions are most often impossible to find. Instead, dynamical-systems theory focuses on studying qualitative properties of the dynamics, such as whether the system is sensitive to perturbations, whether the system displays periodicity, and whether the evolution of the system is bounded to some attractor or is divergent.

### 2.1 Dynamical systems

A dynamical system [11] is defined by an evolution rule  $F(\mathbf{x}, t)$  and a space referred to as phase space in which  $F(\mathbf{x}, t)$  maps an initial position  $\mathbf{x}(0)$  to a later position  $\mathbf{x}(t)$ , that is,  $\mathbf{x}(t) = F(\mathbf{x}(0), t)$ . The parameter  $t$  usually represents time. The evolution rule generates unique trajectories<sup>1</sup> through phase space, and the entire trajectory passing through  $\mathbf{x}(0)$  is referred to as an orbit. If  $t$  is a real-valued parameter, the dynamical system is said to be continuous, while if  $t$  is an integer, the dynamical system is discrete. Usually, an  $m$ -dimensional continuous dynamical system is realised by defining a set of first order differential equations

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t), \quad \mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{x} \in \mathbb{R}^n, t \in \mathbb{R}. \quad (2.1)$$

where  $\mathbf{f}$  is a vector field in  $\mathbb{R}^m$ . The dynamical system is referred to as non-autonomous if  $\mathbf{f}$  explicitly depends on  $t$ , and autonomous otherwise.

An  $m$ -dimensional discrete dynamical system is similarly realised by defining

$$\mathbf{x}(n+1) = \mathbf{f}(\mathbf{x}(n), n), \quad \mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{x} \in \mathbb{R}^n, n \in \mathbb{N}. \quad (2.2)$$

where  $\mathbf{f}$  is a map in  $\mathbb{R}^m$ . Usually, the evolution rule  $F(\mathbf{x}, t)$  cannot be written explicitly, and so the dynamical system is given by the solutions to these

---

<sup>1</sup>The trajectories are unique almost surely, as there may be a set of measure zero of initial conditions for which the trajectories are not defined.

sets of equations. A dynamical system in which a volume evolving in phase space remains constant over time is said to be conservative (e.g. Hamiltonian dynamics). If volumes in phase space shrink over time, the dynamical system is dissipative and therefore not invertible.

Dynamical-systems theory studies the long-term behaviour of time-varying systems. Of particular interest is the stability of the system; that is, whether trajectories diverge or if they are bounded to some region of phase space. Attractors are regions in phase space towards which trajectories converge and remain in. Examples of attractors are fixed points, limit cycles or tori, and strange attractors. The set of initial conditions that converge to the attractor is called the basin of attraction. A dynamical system may have several such basins to different attractors. If a region in phase space is an attractor backwards in time, the region is called a repeller. If, for example, a fixed point is an attractor, it is said to be stable, while it is said to be unstable if it is a repeller.

The stability of attractors/repellers may be studied through linear stability analysis to determine whether a small perturbation away from them will grow or shrink over time. This analysis, however, requires one to first identify the attractors/repellers, which can be difficult. Another approach, which is used to study local stability of trajectories, is to consider how the distance between two initially infinitesimally nearby trajectories evolve over time. The growth or shrinkage of this distance is quantified by the Lyapunov exponents. These will be introduced in the next section.

Why is studying the stability of dynamical systems interesting? As will be seen in the work presented herein, sensitivity to changes to initial conditions can reveal structures of the phase space that define the dynamics. These structures turn out to define decision boundaries for artificial neural networks and transport barriers for microscopic swimmers in fluid flows. Additionally, stability determines whether a system will be able to synchronise to a driving signal; a necessary property in the design of some time-series prediction algorithms.

## 2.2 Lyapunov exponents

In this section, the Lyapunov exponent, which is the central tool in this work, is introduced. The section starts with a definition of Lyapunov exponents and

Lyapunov vectors, followed by a description of how they can be computed numerically.

### 2.2.1 Definition and properties

Lyapunov exponents describe how initially small separations between trajectories defined by a dynamical system evolve in magnitude over time. For some continuous or discrete  $m$ -dimensional dynamical system, denoting the initial, infinitesimal separation at time  $t_0$  as  $\delta \mathbf{x}(t_0) = \delta \mathbf{x}_0$  and the separation at a later time  $t$  as  $\delta \mathbf{x}(t)$ , the maximal Lyapunov exponent is defined as

$$\lambda_1 = \lim_{t \rightarrow \infty} \lim_{\|\delta \mathbf{x}_0\| \rightarrow 0} t^{-1} \ln \frac{\|\delta \mathbf{x}(t)\|}{\|\delta \mathbf{x}_0\|}, \quad (2.3)$$

where  $\|\cdot\|$  is some norm<sup>2</sup>. In words, the maximal Lyapunov exponent quantifies the typical average exponential rate of separation between trajectories in the dynamical system. Hence, a positive exponent signifies a growth in the separation, while the converse is implied by a negative exponent. The word ‘typical’ is used here because a typical perturbation  $\delta \mathbf{x}(0)$  will have a component in the direction in phase space associated with the maximal Lyapunov exponent, and so its rate of separation will be completely dominated by this exponent in the considered limit (assuming non-degeneracy). In fact, a spectrum of  $m$  Lyapunov exponents can be defined, with associated orthogonal Lyapunov vectors spanning  $\mathbb{R}^m$ . In what follows I will define the Lyapunov spectrum more thoroughly, beginning by defining the monodromy matrix, whose singular values and vectors will turn out to be closely related to the Lyapunov spectrum.

### The monodromy matrix and Lyapunov exponents

Consider a continuous dynamical system defined through

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}). \quad (2.4)$$

Given two initial coordinates separated by the infinitesimal vector  $\delta \mathbf{x}_0 = \mathbf{x}'(0) - \mathbf{x}(0)$ , the dynamics of the separation can be obtained through the linearised dynamics:

$$\delta \dot{\mathbf{x}} \approx \nabla \mathbf{f}(\mathbf{x}) \delta \mathbf{x}. \quad (2.5)$$

---

<sup>2</sup>The definition of the Lyapunov exponent is independent of the employed norm [12].

The solution to Eq. 2.5 is formally given by

$$\delta \mathbf{x}(t) = \mathbb{M}(t_0, t) \delta \mathbf{x}_0 \quad (2.6)$$

where  $\mathbb{M}(t_0, t) = \exp\left(\int_{t_0}^t \nabla \mathbf{f}(\mathbf{x}(\tau)) d\tau\right)$ . The matrix  $\mathbb{M}$  thus maps the initial separation  $\delta \mathbf{x}_0$  to later separation  $\delta \mathbf{x}(t)$  and is referred to as the monodromy matrix. Note that taking the time derivative of  $\mathbb{M}(t_0, t)$  yields a matrix differential equation

$$\dot{\mathbb{M}} = \nabla \mathbf{f}(\mathbf{x}(t)) \mathbb{M}, \quad \mathbb{M}(t_0, t_0) = \mathbb{I} \quad (2.7)$$

which is referred to as the variational equation [25]. The evolved monodromy matrix can thus be computed by integrating the dynamical system together with the variational equation.

Since we are interested in how the magnitude of the perturbation changes over time, we compute the Euclidian norm of  $\delta \mathbf{x}$ ,

$$\|\delta \mathbf{x}(t)\| = \sqrt{\delta \mathbf{x}_0^\top \mathbb{M}^\top \mathbb{M} \delta \mathbf{x}_0}. \quad (2.8)$$

The matrix  $\mathbb{M}^\top \mathbb{M}$  is referred to as the right Cauchy-Green matrix, and is a central object of study in continuum mechanics, used for example to study instabilities in plastic deformation [26], crack initiation [27], and the alignment of rods in fluids [14, 28]. We now decompose the monodromy matrix using singular value decomposition to obtain  $\mathbb{M} = \mathbb{U} \mathbb{S} \mathbb{V}^\top$ , where  $\mathbb{U}$  and  $\mathbb{V}$  are orthonormal matrices whose column vectors  $\mathbf{u}_i(t)$  and  $\mathbf{v}_i(t)$  are the left and right singular vectors respectively, and  $\mathbb{S}$  is a diagonal matrix with non-negative entries  $S_{ij} = \sigma_i \delta_{ij}$ , where  $\sigma_i$  are the singular values of  $\mathbb{M}$ . These are, without loss of generality, arranged in descending order so that  $\sigma_i \geq \sigma_{i+1}$ . Inserting this expression into Eq. 2.8, we find that

$$\|\delta \mathbf{x}(t)\| = \sqrt{\delta \mathbf{x}_0^\top \mathbb{V} \mathbb{S}^2 \mathbb{V}^\top \delta \mathbf{x}_0}. \quad (2.9)$$

That is, the eigenvalues of the right Cauchy-Green matrix are the squared singular values of  $\mathbb{M}$  and its eigenvectors are the right singular vectors  $\mathbf{v}_i(t)$ . The geometric interpretation of  $\mathbf{v}_i(t)$  is that an initial perturbation  $\delta \mathbf{x}_0$  tangent with this direction will be scaled by a factor  $\sigma_i$ . Going back to the definition of the Lyapunov exponent (Eq. 2.3), selecting the perturbation  $\delta \mathbf{x}_0$  to be tangential to one of the eigenvectors  $\mathbf{v}_i$  we have

$$\lambda_i(t) = t^{-1} \ln \sigma_i \quad (2.10)$$

which is referred to as the  $i$ :th finite-time Lyapunov exponent (FTLE). The associated right singular vector  $\mathbf{v}_i(t)$  is the finite-time Lyapunov vector (FTLV). To obtain the Lyapunov exponent, we take the infinite-time limit. The convergence

$$\lambda_i = \lim_{t \rightarrow \infty} t^{-1} \ln \sigma_i \quad (2.11)$$

is ensured by Oseledec's theorem [12, 13], which holds if the dynamics are stationary and ergodic. The theorem also shows that the FTLVs converge to the Lyapunov vectors, which are non-random.

### Interpretation and properties

Now let us consider an arbitrary initial perturbation, which we write in the basis of the FTLVs,

$$\delta \mathbf{x}_0 = \sum_i c_i \mathbf{v}_i(t). \quad (2.12)$$

The squared magnitude of the perturbation at time  $t$  will then be

$$\|\delta \mathbf{x}(t)\|^2 = \sum_i c_i^2 e^{2t\lambda_i(t)}, \quad (2.13)$$

where we have written the singular values in terms of the FTLEs. Assuming  $\lambda_1(t) > \lambda_2(t)$ , we see that the growth of the perturbation is completely dominated by the maximal FTLE in the large- $t$  limit. Since the Lyapunov exponent has units [1/s], the inverse of the Lyapunov exponent represents a characteristic time scale of the dynamical system and is called the Lyapunov time. If the maximal Lyapunov exponent is positive, this time scale quantifies for how long the dynamics can be accurately predicted, since any deviation when measuring the dynamics will have become significant beyond this point in time.

Now consider a unit cube  $C(0)$  in  $\mathbb{R}^m$  spanned by the FTLVs. Evolving the FTLVs using the monodromy matrix will result in a parallelepiped spanned by the scaled left singular vectors  $\sigma_i \mathbf{u}_i$ . Since the left singular vectors are orthogonal, the resulting volume is given by the product of singular values. Hence, again writing the singular values in terms of the FTLEs, the evolved volume will be

$$\text{Vol}[C(t)] = e^{t \sum_{i=1}^n \lambda_i}. \quad (2.14)$$



Thus, we see that the sum of Lyapunov exponents tell us the rate of contraction/expansion of volumes in phase space. If the sum is negative, the volume will shrink, yielding dissipative dynamics. If the sum equals zero, volume is conserved and the dynamics are conservative.

### Random-matrix theory

Sometimes the dynamics of a discrete-time system are best modelled as a product of random matrices. Examples of such systems include disordered systems like spin models with random interactions, or chains of harmonic oscillators with random masses [13]. Random-matrix products also arise in the context of artificial neural networks, which are studied in this work. Here, some results concerning the Lyapunov exponents of systems governed by random-matrix products are presented.

Consider an  $m$ -dimensional discrete dynamical system governed by the dynamics

$$\mathbf{x}(n+1) = \mathbb{A}_n \mathbf{x}(n), \quad \mathbf{x}(0) = \mathbf{x}_0, \quad (2.15)$$

where  $\mathbb{A}_n$  is an  $m \times m$  random matrix sampled from some distribution. The state at time step  $n$  can then be written as

$$\mathbf{x}(n) = \mathbb{A}_{n-1} \mathbb{A}_{n-2} \dots \mathbb{A}_0 \mathbf{x}_0 = \mathbb{M}(0, n) \mathbf{x}_0. \quad (2.16)$$

If the distribution is chosen so that the singular values of  $\mathbb{A}_n$  are bounded, the Furstenberg theorem [13] states that the following limit exists:

$$\lambda_1 = \lim_{n \rightarrow \infty} n^{-1} \ln(\sigma_1(\mathbb{M})). \quad (2.17)$$

Here,  $\sigma_1(\mathbb{M})$  is the maximal singular value of the random realisation  $\mathbb{M}$  of the matrix product. The theorem also states that this limit is non-random, i.e.

$$\lambda_1 = \lim_{n \rightarrow \infty} n^{-1} \langle \ln(\sigma_1(\mathbb{M})) \rangle, \quad (2.18)$$

where the average is taken over the distribution of matrix products. Another result is Oseledec's theorem for products of random matrices, which states that the following limit holds:

$$\lim_{n \rightarrow \infty} [\mathbb{M}(0, n)^\top \mathbb{M}(0, n)]^{1/2n} = \mathbb{V}. \quad (2.19)$$

The logarithm of the eigenvalues of  $\mathbb{V}$  are the Lyapunov exponents, and the eigenvectors are the Lyapunov vectors.

## 2.3 Numerical computation of Lyapunov exponents

Finding analytic expressions for Lyapunov exponents is in general difficult. Instead, it is more common to resort to numerical approximation schemes. To obtain the Lyapunov exponents, one must compute the singular values of the monodromy matrix  $\mathbb{M}(t_0, t)$ . Unfortunately, the singular values of the matrix grow or shrink exponentially with time, making directly applying singular value decomposition inaccurate. A way to circumvent this numerical overflow is to decompose the monodromy matrix into time steps and compute the average expansion/contraction performed at each time step. In the continuous case, one discretises the variational equation (Eq. 2.7) as

$$\mathbb{M}(t_0, t + \delta t) = \mathbb{M}(t_0, t) + \delta t [\mathbb{J}(t)\mathbb{M}(t_0, t)] = [\mathbb{I} + \delta t \mathbb{J}(t)]\mathbb{M}(t_0, t). \quad (2.20)$$

and writes  $\mathbb{M}(t_0, t)$  as

$$\mathbb{M}(t_0, t) = [\mathbb{I} + \delta t \mathbb{J}(t - \delta t)][\mathbb{I} + \delta t \mathbb{J}(t - 2\delta t)] \dots [\mathbb{I} + \delta t \mathbb{J}(0)]. \quad (2.21)$$

For discrete dynamics, the decomposition is straight forward:

$$\mathbb{M}(n_0, n) = \mathbb{J}(n-1)\mathbb{J}(n-2) \dots \mathbb{J}(0). \quad (2.22)$$

For notational brevity, we will continue with the discrete case only. The procedure is equivalent for the continuous case. Denoting the matrices in the decomposition as  $\mathbb{A}_i$ , we decompose  $\mathbb{A}_1 = \mathbb{Q}_1\mathbb{R}_1$ , where  $\mathbb{Q}_1$  is an orthonormal matrix and  $\mathbb{R}_1$  is an upper triangular matrix. Then, we decompose the matrix  $\mathbb{A}_2\mathbb{Q}_1 = \mathbb{Q}_2\mathbb{R}_2$ . This procedure continues for all  $n$  matrices, resulting in that the monodromy matrix can be written as

$$\mathbb{M}(n_0, n) = \mathbb{Q}_n\mathbb{R}_n\mathbb{R}_{n-1} \dots \mathbb{R}_1 = \mathbb{Q}_n\mathbb{R}, \quad (2.23)$$

where  $\mathbb{R}$  is also an upper triangular matrix, whose diagonal elements are the sums of the diagonal elements of the  $\mathbb{R}_k$ -matrices. One can then show that [29, 30] the Lyapunov exponents can be computed as

$$\lambda_i = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n \ln |R_{k,ii}|. \quad (2.24)$$

This method is accurate in computing the Lyapunov exponents where as many iterations as is needed can be used. However, the algorithm introduces errors when computing FTLEs. To combat this error, T. Okushima

[31] introduced a correction scheme based on the LR method for finding the eigenvalues of the  $\mathbb{M}^T \mathbb{M}$  matrix. The LR method decomposes a matrix into a lower and upper triangular matrix  $\mathbb{A}_1 = \mathbb{L}_1 \mathbb{R}_1$ . Then, a new matrix is produced through  $\mathbb{A}_2 = \mathbb{R}_1 \mathbb{L}_1$  which has the same eigenvalues as  $\mathbb{A}_1$ . By iterating this algorithm,  $\mathbb{A}_n$  will converge to a lower-triangular matrix as  $n \rightarrow \infty$ . The eigenvalues can then be read off from the diagonal. The method presented in [31] is a combination of the QR method and the LR method.

## 2.4 Time-delay embedding

When observing the evolution of a chaotic dynamical system, one may only have access to a single, 1-dimensional observable of the system, say  $\varphi(\mathbf{x}(t))$ . However, due to Takens' embedding theorem, the history of this observable is enough to reconstruct the dynamics of the system. Suppose that  $\varphi(\mathbf{x}(t))$  is measured at a discrete set of delays, and construct a dynamical system out of these delayed observations,

$$\begin{aligned} y_1(t) &= \varphi(\mathbf{x}(t)), \\ y_2(t) &= \varphi(\mathbf{x}(t - \tau)), \\ &\vdots \\ y_k(t) &= \varphi(\mathbf{x}(t - (N - 1)\tau)). \end{aligned} \tag{2.25}$$

As long as a multiple of the delay  $\tau$  does not coincide with a period of the system, and the observable  $\varphi$  fulfils some mild regularity conditions [32], Takens' embedding theorem states that, if  $k > 2d_A$ , the dynamical system in Eq. 2.25 is an embedding of the underlying dynamical system in  $\mathbb{R}^k$ . Here,  $d_A$  is the box-counting dimension of the attractor of the chaotic dynamical system [10].

Takens' embedding theorem does not impose any further conditions on the delay  $\tau$ , but some delays may be more efficient than others. A common way of finding a good time delay is to find the first local minimum of the average mutual information between delayed observations as the time delay is increased [32]. Mutual information is a concept borrowed from information theory, where the mutual information between two variables quantifies how much information about one of the variables is obtained by observing the other. Here, 'information' is quantified through information entropy [33],

where gaining information results in a decrease in the entropy. By finding the first local minimum of the average mutual information among the time-delayed variables, the amount of information obtained from observing all variables is maximised.

### 3 Lagrangian coherent structures

The study of dynamical systems relies on identifying attractors and repellers in phase space which determine the long-term behaviour of the dynamics. Oftentimes, identifying such structures is not easy, for example when the dynamical system is not defined through a known set of differential equations. One may instead search for structures that organise phase space by being coherent structures that persist on significant time scales. Coherent structures can be defined in several ways, such as being regions with persistent vorticity [34, 35], regions where fluid particles display distinct statistics [36], regions through which diffusive transport is extremised [37] or regions with distinct FTLE distributions [38, 39]. Lagrangian coherent structures (LCS) [23] have received much attention in the past decade [40, 41, 42, 43], defined as structures that locally maximises/minimises deformations of phase-space elements over some time span  $[t_0, t]$ . We proceed by presenting this definition. LCS have been used to study mixing in fluids [44], detecting attractors and repellers in inertial particle dynamics [45], predicting pollution transport patterns [46], and for optimal navigation in unsteady flows [18]. Since these structures are Lagrangian, they are made up of trajectories of the dynamical system. Hence, no trajectories can cross through the LCS, making them transport barriers.

#### 3.1 Definition

Consider a dynamical system with evolution rule  $F(\mathbf{x}, t)$ . Let  $M_0$  be some smooth surface of initial positions in phase space at time  $t_0$ . Applying the evolution rule to each point in  $M_0$  will result in an evolved surface  $M_t = F(M_0, t)$ . Letting  $\mathbf{n}_0(\mathbf{x})$  be an infinitesimal normal vector of the initial surface at  $\mathbf{x} \in M_0$ , we can evolve the vector using the monodromy matrix of the dynamical system. The evolved vector  $\hat{\mathbf{n}}_0 = \mathbb{M}\mathbf{n}_0(\mathbf{x})$  is in general not the

normal vector of the evolved surface, which is instead given by  $\mathbf{n}_t = \mathbb{M}^\top \mathbf{n}_0(\mathbf{x})$  [25].

As previously mentioned, the definition of the LCS are the structures which maximally/minimally deforms phase-space elements. If the dynamics in phase space near  $\mathbf{x}$  causes elements to be normally repelled from the surface, the evolved initial normal vector  $\hat{\mathbf{n}}_0$  will become tangential with the evolved surface's normal vector. If, on the other hand, the dynamics near  $\mathbf{x}$  shears elements along the surface,  $\hat{\mathbf{n}}_0$  will align more in a direction orthogonal to the evolved surface's normal vector. Normalising these vectors, the component of  $\hat{\mathbf{n}}_0$  orthogonal with  $\mathbf{n}_t$  is defined as the tangential shear, whereas the component tangential with  $\mathbf{n}_t$  is the normal repulsion.

The LCS can now be defined as follows. A hyperbolic LCS is defined as a surface  $M_0$  whose normal repulsion is locally maximal or minimal over the time span  $[t_0, t]$  for all  $\mathbf{x} \in M_0$ . If the normal repulsion is maximal, the hyperbolic LCS is a repelling hyperbolic LCS, whereas the surface associated with the minimal normal repulsion is an attracting hyperbolic LCS. Similarly, a surface which maximises or minimises the tangential shear is called a parabolic LCS. An elliptic LCS is a parabolic LCS, but where the surface  $M_0$  is closed.

It has been shown [47] that for a surface to be an LCS, it must be orthogonal to a vector field associated with the right singular vectors of the monodromy matrix. For hyperbolic LCS, this is the vector field associated with the largest/smallest singular values for repelling and attracting LCS respectively. In the case of parabolic and elliptic LCS, the orthogonal vector field is given by [47]

$$\boldsymbol{\eta}_\pm(\mathbf{x}_0, t) = \sqrt{\frac{\sigma_1}{\sigma_1 + \sigma_3}} \mathbf{v}_1 \pm \sqrt{\frac{\sigma_3}{\sigma_1 + \sigma_3}} \mathbf{v}_3. \quad (3.1)$$

This condition provides a means of detecting LCS when the dynamical system under consideration is in  $\mathbb{R}^3$ , as described below.

LCS are associated with the FTLEs of the dynamical system. For example, repelling hyperbolic LCS coincides with ridges of positive FTLE, while maximally shearing parabolic and elliptic LCS coincide with ridges where  $|\sigma_1 - \sigma_3|$  is maximal [47]. Importantly, however, for ridges of FTLEs to be LCS, the ridge must be orthogonal to the associated vector field.

### 3.2 Finding the structures

To find the LCSs of a dynamical system, one must first compute the monodromy matrix using the variational equation (Eq. 2.7). Once the deformation matrix is known over a dense grid of initial conditions in the domain of interest, the vector field associated with the LCS can be obtained and interpolated. From this point, if the LCS is in  $\mathbb{R}^3$ , we can use the fact that the LCS must be orthogonal to a known vector field as follows, where we use the parabolic LCS as an example. Introduce a plane with normal vector  $\mathbf{n}_\perp$ . Now, if the plane passes through the elliptic LCS, the cross product between  $\mathbf{n}_\perp$  and  $\boldsymbol{\eta}_\pm(\mathbf{x}_0, t)$  will produce a vector that is necessarily tangential with the LCS. Hence, we can construct a dynamical system that will have the LCS as an attractor or repeller:

$$\dot{\mathbf{x}} = \mathbf{n}_\perp \wedge \boldsymbol{\eta}_\pm(\mathbf{x}, t). \quad (3.2)$$

Closed trajectories of this method maps out elliptic LCS. The same procedure, using the maximal or minimal FTLV can be used to detect hyperbolic LCS. In computing the LCS, one must choose a grid that is sufficiently dense. Additionally, the considered time interval  $[t_0, t]$  over which the monodromy matrix for each grid point is integrated should be chosen using the fact that the FTLV converge to be normal to the LCS exponentially with a rate proportional to  $t^{-1} \ln \sigma_1 / \sigma_2$  [17].

## 4 Artificial neural networks

The past decade has seen an explosion in the interest in artificial neural networks due to their demonstrated ability to produce state-of-the-art results in classification [48, 49], generative [50, 51], and time-series prediction tasks [52]. Arguably one of the primary reasons for this success lies in the universal approximation theorem [53], which shows that any sufficiently smooth function can be approximated by an ANN, given that the network is wide enough. However, even narrow networks can model complex functional relationships if *hidden layers*, i.e. intermediate, consecutive computational layers between input and output, are introduced, resulting in a deep neural network (DNN). Such networks display exponential expressivity [9], which allows them to unfold intricate data manifolds at an exponential rate per

---

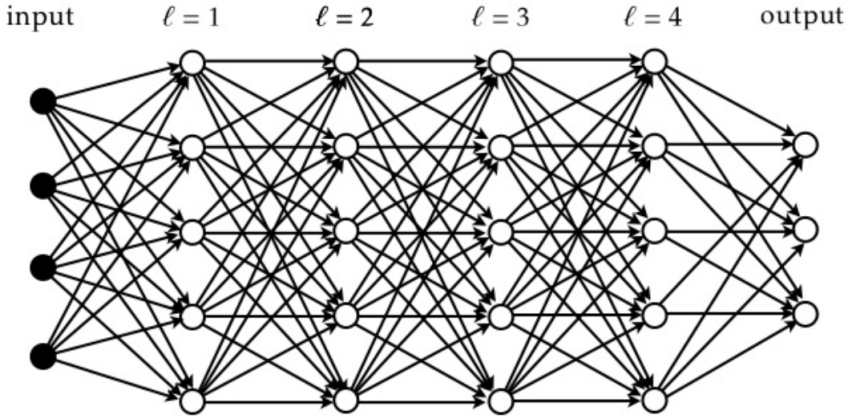
layer. This results in that DNNs are able to identify highly intricate features in datasets. In Paper A, we investigate how this exponential expressivity is exploited by the network.

While DNNs have shown promising results in applications, we still lack a complete understanding of what and how the network learns. As for the “what” question, the above exposition leads to the question “if the network can identify such complex features, how do we know what it has actually learnt?”. This question is at the heart of the blackbox problem of DNNs [54, 55, 56] – the fact that we cannot easily read off what information has been captured by a trained network from its parameters. By testing the network against an independent test dataset, we can verify whether the network has captured the global features contained in the training dataset that are relevant for the task, but this does not exclude the possibility that the network has learnt additional, spurious relationships. This is an important question as it can compromise the safety of using DNNs in high-stake situations such as autonomous driving, medical diagnostics or policy making [57, 58, 59].

The question of how the network learns is also important, as answering it can help optimise training, understand the limitations of the network, and allow us to design learning algorithms that minimise spurious outcomes. As nonlinear, high-dimensional functions consisting of a large number of parameters, they are generally difficult to analyse.

Dynamical-systems theory turns out to be a useful tool to study DNNs. By studying the propagation of input signals through the network, progress has been made in understanding how a network should be initialised so that information carried by the input signals is mapped faithfully to the output, making training more efficient [60]. The exponential expressivity of neural networks was first described in terms of the maximal singular value of the monodromy matrix of the neural network [9], and the information-propagation depth, that is, the number of layers information about the input signal can propagate before it dissipates, was quantified using the Lyapunov time of the network [61].

By viewing the layers of a DNN as a time index, the propagation of the signal can be analysed as a discrete dynamical system. One should keep in mind that the mapping between each consecutive layer changes and may even change dimension. For recurrent networks such as Hopfield networks [62, 63], autoencoders [64], and reservoir computers [65], the interpretation as dynamical systems is more straightforward, as the output of the network



**Figure 4.1:** Example schematic of a feed-forward neural network architecture. The shown network has an input layer with 4 neurons, 4 layers with 5 neurons per layer, and an output layer with 3 neurons. Reproduced from [62] with permission.

is used as an input for the next iteration. Hence, the full network serves as the map, and the discrete dynamics are evolved by applying the map and feeding the output back as an input.

In what follows, the network architectures which have been studied in this work will be presented. These are the feed-forward architecture, made of layers of computational nodes that propagate an input signal forward, and the echo-state network. The echo-state network is a recurrent neural with the primary difference being that only the parameters associated with the output layer are trained, while the remaining parameters remain fixed.

## 4.1 Feed-forward neural networks

One of the most basic realisations of a neural network is the feed-forward architecture, consisting of consecutive layers of neurons, each receiving inputs from the previous layer and propagating their outputs to the next layer, as shown in Fig. 4.1. The computation performed by each neuron is



given by [62]

$$\mathbf{x}_i^{(\ell)} = g\left(b_i^{(\ell)}\right), \quad b_i^{(\ell)} = \sum_{j=1}^{N_i} w_{ij}^{(\ell)} x_j^{(\ell-1)} - \theta_i^{(\ell)}. \quad (4.1)$$

Here,  $x_i^{(\ell)}$  is the output of the  $i$ :th neuron in the  $\ell$ :th layer,  $b_i^{(\ell)}$  is referred to as the local field of the neuron,  $g$  is a nonlinear<sup>1</sup> function referred to as the activation function,  $w_{ij}^{(\ell)}$  is the weight connecting the  $j$ :th neuron in the preceding layer to the current neuron, and  $\theta_i^{(\ell)}$  is the threshold (or bias, if the sign is flipped) of the neuron. The weight matrix whose entries are  $w_{ij}^{(\ell)}$  is denoted  $\mathbb{W}^{(\ell)}$ . Setting the input neurons to  $\mathbf{x}^{(0)} = \mathbf{x}$ , where  $\mathbf{x}$  is some input signal, Eq. 4.1 provides the discrete dynamics with which the signal is propagated through the network. Since the map changes per layer, the dynamics of the signal propagation could be compared to a non-autonomous discrete dynamical system. However, since the map may also change in dimension, this analogy is not straightforward. Nevertheless, tools from dynamical-systems theory can be employed to analyse the neural networks, as is shown below.

#### 4.1.1 Supervised learning

In a supervised learning task, an input  $\mathbf{x}_\mu$  is associated with a ground-truth output  $\mathbf{y}_\mu$ , where  $\mu$  denotes the index of the input-output pair. The goal of the learning task is to select the network parameters that minimise some error function  $H = \sum_{\mu=1}^p \mathcal{L}(\mathbf{y}_\mu, \hat{\mathbf{y}}_\mu)$ , where  $\hat{\mathbf{y}}_\mu = \mathbf{x}_\mu^{(L+1)}$  is the output of the network and  $\mathcal{L}$  is a convex function referred to as the cost function with its minimum at  $\mathbf{y}_\mu = \hat{\mathbf{y}}_\mu$  for all  $\mu$ . A common choice of  $\mathcal{L}$  is the quadratic cost function using the Euclidian norm,  $\mathcal{L}(\mathbf{y}_\mu, \hat{\mathbf{y}}_\mu) = \frac{1}{2} \|\mathbf{y}_\mu - \hat{\mathbf{y}}_\mu\|^2$ , though many other choices exist [66]. One may also modify the energy function by for example introducing terms that penalise large parameter values, e.g. L1- or L2-regularisation [62]. With the energy function defined, the network parameters are updated through gradient descent:

$$\mathbb{W}_{t+1}^{(\ell)} = \mathbb{W}_t^{(\ell)} - \eta \nabla_{\mathbb{W}_t^{(\ell)}} H, \quad (4.2)$$

---

<sup>1</sup>If a linear activation function is used, a deep neural network is equivalent to a network without hidden layers, since the action of the network will be a linear transformation of the input.

with the thresholds updated similarly. Here, a time index  $t$  has been introduced, referring to the training epoch. The parameter  $\eta$  is called the learning rate. A large number of modifications to the standard gradient descent algorithm exists. Among the most standard extensions is the stochastic gradient descent, where noise is introduced to the update rule by training the network on a random subset of the training dataset every iteration. This is done so that the network is able to escape local minima, which the standard gradient-descent algorithm is prone to get stuck in. More sophisticated modifications exist, such as Adam [67], which uses an adaptive learning rate.

#### 4.1.2 Network initialisation and the unstable gradient problem

How should the network parameters be initialised? This question becomes particularly relevant when many hidden layers are used, because of the unstable gradient problem (UGP) which makes training difficult. The UGP can be understood through the lens of dynamical-systems theory as follows [62].

Consider the gradient of the energy function with respect to threshold  $\theta^{(\ell)}$ , and let us assume that we only have a single input pattern  $p = 1$  to simplify notation and since it is not relevant for the argument. Computing this gradient yields

$$\nabla_{\theta^{(\ell)}} H = \delta \mathbf{y}^\top \mathbb{D}^{(L)} \mathbb{W}^{(L)} \dots \mathbb{D}^{(\ell+1, \mu)} \mathbb{W}^{(\ell+1)} \mathbb{D}^{(\ell)} = \delta \mathbf{y}^\top \mathbb{M}(\ell, L) \mathbb{D}^{(\ell)}, \quad (4.3)$$

where  $\delta \mathbf{y} = \frac{d\mathcal{L}}{d\mathbf{y}}$  is the error vector (e.g. for the quadratic cost function,  $\delta \mathbf{y} = \mathbf{y} - \hat{\mathbf{y}}$ ), and  $\mathbb{D}^{(\ell)}$  is a diagonal matrix with elements  $D_{ij}^{(\ell)} = \frac{dg}{db} \Big|_{b=b_i^{(\ell)}} \delta_{ij}$ . As will be seen later,  $\mathbb{M}$  is the monodromy matrix of the dynamics of a signal propagating forward through the network. If we assume that both  $\mathbb{D}^{(\ell)}$  and  $\mathbb{W}^{(\ell)}$  have entries that are independent and, respectively, identically distributed<sup>2</sup> with finite variance, the following non-random limit exists according to the Furstenberg theorem (Section 2.2.1),

$$\lambda_1 = \lim_{(L-\ell) \rightarrow \infty} \frac{1}{L-\ell} \langle \ln(\sigma_1) \rangle, \quad (4.4)$$

where  $\sigma_1$  is the maximal singular value of  $\mathbb{M}$ , and the average is taken over the ensemble of  $\mathbb{M}$ -matrices. The interpretation is that the magnitude of

<sup>2</sup>In [9] it was shown that this assumption on  $\mathbb{D}^{(\ell)}$  is a good approximation for randomly initialised, infinitely wide networks.

the error vector has an average exponential growth rate per layer given by the maximal Lyapunov exponent  $\lambda_1$ . Hence, if  $\lambda_1 > 0$ , the error vector will explode in magnitude given sufficiently many layers, or vanish in magnitude if  $\lambda_1 < 0$ . This is the essence of the UGP. One would therefore like to initialise the network so that  $\lambda_1 \sim 0$  so that the error vector neither explodes nor vanishes [60]. As we will see later, the Xavier [68] and He [69] initialisation schemes achieves this, at least for wide networks.

### 4.1.3 Network dynamics

It is not only the backward propagation of errors that is affected by the UGP. Consider two similar input signals  $\mathbf{x}$  and  $\mathbf{x}'$  where  $\delta \mathbf{x}_0 = \mathbf{x} - \mathbf{x}'$  and  $\|\delta \mathbf{x}_0\| \ll 1$ . The evolution of the magnitude of  $\delta \mathbf{x}_0$  is described by the linearised network dynamics,

$$\delta \mathbf{x}(\ell) \approx \mathbb{D}^{(\ell)} \mathbb{W}^{(\ell)} \dots \mathbb{D}^{(1)} \mathbb{W}^{(1)} \delta \mathbf{x}_0 = \mathbb{M}(0, \ell) \delta \mathbf{x}_0. \quad (4.5)$$

Here, we see that  $\mathbb{M}$  is in fact the monodromy matrix of the discrete dynamics of the neural network<sup>3</sup>. If  $\lambda_1 < 0$  for large  $L$ , the difference between the two input signals disappear (i.e. they become completely correlated). If on the other hand  $\lambda_1 > 0$ , the input signals will become completely decorrelated, and the information contained in their correlation disappears. The characteristic depth that information can travel in a feed-forward neural network was computed in [61], and is closely related to the Lyapunov time of the signal dynamics. When  $\lambda_1 = 0$ , this depth diverges.

The chaotic dynamics displayed by neural networks is the source of their exponential expressivity. As shown by Poole et al. [9], an infinitely wide network using the tanh activation function initialised in the chaotic regime is able to attenuate the curvature of a data manifold with a rate associated with the maximal Lyapunov exponent of the system. Due to the boundedness of the nonlinear activation function, however, the data manifold may also be folded by the network dynamics. Hence, the network dynamics are similar to chaotic dynamical systems where volumes in phase space are stretched and folded over time.

This result was derived in the infinite-width case, where the maximal Lyapunov exponent may be approximated using mean-field theory, as was

---

<sup>3</sup>In the machine-learning literature [9, 60, 61], the matrix  $\mathbb{M}(0, L)$  is referred to as the input-output Jacobian.

shown in [60, 62]. In this case, the maximal Lyapunov exponent is given by

$$\lambda_1 \approx \lim_{N \rightarrow \infty} \frac{1}{2} \ln(GN\sigma_W^2), \quad (4.6)$$

where  $\sigma_W^2$  is the variance of the entries of the weight matrices  $\mathbb{W}^{(\ell)}$  and  $G = \langle [D_{ii}^{(\ell)}]^2 \rangle$ . Using the tanh activation function, the variance of the elements of the  $\mathbb{D}^{(\ell)}$ -matrices is shown to converge rapidly over layers to a fixed value dependent on  $\sigma_W^2$  and the variance of the thresholds. In Section 6.2, an expression for the maximal Lyapunov exponent is derived for finite widths. This expression is shown to equal Eq. 4.6 as the large- $N$  limit is taken.

#### 4.1.4 Infinite-width limits

As mentioned before, since neural networks are nonlinear, high-dimensional functions, they are in general difficult to analyse. However, significant progress has been made when taking the infinite-width limit, where the evolution of the network during training becomes deterministic. Different limits exist depending on how the network parameters are scaled. In Eq. 4.1, the so-called standard [70] scaling is used. If the local field is instead computed as

$$b_i^{(\ell)} = \frac{1}{\sqrt{N_\ell}} \sum_{j=1}^{N_\ell} w_{ij}^{(\ell)} x_j^{(\ell-1)} - \theta_i^{(\ell)}, \quad (4.7)$$

i.e. by introducing the scaling  $1/\sqrt{N_\ell}$ , the infinite-width limit will correspond to the neural-tangent kernel (NTK) limit [71]. The NTK limit has been popular in recent years, as it was shown that the evolution of a neural network during training through gradient descent is given by a linear differential equation with constant parameters. The parameters of the differential equation are dependent on the network parameters, which, although the network is trained, are shown to barely change. This phenomenon is referred to as lazy-learning [72]: the changes in the individual network parameters are negligible, but the collective effect on the network is significant. Using the standard scaling, the infinite-width limit also results in lazy-learning [70], as is confirmed by results in Paper A. If the weights in Eq. 4.1 are instead scaled by  $N_\ell^{-1}$ , referred to as mean-field scaling [73], lazy-learning is not induced in the infinite-width limit. Instead, the network evolves according to a partial differential equation of diffusion-type. This type of learning, where

the network parameters change significantly, is known as feature-learning. Limits exist for other scalings, but each of them either fall into either the lazy-learning or feature-learning category [70].

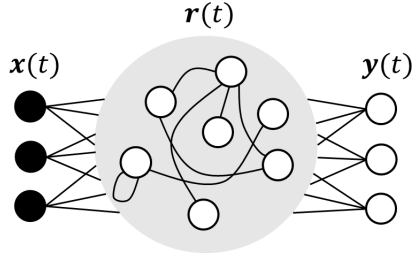
## 4.2 Echo-state networks

The echo-state network (ESN) [65] is a type of recurrent neural network where only the parameters associated with the output layer is trained, while the remaining parameters remain fixed from initialisation. ESNs belong to a branch of machine learning referred to as reservoir computing. In reservoir computing, a large network of nonlinearly interacting computational nodes – the ‘reservoir’ – is driven by some input signal. The connections between these nodes form loops, making the dynamics recurrent and providing the reservoir with a dynamic memory. Under certain conditions, the dynamics of the reservoir can synchronise with the input signal, making the dynamics of the reservoir a high-dimensional embedding of the input dynamics. Linear combinations of the reservoir states can then be used to, for example, predict the dynamics of the input signal for time-series prediction tasks. The interest in reservoir computing can be contributed to two primary factors: (1) reservoir computing algorithms have shown state-of-the-art performance compared to recurrent neural network, while being much easier to train since only the parameters associated with the output are trained [74, 75, 76], and (2) reservoir computing can be implemented in physical systems with a large number of degrees of freedom that interact nonlinearly, such as optical node arrays, analog electric, circuits laser cavities [77], and, intriguingly, a bucket of water [78]. Implementing reservoir computing in physical systems can yield highly efficient analog computers, operating in a completely different way from standard computers.

An ESN is a particular realisation of a reservoir computer (Fig. 4.2). The network dynamics during training is given by

$$r_i(t+1) = g \left( \sum_{j=1}^N a_{ij} r_j(t) + \sum_{k=1}^n w_{ik}^{(\text{in})} x_k(t) \right), \quad (4.8a)$$

$$y_i(t+1) = \sum_{j=1}^N w_{ij}^{(\text{out})} f(r_j(t+1)) \quad (4.8b)$$



**Figure 4.2:** Schematic of an echo-state network with three input nodes, a reservoir layer, and three output nodes.

where the matrices  $\mathbb{A}$  and  $\mathbb{W}^{(\text{in})}$  have randomly initialised elements  $a_{ij}$  and  $w_{ij}^{(\text{in})}$  respectively,  $r_i(t)$  is the  $i$ :th reservoir state at time  $t$  and  $x_i(t)$  is the input signal that drives the reservoir dynamics during training. The activation function  $g$  is usually selected to be the tanh activation function, although other choices are possible as long as the activation function is monotonically increasing and bounded [65]. The function  $f$  is applied to the reservoir states before they are projected to output space using the output weights  $\mathbb{W}^{(\text{out})}$ . While  $f$  is usually selected to be the identity function, other choices exist [76]. It should also be noted that, while Eq. 4.8b is the standard way of projecting the reservoir states to output space, modifications such as the Lu readout [79] have been introduced. When using the Lu readout, in addition to projecting the reservoir states  $r_i(t)$ , the squared reservoir states  $r_i^2(t)$  are also projected, making the output weight matrix  $\mathbb{W}^{(\text{out})}$  have the dimension  $n \times 2N$ , where  $n$  is the output dimension. This is done to break symmetries in the reservoir dynamics, as these symmetries might otherwise cause the network to learn mirrored versions of the input time series.

When using the ESN for time-series prediction, once the ESN has been trained, the input signal is replaced by the output  $y_i(t)$  to form the autonomous dynamics

$$r_i(t+1) = g \left( \sum_{j=1}^N a_{ij} r_j(t) + \sum_{k=1}^n w_{ik}^{(\text{in})} y_k(t) \right), \quad (4.9a)$$

$$y_i(t+1) = \sum_{j=1}^N w_{ij}^{(\text{out})} f(r_j(t+1)), \quad (4.9b)$$

so that the ESN can be used to generate predictions of the input time series. Since the map remains the same for each iteration, the ESN is a discrete dynamical system.

#### 4.2.1 Echo-state property

For it to be possible to use the ESN for time-series prediction, the parameters of the network must be chosen so that the reservoir dynamics synchronise with the input dynamics. This is called the echo-state property. The echo-state property is the requirement that, given two different initial conditions of the same network driven by the same input signal, the reservoir-state dynamics will synchronise in the asymptotic limit. That is, given  $\mathbf{r}(0) = \mathbf{r}_0$  and  $\mathbf{r}'(0) = \mathbf{r}'_0$ , the following will hold for an ESN with the echo-state property:

$$\lim_{t \rightarrow \infty} \|\mathbf{r}(t) - \mathbf{r}'(t)\| = 0. \quad (4.10)$$

It is clear that this condition is fulfilled if the ESN dynamics has a maximal Lyapunov exponent that is negative. Importantly, it is the *driven* dynamics (Eq. 4.8a) that should have a negative Lyapunov exponent. Once the network has been trained and the driving signal  $\mathbf{x}(t)$  in Eq. 4.8a is replaced by  $\mathbf{y}(t)$ , the Lyapunov exponent of the autonomous dynamics should match that of the predicted time series.

#### 4.2.2 Initialisation and hyperparameters

When designing an ESN, several design parameters must be selected. Among these are the dimension of the reservoir  $N$ , connection sparsity  $s$  (in this work, if  $s = 0$ , the weight matrix  $\mathbb{A}$  is a zero-matrix, whereas  $s = 1$  implies that  $\mathbb{A}$  is dense), topology of the connection matrix, scale of  $\mathbb{A}$  (e.g. spectral radius, maximal singular value, variance of matrix entries  $\sigma_{\mathbb{A}}^2$ ), scale of the input matrix, and distribution of the two matrices. Furthermore, parameters external to the reservoir architecture must be selected, such as the rate at which a continuous input signal is sampled  $\delta t$ , the total training time  $\delta t T$ , and the choice of hyperparameters used in training. Ridge regression (Tikhonov regularization [80]) is used in most recent works to train the output weights, where the ridge parameter is the relevant hyperparameter. As of yet, we lack insight on how these parameters should be selected to optimally train the ESN and tune its dynamical properties. Several works have focused on how

the network-specific parameters affect performance. It has been shown that increasing the reservoir dimension monotonically increases the prediction performance [81]. Several topologies have been investigated without showing significant differences in performance [76, 82]. The spectral radius has been extensively discussed. In the original paper of Jaeger [65] where the ESN was introduced, it was suggested that the spectral radius should be chosen to reflect the time scale of the input signal, e.g. a larger spectral radius should be chosen for slower dynamics. In general, the design of ESNs is based on heuristics, which is the motivation for Paper B, where we investigate why some parameter choices work while others do not.

### 4.2.3 Training the echo-state network

To train the ESN for time-series prediction, the target values  $\mathbf{t}(t)$  are set to be equal to the input time series. Then, one minimises an energy function  $H = \sum_{t=1}^T \|\mathbf{t}(t) - \mathbf{y}(t)\|^2$ , where  $T$  is the number of iterations during training. This is usually [76, 83] done through ridge regression

$$\mathbb{W}^{(\text{out})} = \mathbb{Y}\mathbb{R}^\top (\mathbb{R}\mathbb{R}^\top + k\mathbb{I})^{-1}. \quad (4.11)$$

Here,  $\mathbb{Y}$  is a matrix whose columns are given by  $\mathbf{y}(t)$ , and  $\mathbb{R}$  is a matrix whose columns are given by  $\mathbf{r}(t)$ . The ridge parameter  $k$  is introduced to combat overfitting. In Paper B, we show that if this parameter is set too large, it may prevent the ESN from predicting chaotic dynamics.

## 5 Microswimmers and transport barriers

The dynamics of small, self-propelled particles, known as microswimmers, are studied in many contexts, ranging from the modelling of plankton and bacteria [84, 85], to artificial particles such as Janus rods [86]. Microswimmers range in sizes from micro- to nanometers [87], and hence operate in the low Reynolds regime where viscosity dominates. In many of the settings where microswimmers are studied, one seeks to find optimal navigation strategies to achieve certain goals, such as efficient upward migration against gravitation [88], or evasion of predators [89]. The navigation strategy is affected by the motion of the fluid in which the microswimmer is submerged, as it may give rise to transport barriers which prevent it from reaching regions



in the flow [24, 90]. The shape of the particle may also affect the dynamics: non-spherical, rod-like particles have been shown to cluster and preferentially align [91, 92, 93]. In an on-going project, we study transport barriers in the phase space of the deterministic dynamics of a self-propelled spheroidal microswimmer, with the aim to understand how introducing random motion to the swimming dynamics may help the microswimmer to escape through the barriers. In this section, we present the microswimmer model and flow used in the project.

## 5.1 The model

We consider a 2-dimensional spheroidal particle. Its shape is determined by a shape factor

$$\Lambda = \frac{\lambda^2 - 1}{\lambda^2 + 1}, \quad (5.1)$$

where  $\lambda$  is the aspect ratio between the semi-axis in the swimming direction and the perpendicular semi-axis. Thus, for example, if  $\Lambda = 0$ , the swimmer is a sphere. The swimmer has a constant swimming speed  $v_p$ . The size of the particle is assumed to be small enough so that its swimming does not affect the surrounding fluid, and its dynamics are completely dominated by the flow. Hence, if the swimming speed is set to zero, the swimmer will move as a tracer. The phase space of the microswimmer is 3-dimensional, because its orientation is an additional degree of freedom. The orientation is affected by the vorticity of the flow, and, if the particle is non-spherical, the flow strain due to Jeffrey torque [94]. Thus, the dynamics of the microswimmer is

$$\dot{\mathbf{x}} = \mathbf{u}(\mathbf{x}, t) + v_p \hat{\mathbf{n}}, \quad (5.2a)$$

$$\dot{\hat{\theta}} = \frac{1}{2} \omega_f \wedge \hat{\mathbf{n}} + \Lambda \hat{\mathbf{n}} \wedge \mathbb{S}(\mathbf{x}, t) \hat{\mathbf{n}}, \quad (5.2b)$$

where  $\mathbf{u}(\mathbf{x}, t)$  is the fluid flow,  $\hat{\mathbf{n}}$  is the orientation of the swimmer,  $\omega_f = \nabla \wedge \mathbf{u}$  is the vorticity, and  $\mathbb{S}$  is the strain rate tensor. The strain rate tensor is the symmetric part of the spatial gradient of the flow [25]. This model has been widely used to model plankton such as copepods [88, 92, 93, 95], with additional terms to account for gyrotaxis which we do not consider here.

## 5.2 The flow

We study the swimmer in the Taylor-Green vortex (TGV) flow [96], which is a solution to the Navier-Stokes equation in Cartesian coordinates. The TGV flow shares some of the properties of statistically uniform isotropic turbulence, and is often studied as a model flow for microswimmers [97, 98, 99]. We study the steady TGV flow, which is given by

$$u_x(x, y) = u_0 \cos\left(\frac{x}{L}\right) \sin\left(\frac{y}{L}\right), \quad (5.3a)$$

$$u_y(x, y) = -u_0 \sin\left(\frac{x}{L}\right) \cos\left(\frac{y}{L}\right), \quad (5.3b)$$

where  $u_0$  is the maximal flow speed, and  $L$  is the characteristic length scale of the flow. When studying the dynamics of the microswimmer in the TGV, we nondimensionalise the dynamics using the dimensionless variables  $\hat{\mathbf{x}} = \mathbf{x}/x_0$  and  $\tau = t/t_0$ , where  $x_0 = L$  and  $t_0 = L/u_0$ . In performing this nondimensionalisation, the swimming speed is also rescaled, resulting in the dimensionless swimming speed  $v_s = v_p/u_0$ .

PART II  
PRESENT WORK

## 6 Finite-time Lyapunov exponents of deep neural networks

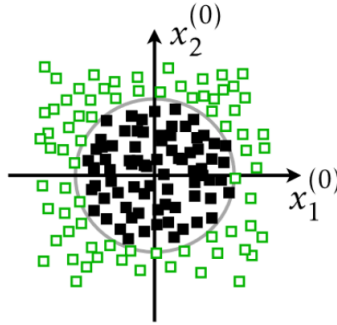
Deep neural networks display exponential expressivity as discussed in Section 4. The expressivity allows the network to disentangle complicated data manifolds through its chaotic signal-propagation dynamics. For this to be possible, the network must have a positive finite-time Lyapunov exponent (FTLE). These results, derived by Poole et al. [9], deals with randomly initialised networks of infinite width. In this limit, the FTLE becomes constant and input independent, as shown by the mean-field result (Eq. 4.6). If the network is instead of finite width, the FTLE follows a distribution. Initially, this distribution is centred around the infinite-width limit according to some random-matrix distribution<sup>1</sup>. An interesting question, which is studied in Paper A, is what this distribution looks like once the network is trained. What structures can be discerned in the distribution that reflects what the network has learnt? The main question of the paper is how a deep neural network makes use of its exponential expressivity to learn classification tasks.

In order to analyse this, we construct a simple 2-dimensional, binary classification task (Fig. 6.1), where coordinates that lie inside a circle of radius 1 have assigned target value -1, and the ones outside have target value 1. This classification task allows us to visualise the FTLE distribution. Using this dataset, we compute the FTLE for each input and study the effects of changing the width and depth of the network. Our analysis is also applied to a more realistic classification task, the MNIST dataset of handwritten digits [100]. We find that the conclusions made for the simpler classification task transfers to this task as well, although the input space is  $28 \times 28$ -dimensional.

What can we expect the distribution to look like? Since the network must learn to distinguish between classes, a small perturbation that move an input coordinate to a different class should yield a significant change in the output of the network. Hence, there should be positive exponents at the boundary between classes. Furthermore, the direction in which the largest rate of separation occurs (i.e. the maximal Lyapunov vector) ought to be orthogonal

---

<sup>1</sup>The large-deviation form of this distribution can be interesting to study in order to understand how likely unstable-gradient problems are to arise as a function of network width.



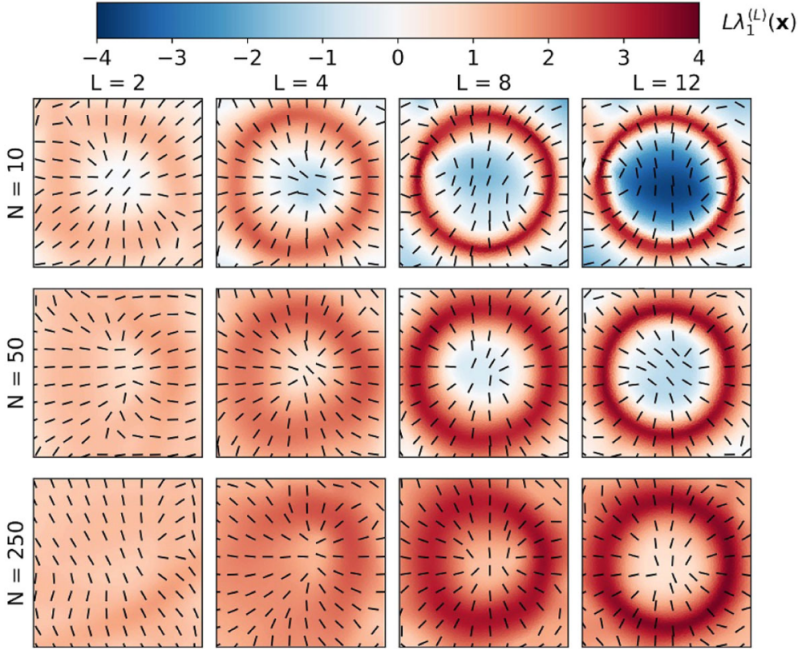
**Figure 6.1:** 2-dimensional classification task, where the data points marked by a green square have target value  $y_\mu = 1$ , while those marked by a black square have target value  $y_\mu = -1$ . Figure taken from Paper A with permission.

to the boundary between classes.

How is the distribution affected by architecture? We know that a deeper network has greater expressivity, meaning that smaller structures in the input data manifold can be resolved. This could yield finer structures in the FTLE distribution. As for the width, we know that a network whose width tends to infinity may enter the lazy-learning regime. How is this reflected in the distribution?

## 6.1 Learning regimes

Figure 6.2 shows the FTLE distribution of networks trained on the 2-dimensional classification task for different widths  $N$  and depths  $L$ . Importantly, what is shown is  $L\lambda_1^{(L)}(\mathbf{x})$  and not just the maximal FTLE  $\lambda_1^{(L)}$ . This quantity shows how much the magnitude of a perturbation is changed across the entire network, as the FTLE is the average rate of separation *per layer*. This quantity turns out to converge to a fixed value once  $L$  becomes large enough, because the network only needs to separate the data on a magnitude of order 1 (the output should be either 1 or -1). If  $\lambda_1^{(L)}(\mathbf{x})$  had been used, the magnitude of the FTLE field would have decreased for large  $L$ , because the same amount of separation should be performed for a larger number of layers, meaning that the contribution of each layer becomes less significant. This turns out to be an important detail when considering the  $L \rightarrow \infty$  limit, which we will discuss later in Section 6.1.1. Another important point is that the FTLE



**Figure 6.2:** FTLE distribution in input space for different widths  $N$  and depths  $L$  of fully connected feed-forward neural networks trained on the dataset from Fig. 6.1. The heatmap shows  $L\lambda_1^{(L)}(\mathbf{x})$  at different input coordinates, and the dashed lines are the associated FTLVs. Figure reproduced from Paper A with permission.

$\lambda_1^{(L)}(\mathbf{x})$  is computed up until the last hidden layer  $L$ , and not until the output layer at  $L + 1$ . This is done so that the action of the hidden layers can be discerned. Had  $\lambda_1^{(L+1)}(\mathbf{x})$  been used, the result would have not been affected by changing width and depth. The dashed lines are the maximal finite-time Lyapunov vectors (FTLV) associated with the FTLEs. No direction is assigned to these, as the vectors are only defined up to a sign. The result shows that there appear to be two distinct parameter regimes: for large  $L$  and small  $N$ , there are strong ridges at the boundary between the two classes, whereas for large  $N$  and small  $L$ , no structure can be discerned. We refer to these two regimes as the ridge-learning and random-embedding regimes, respectively.

### 6.1.1 Ridge learning

In the ridge-learning regime, the network creates basins of attraction for each class, separated by a ridge of positive exponents. We see that the maximal FTLV are orthogonal to the ridges, showing that these ridges can be compared to hyperbolic Lagrangian coherent structures (LCS). As we discuss in the paper, one should keep in mind that the dimension may change per layer in a neural network, while LCSs are usually defined in dynamical systems where the dimension remains constant between time steps. Nevertheless, the observed structures reveal how the network has learnt to partition the input space such that it solves the classification task.

Studying Fig. 6.2, we see that in the case when  $N = 10$  and  $L = 12$ , there is a weak ridge situated at the upper left corner, away from the strong circular ridge. This ridge is a spurious structure not shared by other realisations of the same network, as opposed to the strong ridge structure that appears for any realisation. It appears the network is sensitive to changes in a region of input space that is within the same class. In this way, the weak ridges provides a way to understand how networks has learnt spurious structures, showing how this framework may be used to better understand what the network has learnt.

What happens as  $L$  keeps increasing? As mentioned above, the magnitude of  $L\lambda_1^{(L)}(\mathbf{x})$  eventually stabilises to a fixed value. This means that as  $L$  becomes large, the FTLE on the ridge converges to zero. Furthermore, we find that the ridge does not continue to shrink in width. The surrounding distribution keeps decreasing. Denoting the average FTLE within the circle (the same result is obtained if the average FTLE outside the circle is used) as  $\langle \lambda_1^{(L)}(\mathbf{x}) \rangle_c$ , we find the following relationship numerically:

$$L\langle \lambda_1^{(L)}(\mathbf{x}) \rangle_c \approx -C_c L + \log N, \quad (6.1)$$

where  $C_c \geq 0$  is some constant. Dividing by  $L$  and taking the large- $L$  limit while keeping  $N \ll e^L$ , this relationship suggests that

$$\lim_{L \rightarrow \infty} \langle \lambda_1^{(L)}(\mathbf{x}) \rangle_c = -C_c, \quad (6.2)$$

i.e. that the FTLE converges to some non-zero constant away from the ridge. Since the ridge does not continue to becoming narrower as  $L$  increases, the FTLE field will not be coordinate independent in the large- $L$  limit. Had the

network not been trained, this limit would have converged to a constant (we derive the expression for this constant in Section 6.2). Hence, training breaks ergodicity.

Why does the ridge maintain a finite width? The explanation may lie in the finite dataset. The finite sample size means that there is a smallest distance between sample points, which introduces a region where classification is arbitrary between the classes. The finite width of the ridge reflects the network uncertainty. In fact, as we show for the MNIST dataset, there is a strong positive correlation between uncertainty (quantified as information entropy, see definition in Paper A) and the FTLE. If we were to increase the sample size, the ridge does indeed become narrower. Letting the sample size and depth go to infinity, we would therefore have a ridge occupying a measure-zero set.

### 6.1.2 Random embedding

Equation 6.1 only holds while we have sufficiently small  $N$ , because as  $N > e^L$ , the average FTLE away from the ridge saturates to the value on the ridge so that the distribution becomes uniform, as shown in the case where  $N = 250$  and  $L = 2$  in fig. 6.2. We refer to this regime as the random-embedding regime, because the network embeds data without any clear structure. In this regime, we also find that it is sufficient to train the output layer of the network while keeping the remaining layers fixed from initialisation. This is possible because of Cover’s theorem [101]; the larger the dimension, the more likely it is that a random embedding of a dataset in that dimension will make samples from different classes linearly separable. Hence, once the input has been mapped to the output layer, the output neuron can discriminate between the classes with a linear decision boundary.

Since wide networks can immediately separate classes linearly, the network parameters do not have to change. This indicates that the network is in the lazy-learning regime. As discussed in Section 4.1.4, this is expected for the chosen scaling of the local fields. Whether the network would enter into the random-embedding regime, or remain in the ridge-learning regime if the mean-field scaling is used is an open question.



### 6.1.3 Parameter initialisation

We find that the results in Fig. 6.2 are independent of initialisation. That is, regardless of whether the network is initialised such that the average FTLE is positive or negative, the distribution will evolve to look like in the figure. This means that even if the network is initialised in the vanishing-gradient phase, the training dynamics will eventually make the network escape this phase. The deeper the network is in the vanishing-gradient phase, the longer it takes to escape. The network may also escape the exploding-gradient phase: if the limiting distribution is centred around zero, as is the case for small values of  $N$  and large values of  $L$ , the training can make the network escape this phase.

## 6.2 Lyapunov exponent for randomly initialised neural networks

While not part of Paper A, we present this result here as it may be interesting for future investigations of the network dynamics of feed-forward neural networks. We derive an expression for the maximal Lyapunov exponent of a deep neural network with randomly initialised parameters. The expression is valid for a network with constant width and piece-wise linear activation functions whose derivative takes the values 0 or 1, as is the case with the ReLU activation function and the hard-tanh activation function<sup>2</sup>. The derivation starts by considering the dynamics of an infinitesimal perturbation  $\delta \mathbf{x}^{(0)}$ , which evolves according to the linearised network dynamics. The perturbation at the  $L$ :th layer is then given by

$$\delta \mathbf{x}^{(L)} \approx \mathbb{D}^{(L)} \mathbb{W}^{(L)} \dots \mathbb{D}^{(1)} \mathbb{W}^{(1)} \delta \mathbf{x}^{(0)} = \mathbb{M}(0, L) \delta \mathbf{x}^{(0)}. \quad (6.3)$$

The matrices  $\mathbb{D}^{(\ell)}$  are diagonal with elements  $D_{ij}^{(\ell)} \delta_{ij} = \frac{d}{db} g(b_i^{(\ell)})$ . Hence, for the considered activation functions, these matrices take on the values 0 and

---

<sup>2</sup>The hard-tanh activation function is the piecewise linear approximation of the tanh function with

$$\text{Hardtanh}(x) = \begin{cases} -1, & x < -1 \\ x, & -1 \leq x \leq 1 \\ 1, & x > 1 \end{cases}$$

1. Poole et al. [9] showed that for an infinitely wide network, the distribution of the local fields  $b_i^{(\ell)}$  converges rapidly as  $\ell$  increases, making each matrix  $\mathbb{D}^{(\ell)}$ , and therefore also  $\mathbb{D}^{(\ell)}\mathbb{W}^{(\ell)}$ , independent and identically distributed. Hence, the diagonal matrices can be modelled as matrices whose diagonal elements are Bernoulli distributed with probability  $p$  of being 1 and  $1-p$  of being 0. Although this is shown for infinitely wide networks, we find numerically that this holds also for finite-width networks. The derivation is heavily based on the derivation by Newman [102] for products of random Gaussian matrices. In that paper, two assumptions are made about the matrices in the random product. Firstly, the matrices are independent and identically distributed. Secondly, the matrices must fulfil a certain symmetry; that is, for some matrix  $\mathbb{A}$ , one must have that  $\mathbb{Q}^\top \mathbb{A}^\top \mathbb{A} \mathbb{Q}$  follows the same distribution as  $\mathbb{A}^\top \mathbb{A}$ , where  $\mathbb{Q}$  is an orthogonal matrix. In our case, the first condition is fulfilled as previously stated. The second condition requires that  $[\mathbb{W}^{(\ell)}]^\top \mathbb{D}^{(\ell)} \mathbb{W}^{(\ell)}$  is rotationally symmetric, which it is, at least numerically.

The argument then goes as follows: consider the definition of the Lyapunov exponent in Eq. 2.3. In the discrete case, this definition can be rewritten as

$$\lambda_1 = \lim_{t \rightarrow \infty} \lim_{\|\delta \mathbf{x}_0\| \rightarrow 0} t^{-1} \sum_{k=0}^{t-1} \ln \frac{\|\mathbb{D}^{(k)} \mathbb{W}^{(k)} \delta \mathbf{x}(k)\|}{\|\delta \mathbf{x}(k)\|}. \quad (6.4)$$

Now, since the matrices in each summand are independent and identically distributed, and since the quantity in the logarithm is equally distributed regardless of orientation (following from the symmetry assumption), the maximal Lyapunov exponent may be evaluated as

$$\lambda_1 = \langle \ln \|\mathbb{D}^{(k)} \mathbb{W}^{(k)} \hat{\mathbf{n}}(k)\| \rangle, \quad (6.5)$$

where the law of large numbers has been invoked, and the average is computed over the distribution of  $\mathbb{D}^{(k)} \mathbb{W}^{(k)}$ -matrices. Here,  $\hat{\mathbf{n}}(k)$  is the normal vector oriented in the same direction as  $\delta \mathbf{x}(k)$ . Now, we can again invoke the symmetry assumption to write

$$\langle \ln \|\mathbb{D}^{(k)} \mathbb{W}^{(k)} \hat{\mathbf{n}}(k)\| \rangle = \langle \ln \|\mathbb{D}^{(k)} \mathbb{W}^{(k)} \mathbb{Q} \hat{\mathbf{n}}(k)\| \rangle. \quad (6.6)$$

Now, since  $\mathbb{Q}$  is arbitrary, we may choose  $\mathbb{Q}$  so that  $\mathbb{Q} \hat{\mathbf{n}} = \hat{\mathbf{e}}_1$ , where  $\hat{\mathbf{e}}_1$  selects the first column of  $\mathbb{D}^{(k)} \mathbb{W}^{(k)}$ . Hence, we find that the maximal Lyapunov

exponent can be evaluated as

$$\lambda_1 = \frac{1}{2} \left\langle \log \left[ \sum_{i=1}^N \left( \sum_{j=1}^N D_{ij} W_{j1} \right)^2 \right] \right\rangle. \quad (6.7)$$

This expression is evaluated in the Appendix, yielding

$$\lambda_1(\sigma_W^2, p, N) = \frac{1}{2} \sum_{n=1}^N \binom{N}{n} p^n (1-p)^{N-n} (\ln 2 + \ln \sigma_W^2 + \psi(n/2)) \quad (6.8)$$

where  $\psi(\cdot)$  is the digamma function. This equation is derived under the assumption that none of the  $\mathbb{D}^{(\ell)}$  are a zero matrix. The probability of this happening was derived by A. Dubey [103] and becomes negligible for large enough  $N$ . From Fig. 6.3 we can see a good agreement between the theory and simulations, and that the standard deviation of the FTLE distribution around the mean shrinks as  $N$  increases. This expression is similar to that of Newman [102]: it is the expected value of the Lyapunov exponent of a product of Gaussian random matrices where the dimension is sampled from a binomial distribution.

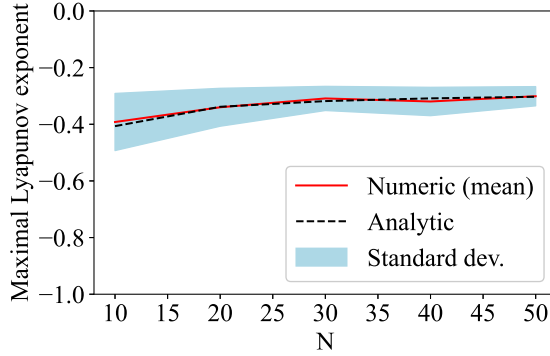
Let us consider the large- $N$  limit. Using the asymptotic expansion of the digamma function for large arguments, we have  $\psi(n) = \ln n + \mathcal{O}(n^{-1})$ . Hence, Eq. 6.8 becomes

$$\lim_{N \rightarrow \infty} \lambda_1(\sigma_W^2, p, N) = \frac{1}{2} \left[ \ln \sigma_W^2 + \langle \ln n \rangle_{n \sim \text{Binom}(p)} \right]. \quad (6.9)$$

Expanding the logarithm around the mean of the distribution  $Np$ , neglecting terms of order  $N^{-1}$  or smaller, we arrive at

$$\lambda_1(\sigma_W^2, p, N) = \frac{1}{2} \ln(pN\sigma_W^2) \quad (6.10)$$

which coincides with the mean-field result [62]. Due to the unstable gradient problem (Section 4), initialisation should be done so that the maximal Lyapunov exponent is zero. From the mean-field expression we see that we should initialise the weights to have a variance  $\sigma_W^2 = (pN)^{-1}$ . Initialising a network with a variance  $\sigma_W^2 = N^{-1}$  is popular and is called Xavier initialisation [68]. If the ReLU activation function is used,  $p \approx 0.5$  because the local



**Figure 6.3:** Lyapunov exponent calculated using Eq. 6.9 and the QR method for a feed-forward neural network employing the hard-tanh activation function and with a weight variance  $\sigma_W^2 = N^{-1}$  and threshold variance  $\sigma_b^2 = 1$ .

fields  $\mathbf{b}^{(\ell)}$  will be distributed symmetrically around zero, justifying the He initialisation  $\sigma_W^2 = 2N^{-1}$  [69].

## 7 Constraints on parameter choices for successful time-series prediction with echo-state networks

Echo-state networks (ESN), a type of recurrent neural network where only the output weights are trained, have been shown to perform on par with state-of-the-art time-series prediction algorithms [83]. This is despite the fact that the input and recurrent connections remain fixed from initialisation. In initialising ESNs, one must make choices on the design of the network such as its dimension  $N$  and the distribution from which the weights are sampled. One must also decide on parameters for the training of the output weights and the frequency at which the time series to be predicted will be sampled. At the time of writing Paper B, these choices were guided primarily by heuristics, except for the requirement that the initialisation must be such that the ESN displays the echo-state property. The most common heuristics employed, introduced by Jaeger [65] in his original paper on ESNs, are that

the network should be initialised such that the reservoir states display ‘rich’ dynamics and that the time scale of the network dynamics should match that of the predicted time series. By ‘rich’ is meant that each reservoir state displays a different mode of the time-series dynamics.

In Paper B, we aim to understand under what parameter choices time-series prediction is successful. Our first finding is that prediction performance is based on a combination of parameters that are usually tuned separately. Secondly, we find qualitatively different parameter dependencies based on whether full or partial information is provided to the network during training.

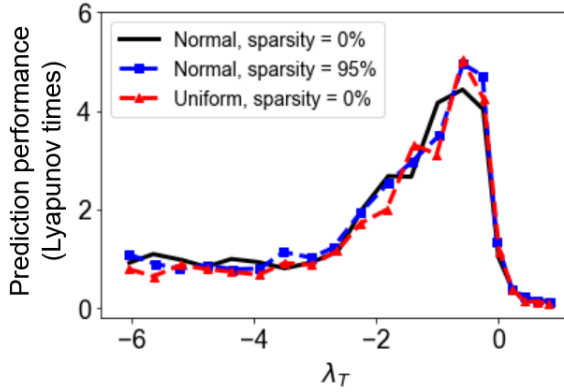
## 7.1 Lyapunov exponent of the reservoir dynamics

In Paper B, we find that the maximal Lyapunov exponent in the large- $N$  limit of the driven ESN dynamics is given by

$$\lambda_T = \frac{1}{2} \left[ \ln(sN\sigma_A^2) + \ln \left( N^{-1} \sum_{i=1}^N \langle D_{ii}^2(t) \rangle \right) \right]. \quad (7.1)$$

Here,  $N$  is the number of neurons in the reservoir,  $\sigma_A^2$  is the variance of the reservoir weights sampled from a random distribution with zero mean,  $s$  is the sparsity (a matrix  $\mathbb{A}$  with  $s \in [0, 1]$  has a fraction  $1 - s$  of the elements randomly set to zero), and  $D_{ii} = \frac{d}{db_i} g(b_i)$ . The average is taken over an ensemble of reservoir states, where it is assumed that the underlying dynamics is statistically stationary and ergodic. The parameters  $s$ ,  $N$ , and  $\sigma_A^2$  are usually tuned separately in literature [74, 104, 105], but as can be seen from Eq. 7.1, the combined parameter  $sN\sigma_A^2$  is what is important. In fact, what we find is that the combined parameter is what matters for performance. In Fig. 7.1, the prediction performance is measured in how many Lyapunov times the ESN is able to successfully predict a time series. The ESN is trained on the  $y$ -component of the Lorenz dynamics,

$$\begin{aligned} \frac{d}{dt} x &= \sigma(y - x), \\ \frac{d}{dt} y &= \rho x - y - xz \\ \frac{d}{dt} z &= xy - \beta z \end{aligned} \quad (7.2)$$

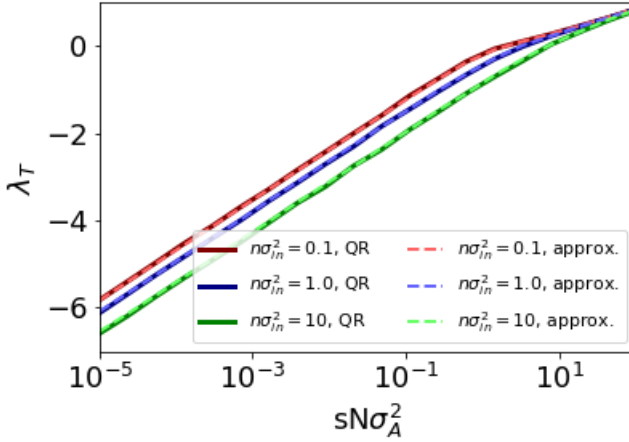


**Figure 7.1:** Prediction performance of an ESN with dimension  $N = 500$  and input variance  $\sigma_{\text{in}}^2 = 0.1$  trained on the  $y$ -component of the Lorenz dynamics. The time-series was sampled with a time step  $\delta t = 0.01$  and trained using ridge regression with a ridge parameter  $k = 0.001$ .

where setting  $\sigma = 10$ ,  $\rho = 28$ , and  $\beta = 8/3$  will yield chaotic dynamics [1]. Three cases are considered: (1)  $\mathbb{A}$  initialised densely with Gaussian weights, (2)  $\mathbb{A}$  initialised with Gaussian weights where 95% of the weights are randomly set to zero, and (3)  $\mathbb{A}$  initialised densely with uniform weights. As can be seen, all performance curves collapse, showing that what controls the performance is the Lyapunov exponent  $\lambda_T$  depending on the combined parameter. In Paper B, we also show that  $\langle D_{ii}^2(t) \rangle$  depends on the combined parameter and the variance of the input weights  $\mathbb{W}^{(\text{in})}$ .

### 7.1.1 Efficient computation of Lyapunov exponent

Equation 7.1 shows us that the important tuning parameter is  $sN\sigma_A^2$ , but it also provides us with a more efficient way of computing the Lyapunov exponent than using the QR method, which can be costly due to the large dimension of the reservoir. Since  $sN\sigma_A^2$  is known from initialisation, by storing the values of the diagonal matrices  $\mathbb{D}(t)$  during training, and computing the variance of the reservoir states over time, the Lyapunov exponent can be obtained efficiently. In Fig. 7.2 is shown the Lyapunov exponent obtained using the QR method and our mean-field approximation over a range of different parameter values, where a very good agreement is found.

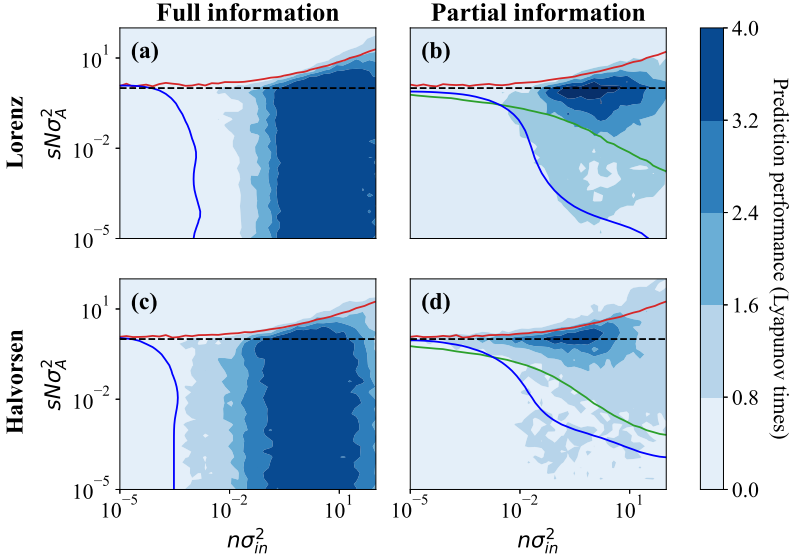


**Figure 7.2:** Lyapunov exponent computed using the QR method and the mean-field approximation in Eq. 7.1 for different input scales  $n\sigma_{\text{in}}^2$ .

## 7.2 Full and partial information

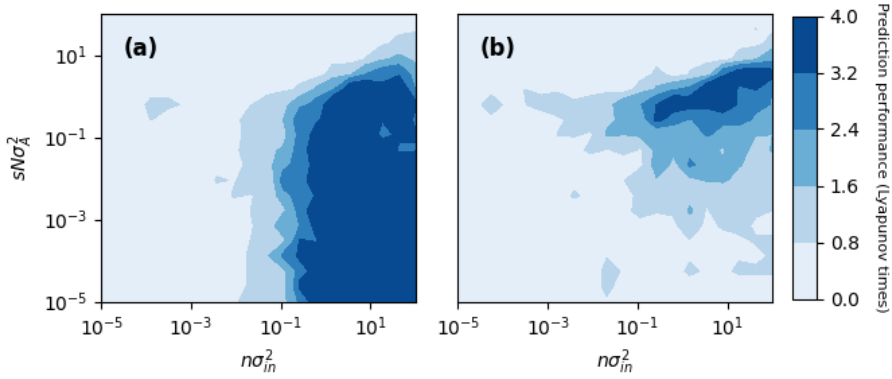
With the parameter search space constrained, we sweep  $sN\sigma_A^2$  and  $n\sigma_{\text{in}}^2$  to find for what values prediction is successful. This is done for two different chaotic time series: the Lorenz system [1] and the Halvorsen system [106], both being 3-dimensional and continuous. The result, taken from Paper B, is shown in Fig. 7.3. We performed the sweep for two cases: full and partial information. In the full-information case, all three phase space coordinates are shown to the network during training. This means that the network must essentially model a Markov process; all information for the next time step is contained in the current time step. In the partial-information case, the network only has access to a single coordinate in phase space. In this context, the network must perform time-delay embedding to reconstruct the dynamics, as was shown in [107]. Our contribution is to show that the region in parameter space where prediction is successful is qualitatively different for the two cases. The red line in Fig. 7.3 shows where the maximal Lyapunov exponent during training  $\lambda_T$  is zero. As expected, a necessary condition for successful prediction is that the exponent is negative. However, the result makes it clear that this is only a necessary but not sufficient condition.

The results in Paper B were obtained for continuous dynamical systems.



**Figure 7.3:** Prediction performance, measured as the average Lyapunov time, for an ESN with dimension  $N = 500$ , trained on the Lorenz and Halvorsen systems system with (a,c) full and (b,d) partial information. The red line indicates where the maximal Lyapunov exponent of the training dynamics  $\lambda_T = 0$ , the blue line indicates where the network dynamics bifurcate from fixed-point dynamics to oscillations, and the green line shows where  $\text{Rank}(\mathbb{R}\mathbb{R}^\top) = 100$  (details in text). The prediction performance was obtained by averaging over 50 independent trials. The figure was copied from Paper B with permission.





**Figure 7.4:** Prediction performance of a reservoir with dimension  $N = 500$  predicting the chaotic Ikeda map. The performance is averaged over 10 independent trials.

Here, we include an additional result of an ESN predicting the discrete Ikeda map [108]

$$\begin{aligned} x(t+1) &= 1 + u[x(t)\cos(\tau) - y(t)\sin(\tau)], \\ y(t+1) &= u[x(t)\sin(\tau) + y(t)\cos(\tau)], \end{aligned} \quad (7.3)$$

where  $u = 0.8$  and  $\tau = 0.4 - 6/(1 + x^2(t) + y^2(t))$ , in Fig. 7.4. For the chosen parameter value, the maximal Lyapunov exponent is  $\lambda_1 = 0.24^1$ .

### 7.2.1 Full information

In the full-information case, successful prediction is independent of  $sN\sigma_A^2$ , and the reservoir connections can in fact be set to zero, resulting in the ESN becoming a nonlinear mapping to an  $N$ -dimensional space. As was shown in [109], this is equivalent to nonlinear vector autoregression. The dependence on the input strength  $n\sigma_{\text{in}}^2$  is explained, at least for the lower bound, by the constraints set by the training method used to obtain the output weights  $\mathbb{W}^{(\text{out})}$ . As previously mentioned, once the network has been trained, the ESN dynamics are made autonomous by replacing the driving signal  $\mathbf{x}(t)$  by  $\mathbb{W}^{(\text{out})}\mathbf{r}(t)$ . The autonomous dynamics will have a different maximal

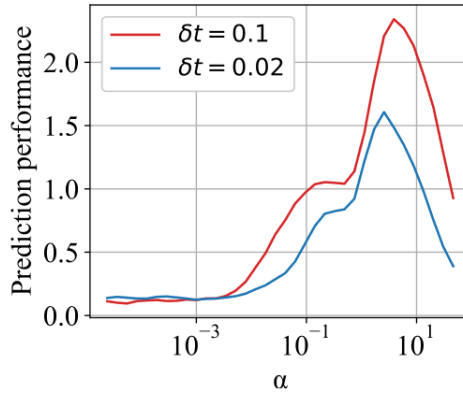
<sup>1</sup>This result was computed using the QR method.

Lyapunov exponent from the non-autonomous training dynamics. This is because the dynamics of a perturbation during training is governed by a product of matrices on the form  $\mathbb{D}(t)\mathbb{A}$ , whereas a perturbation of the autonomous dynamics is evolved through matrices on the form  $\mathbb{D}(t)[\mathbb{A} + \mathbb{W}^{(\text{in})}\mathbb{W}^{(\text{out})}]$ . The autonomous dynamics must have the same Lyapunov spectrum as the predicted dynamics to faithfully reproduce it. However, we show that when ridge regression is employed to obtain  $\mathbb{W}^{(\text{out})}$ , because the ridge parameter limits the magnitude of the elements of the matrix, the input strength  $n\sigma_{\text{in}}^2$  must be large enough in magnitude so that the autonomous dynamics has a positive maximal Lyapunov exponent. The blue line in Fig. 7.3 shows where the autonomous dynamics bifurcates from a fixed point to a periodic orbit which eventually evolves into chaotic dynamics as  $n\sigma_{\text{in}}^2$  increases. We show that the bifurcation occurs at larger values as the ridge parameter is increased. The upper bound for  $n\sigma_{\text{in}}^2$  appears because the tanh activation function saturates for large arguments. Hence, information about the time series is lost and prediction fails.

### 7.2.2 Partial information

In the case where only partial information is provided, a smaller region of parameter phase space results in successful prediction. As mentioned, the ESN performs time-delay embedding in this context [107]. In order for this to be possible, the ESN must be able to sample the dynamics at different time scales. This connects to the heuristic about the ‘richness’ of the dynamics: the dynamics of the different reservoir states  $r_i(t)$  must represent the different time scales. We quantify this by considering the correlation between the  $N$  different time series generated by reservoir states. Constructing an  $N \times T$  matrix  $\mathbb{R}$  whose rows represent the reservoir dynamics over  $T$  time steps during training, the rank of the matrix  $\mathbb{R}\mathbb{R}^\top$  becomes smaller the more correlated the time series are. The higher the rank, the more time scales the ESN has at its disposal to construct a time-delay embedding. In Fig. 7.3, the green line is drawn where the rank of  $\mathbb{R}\mathbb{R}^\top$  equals 100. Above the line, the rank increases gradually, improving the approximation.

Another common heuristic [65] is that the reservoir connection strength  $sN\sigma_A^2$  should be chosen so that the characteristic time scale of the ESN dynamics matches those of the predicted time series. This is not supported by our results. While the sampling time for time-delay embedding can be



**Figure 7.5:** Prediction performance, measured in Lyapunov times, of an ESN with weight matrix elements given by  $A_{ij} = \alpha \frac{i}{N} \delta_{ij}$ , predicting the Lorenz time series sampled at different rates  $\delta t$ . The result is obtained by averaging over 200 independent trials. Reproduced from Paper B with permission.

optimised (Paper B, Fig. 4), Takens' embedding theorem does not rely on the sampling time. Instead, it sets a lower bound for the required number of delays to be sampled. Hence, as long as the ESN can sample sufficiently many time scales from the dynamics, it does not matter which time scales are being sampled. We demonstrate that the time scale of the ESN dynamics and the predicted time series are unrelated by deterministically constructing a reservoir weight matrix  $\mathbb{A}$  whose elements are  $A_{ij} = \alpha \frac{i}{N} \delta_{ij}$  for some constant  $\alpha$  [110]; that is, where the reservoir states  $r_i(t)$  only connect to themselves with increasing strength as  $i$  increases. The time scales sampled by this reservoir are directly given by the diagonal elements of  $\mathbb{A}$ . Now, if the time scale of the ESN should be matched with the predicted time series, it would be necessary to change the weights of  $\mathbb{A}$  when changing the rate  $\delta t$  at which the input time series is sampled, as an iteration of the ESN dynamics would correspond to a different time scale. However, as seen in Fig. 7.5, the reservoir time scale at which prediction is successful remains the same despite changing the rate by an order of magnitude. The increase in prediction performance when increasing  $\delta t$  can be attributed to that  $\delta t = 0.1$  lies closer to the optimal sampling rate of the Lorenz time series given by considering the mutual information between time steps, as discussed in Section 2.4. For the Lorenz time series, the optimal sampling rate is roughly  $\delta t = 0.17$  [32].

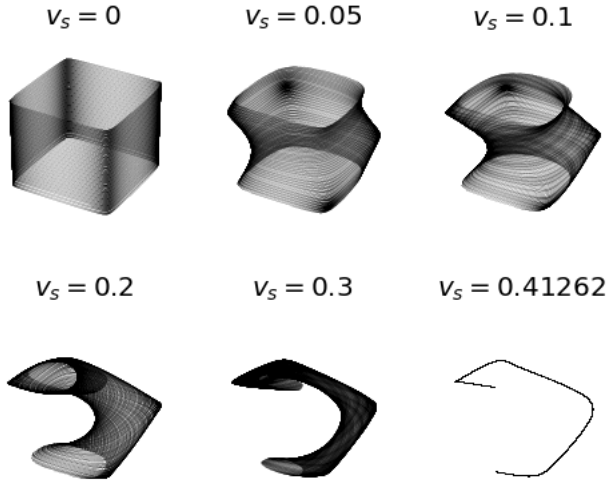
## 8 Transport barriers for microswimmers with orientational diffusion

We study transport barriers for microswimmers in the Taylor-Green vortex (TGV) flow as described in Section 5. Since the flow contains vortices, we look for elliptic LCS in the flow, which are closed transport barriers through which the swimmer cannot escape. Getting stuck in such a structure could prevent optimal foraging or migration of swimmers [111], and could increase the risk of being captured by predators [112, 113]. The structures may also affect navigation strategies between different locations of interest [18, 114]. Elliptic LCS are closed transport barriers that cannot be crossed by the deterministic trajectories of the dynamical system. However, the locomotion of microswimmers is known to induce diffusion in their orientation [87, 115]. This random motion may be exploited by the microswimmers to pass through the elliptic LCS. The aim of this project is to study how orientational diffusion can be used by microswimmers to escape through transport barriers. Why would these transport barriers be relevant if the dynamics are no longer deterministic? While the variational definition presented in Section 3 is the most commonly used, LCS have also been defined as regions in phase space where transport due to weak diffusion is extremised [37, 116]. While the equivalence between the two definitions has only been shown for 2-dimensional dynamical systems with isotropic diffusion, numerical results show that the barriers tend to overlap [37].

So far, only spherical particles ( $\Lambda = 0$ ) have been studied, in which case the swimming dynamics are conservative. The orientational diffusion is introduced in the model by adding a white-noise term to the orientational dynamics with variance  $2D_\theta t$ .

### 8.1 Deformation of transport barriers

As a first result, we find that elliptic LCS indeed exist for the dynamical system described in Eq. 5.2b. These structures are found using the method described in Section 3. Furthermore, we find that as the swimming speed increases, the elliptic LCS is deformed, eventually disappearing, as seen in Fig. 8.1. The swimming speed at which the structure disappears agrees with earlier



**Figure 8.1:** Elliptic LCS in the dynamics of a spherical microswimmer in the TGV flow for different swimming speeds. The vertical axis corresponds to the orientation of the swimmer, while the two horizontal axes correspond to the location of the swimmer.

results [117] where, for the same model, it was shown that the dynamical system bifurcates from having a trapping region to being completely chaotic at a certain swimming speed. Naturally, if the swimming speed  $v_s = 0$ , the swimmer cannot escape through the barrier using diffusion, since the spatial dynamics decouple from the orientational dynamics. We see, however, that as the swimming speed increases, the deformation creates regions in the elliptic LCS through which vertical transport is possible. Fig. 8.2 shows the locations where swimmers with weak orientational diffusion  $D_\theta = 0.001$  escape through the structure. The more tangential with the horizontal direction the normal of the LCS is, the less likely it is for the swimmer to escape through the structure. Hence, we expect that the deformation of the LCS will lead to that escaping using orientational diffusion is faster the larger the swimming speed.



**Figure 8.2:** Locations where swimmers with a swimming speed  $v_s = 0.1$  and orientational diffusion  $D_\theta = 0.001$  escape through the LCS.

---

## 9 Conclusions and Outlook

Closed-form solutions to complex, chaotic dynamical systems are in general difficult, and most often impossible, to find. In this work, tools borrowed from dynamical-systems theory – Lyapunov exponents and Lagrangian coherent structures – have been used to gain insight into three different kinds of complex systems; feed-forward neural networks, echo-state networks, and the dynamics of microswimmers. The tools allow us to discern structures in phase space that determine properties of the complex system. Below, I summarise the conclusions from the presented work together with some open questions and outlooks.

### 9.1 Finite-time Lyapunov exponents of deep neural networks

In the case of feed-forward neural networks, it was found using maximal finite-time Lyapunov exponents (FTLE) that the dynamics of a network trained on a classification task are particularly sensitive to changes in the input near boundaries between different classes. This results in the existence of ridges of large FTLEs at these boundaries. The direction in input space associated with the high sensitivity, the maximal finite-time Lyapunov vectors (FTLV), were found to be orthogonal to the ridges, making these structures akin to hyperbolic Lagrangian coherent structures. The ridges were found to be strongly correlated with the network's prediction uncertainty; a result leading us to conclude that the ridges are indeed associated with the decision boundaries of the neural network. We also found that, as the width of the network increases to be much larger than its depth, these ridges disappear and the network instead performs random embedding of the inputs in high-dimensional space. It is found that as the ridges disappear, the network no longer has to be trained (apart from the output weights), as the network is already able to make inputs from different classes be linearly separable from initialisation.

All in all, the FTLEs of feed-forward neural networks yield insight into how the network views the input space and how it learns to separate inputs from different classes. This is a step forward in the endeavour to understand the inner workings of neural networks. However, many questions remain to

be answered. For instance, we do not yet know whether this framework is applicable to other network architectures such as convolutional or recurrent neural networks. Additionally, the dependence on width has only been shown when the standard scaling of the local fields is used, where it is known that the network enters the lazy-learning regime for large widths. Will the ridges remain if the local fields are instead scaled in such a way that the network does not enter this regime, and in such a case, can the variance of the FTLE across input space be used as a metric for whether networks learn features or not? Another question is whether the disappearance of ridges in the large-width limit can be connected to the observation that wide networks are more difficult to perform adversarial attacks on than narrow ones: if the network dynamics is uniformly sensitive across input space, designing adversarial examples may be difficult.

## 9.2 Constraints on parameter choices for successful time-series prediction with echo-state networks

As for echo-state networks (ESN), our work reveals that the maximal Lyapunov exponent of the network dynamics during training is a key parameter determining the network performance for time-series prediction. The Lyapunov exponent is computed in the mean-field limit and is shown to depend on a combination of parameters that are usually tuned separately. These are the network dimension  $N$ , the sparsity of the connections between reservoir neurons  $s$ , and the variance of these connections  $\sigma_A^2$ . Our results reveal that it is the combined parameter  $sN\sigma_A^2$  that determine prediction performance. Hence, this minimises the search space when tuning network parameters, at least for networks large enough for the mean-field approximation to be accurate. We then explore the reduced parameter space and find that there is a qualitative difference in the parameter dependence when networks are provided with full or partial information about the input time series. The region where time-series prediction is successful is then explained borrowing insight from time-delay embedding and bifurcation analysis. It is, for instance, found that the degree with which the different reservoir states are correlated over time impacts the performance of the network. This is because the more correlated the reservoir states are, the fewer time scales the reservoir dynamics sample. The more time scales that are sampled, the more



accurate the model becomes.

The ESNs in Paper B were initialised randomly with a large number of computational nodes so that mean-field theory could be applied. Can the insight that prediction performance is improved when more time scales of the input are sampled be used to deterministically construct a reservoir with good prediction performance? Another question concerns the time scale of the ESN dynamics. Since the prediction performance does not seem to be related to tuning the the characteristic time scale of the ESN training dynamics (the Lyapunov time) to the input time series, is the best performance achieved by simply maximising the number of time scales sampled by the reservoir? In Paper B, an ESN was deterministically constructed as so that each neuron only connected with itself. The connection strength was different for each neuron to facilitate the sampling of different time scales. While this was only used to demonstrate that the ESN characteristic time scale is independent of the characteristic time scale of the predicted time series, the ability to directly tune the time scale of the ESN may be used to further investigate the relation between prediction performance and the dynamic memory of the reservoir. Other simple, deterministic topologies may also be investigated, as how the topology of the reservoir affects prediction performance is still largely an open question [118, 119]. Finally, it would be interesting to see whether the conclusions found in Paper B transfer to physically realised reservoir computers.

### 9.3 Transport barriers for microswimmers with orientational diffusion

In the on-going project on microswimmers, our results reveal that elliptic LCS exist in the swimming dynamics of a microswimmer in the TGV flow. Furthermore, the results show that the LCS deforms as the swimming speed increases. The deformation causes the structure to have regions where escape using orientational diffusion is possible.

In the future, we intend to find the average flux through the LCS as a function of swimming speed and diffusion strength. Additionally, we wish to study how the LCS is affected when the shape of the swimmer is no longer a sphere, so that the dynamics are no longer conservative. Having quantified the rate of escape for swimmers with orientational diffusion, we will compare

this to the rate of escape for swimmers that can preferentially sample in which direction to reorient, given some flow signal such as local strain. Training such a swimmer to maximise escape rate using reinforcement learning, we will study how the swimmer exploits its orientational dynamics to escape the LCS. This will be done for different swimming speeds and shape factors. Do different strategies emerge when the swimmer is non-spherical, and does the deformation of the LCS allow for different escape strategies to be used? Other optimisation goals may also be considered, such as how the energy expended by the swimmer to escape the structure can be minimised. Another interesting question is how the escape strategy changes when the swimmer performs run-and-tumble [120, 121] instead of Brownian motion. A continuation of this work is to consider time-varying flows such as isotropic turbulence instead of the TGV flow, to see whether similar conclusions can be drawn from the vortex structures found in such flows.

Understanding how orientational diffusion can be used to escape LCS may help in understanding the swimming strategies of real-world microswimmers such as plankton and bacteria, and may help in the design of optimal-navigation strategies of artificial microswimmers.

## Bibliography

- [1] LORENZ, E. N & HAMAN, K 1996 The essence of chaos. *Pure and Applied Geophysics* **147** (3), 598–599.
- [2] HOLMES, P 1990 Poincaré, celestial mechanics, dynamical-systems theory and “chaos”. *Physics Reports* **193** (3), 137–163.
- [3] BEC, J, GUSTAVSSON, K & MEHLIG, B 2024 Statistical models for the dynamics of heavy particles in turbulence. *Annual Review of Fluid Mechanics* **56**, 189–213.
- [4] GUSTAVSSON, K & MEHLIG, B 2016 Statistical models for spatial patterns of heavy particles in turbulence. *Advances in Physics* **65** (1), 1–57.
- [5] SOMMERER, J. C & OTT, E 1993 Particles floating on a moving fluid: A dynamically comprehensible physical fractal. *Science* **259** (5093), 335–339.
- [6] KITZBICHLER, M. G, SMITH, M. L, CHRISTENSEN, S. R & BULLMORE, E 2009 Broadband criticality of human brain network synchronization. *PLoS computational biology* **5** (3), e1000314.
- [7] ROGERS, T. L, JOHNSON, B. J & MUNCH, S. B 2022 Chaos is not rare in natural ecosystems. *Nature Ecology & Evolution* **6** (8), 1105–1111.
- [8] HUDSON, J & MANKIN, J 1981 Chaos in the belousov–zhabotinskii reaction. *The Journal of Chemical Physics* **74** (11), 6171–6177.
- [9] POOLE, B, LAHIRI, S, RAGHU, M, SOHL-DICKSTEIN, J & GANGULI, S 2016 Exponential expressivity in deep neural networks through transient chaos. *Advances in Neural Information Processing Systems* pp. 3368–3376.
- [10] OTT, E 2002 *Chaos in dynamical systems*. Cambridge university press.
- [11] CVITANOVIC, P, ARTUSO, R, MAINIERI, R, TANNER, G, VATTAY, G, WHELAN, N & WIRZBA, A 2005 Chaos: classical and quantum. *ChaosBook.org (Niels Bohr Institute, Copenhagen 2005)* **69**, 25.
- [12] Lyapunov exponents: A tool to explore complex dynamics.

- [13] CRISANTI, A, PALADIN, G & VULPIANI, A 2012 *Products of random matrices: in Statistical Physics*, , vol. 104. Springer Science & Business Media.
- [14] NI, R, OUELLETTE, N. T & VOTH, G. A 2014 Alignment of vorticity and rods with lagrangian fluid stretching in turbulence. *Journal of Fluid Mechanics* **743**, R3.
- [15] BEZUGLYY, V, MEHLIG, B & WILKINSON, M 2010 Poincaré indices of rheoscopic visualisations. *Europhysics Letters* **89** (3), 34003.
- [16] WILSON, M. M, PENG, J, DABIRI, J. O & ELDREDGE, J. D 2009 Lagrangian coherent structures in low reynolds number swimming. *Journal of Physics: Condensed Matter* **21** (20), 204105.
- [17] HALLER, G 2011 A variational theory of hyperbolic lagrangian coherent structures. *Physica D: Nonlinear Phenomena* **240** (7), 574–598.
- [18] KRISHNA, K, BRUNTON, S & SONG, Z 2021 Ftle of optimally controlled agents in unsteady flow fields. In *APS Division of Fluid Dynamics Meeting Abstracts*, pp. T07–008.
- [19] BERON-VERA, F, BROWN, M. G, OLASCOAGA, M, RYPINA, I. I, KOÇAK, H & UDOVYDCHENKOV, I. A 2008 Zonal jets as transport barriers in planetary atmospheres. *Journal of the atmospheric sciences* **65** (10), 3316–3326.
- [20] BERON-VERA, F & OLASCOAGA, M 2009 An assessment of the importance of chaotic stirring and turbulent mixing on the west florida shelf. *Journal of physical oceanography* **39** (7), 1743–1755.
- [21] LEKIEN, F, COULLIETTE, C, MARIANO, A. J, RYAN, E. H, SHAY, L. K, HALLER, G & MARSDEN, J 2005 Pollution release tied to invariant manifolds: A case study for the coast of florida. *Physica D: Nonlinear Phenomena* **210** (1-2), 1–20.
- [22] SHADDEN, S. C 2011 Lagrangian coherent structures. *Transport and mixing in laminar flows: from microfluidics to oceanic currents* pp. 59–89.
- [23] HALLER, G 2015 Lagrangian coherent structures. *Annu. Rev. Fluid Mech* **47** (1), 137–162.

- [24] KRISHNA, K, BRUNTON, S. L & SONG, Z 2023 Finite time lyapunov exponent analysis of model predictive control and reinforcement learning. *arXiv preprint arXiv:2304.03326*.
- [25] GURTIN, M. E, FRIED, E & ANAND, L 2010 *The mechanics and thermodynamics of continua*. Cambridge university press.
- [26] REN, J, CHEN, C, LIU, Z, LI, R & WANG, G 2012 Plastic dynamics transition between chaotic and self-organized critical states in a glassy metal via a multifractal intermediate. *Physical Review B* **86** (13), 134303.
- [27] JIN, X, CHEN, Y, WANG, L, HAN, H & CHEN, P 2021 Failure prediction, monitoring and diagnosis methods for slewing bearings of large-scale wind turbine: A review. *Measurement* **172**, 108855.
- [28] CUI, Z, DUBEY, A, ZHAO, L & MEHLIG, B 2020 Alignment statistics of rods with the lagrangian stretching direction in a channel flow. *Journal of Fluid Mechanics* **901**, A16.
- [29] WOLF, A, SWIFT, J. B, SWINNEY, H. L & VASTANO, J. A 1985 Determining lyapunov exponents from a time series. *Physica D: nonlinear phenomena* **16** (3), 285–317.
- [30] SHIMADA, I & NAGASHIMA, T 1979 A numerical approach to ergodic problem of dissipative dynamical systems. *Progress of theoretical physics* **61** (6), 1605–1616.
- [31] OKUSHIMA, T 2003 New method for computing finite-time lyapunov exponents. *Physical review letters* **91** (25), 254101.
- [32] KANTZ, H & SCHREIBER, T 2004 *Nonlinear time series analysis*, , vol. 7. Cambridge university press.
- [33] ASH, R. B 2012 *Information theory*. Courier Corporation.
- [34] AREF, H 1983 Integrable, chaotic, and turbulent vortex motion in two-dimensional flows. *Annual Review of Fluid Mechanics* **15** (1), 345–389.
- [35] PROVENZALE, A 1999 Transport by coherent barotropic vortices. *Annual review of fluid mechanics* **31** (1), 55–93.

- [36] ELHMAÏDI, D, PROVENZALE, A & BABIANO, A 1993 Elementary topology of two-dimensional turbulence from a lagrangian viewpoint and single-particle dispersion. *Journal of Fluid Mechanics* **257**, 533–558.
- [37] KARRASCH, D & KELLER, J 2020 A geometric heat-flow theory of lagrangian coherent structures. *Journal of Nonlinear Science* **30** (4), 1849–1888.
- [38] PIERREHUMBERT, R 1991 Large-scale horizontal mixing in planetary atmospheres. *Physics of Fluids A: Fluid Dynamics* **3** (5), 1250–1260.
- [39] PIERREHUMBERT, R. T & YANG, H 1993 Global chaotic mixing on isentropic surfaces. *Journal of the atmospheric sciences* **50** (15), 2462–2480.
- [40] SCHRÖDER, A & SCHANZ, D 2023 3d lagrangian particle tracking in fluid mechanics. *Annual Review of Fluid Mechanics* **55**, 511–540.
- [41] DUNKERTON, T. J, MONTGOMERY, M & WANG, Z 2009 Tropical cyclogenesis in a tropical wave critical layer: Easterly waves. *Atmospheric Chemistry and Physics* **9** (15), 5587–5646.
- [42] VAN SEBILLE, E, ALIANI, S, LAW, K. L, MAXIMENKO, N, ALSINA, J. M, BAGAEV, A, BERGMANN, M, CHAPRON, B, CHUBARENKO, I, CÓZAR, A *et al.* 2020 The physical oceanography of the transport of floating marine debris. *Environmental Research Letters* **15** (2), 023003.
- [43] KASHINATH, K, MUSTAFA, M, ALBERT, A, WU, J, JIANG, C, ESMAEILZADEH, S, AZIZZADENESHELI, K, WANG, R, CHATTOPADHYAY, A, SINGH, A *et al.* 2021 Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A* **379** (2194), 20200093.
- [44] ALLSHOUSE, M. R & PEACOCK, T 2015 Lagrangian based methods for coherent structure detection. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **25** (9).
- [45] SUDHARSAN, M, BRUNTON, S. L & RILEY, J. J 2016 Lagrangian coherent structures and inertial particle dynamics. *Physical Review E* **93** (3), 033108.

- [46] NOLAN, P. J, FOROUTAN, H & ROSS, S. D 2020 Pollution transport patterns obtained through generalized lagrangian coherent structures. *Atmosphere* **11** (2), 168.
- [47] BLAZEWSKI, D & HALLER, G 2014 Hyperbolic and elliptic transport barriers in three-dimensional unsteady flows. *Physica D: Nonlinear Phenomena* **273**, 46–62.
- [48] HIJAZI, S, KUMAR, R, ROWEN, C *et al.* 2015 Using convolutional neural networks for image recognition. *Cadence Design Systems Inc.: San Jose, CA, USA* **9** (1).
- [49] ZHANG, N, CAI, Y.-X, WANG, Y.-Y, TIAN, Y.-T, WANG, X.-L & BADAMI, B 2020 Skin cancer diagnosis based on optimized convolutional neural network. *Artificial intelligence in medicine* **102**, 101756.
- [50] CRESWELL, A, WHITE, T, DUMOULIN, V, ARULKUMARAN, K, SENGUPTA, B & BHARATH, A. A 2018 Generative adversarial networks: An overview. *IEEE signal processing magazine* **35** (1), 53–65.
- [51] VASWANI, A, SHAZEER, N, PARMAR, N, USZKOREIT, J, JONES, L, GOMEZ, A. N, KAISER, Ł & POLOSUKHIN, I 2017 Attention is all you need. *Advances in neural information processing systems* **30**.
- [52] SALEHINEJAD, H, SANKAR, S, BARFETT, J, COLAK, E & VALAEE, S 2017 Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*.
- [53] HORNIK, K, STINCHCOMBE, M & WHITE, H 1989 Multilayer feedforward networks are universal approximators. *Neural networks* **2** (5), 359–366.
- [54] OLDEN, J. D & JACKSON, D. A 2002 Illuminating the “black box”: a randomization approach for understanding variable contributions in artificial neural networks. *Ecological modelling* **154** (1-2), 135–150.
- [55] SHWARTZ-ZIV, R & TISHBY, N 2017 Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.
- [56] TZENG, F.-Y & MA, K.-L 2005 *Opening the black box-data driven visualization of neural networks*. IEEE.

- [57] RUDIN, C 2019 Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence* **1** (5), 206–215.
- [58] VARSHNEY, K. R & ALEMZADEH, H 2017 On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. *Big data* **5** (3), 246–255.
- [59] ZABLOCKI, É, BEN-YOUNES, H, PÉREZ, P & CORD, M 2022 Explainability of deep vision-based autonomous driving systems: Review and challenges. *International Journal of Computer Vision* **130** (10), 2425–2452.
- [60] PENNINGTON, J, SCHOENHOLZ, S. S & GANGULI, S 2017 Resurrecting the sigmoid in deep learning through dynamical isometry: Theory and practice. *Advances in Neural Information Processing Systems 2017-Decem* (Nips), 4786–4796.
- [61] SCHOENHOLZ, S. S, GILMER, J, GANGULI, S & SOHL-DICKSTEIN, J 2016 Deep information propagation. *arXiv preprint arXiv:1611.01232* .
- [62] MEHLIG, B 2021 *Machine learning with neural networks: an introduction for scientists and engineers*. Cambridge University Press.
- [63] RAMSAUER, H, SCHÄFL, B, LEHNER, J, SEIDL, P, WIDRICH, M, ADLER, T, GRUBER, L, HOLZLEITNER, M, PAVLOVIĆ, M, SANDVE, G. K *et al.* 2020 Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217* .
- [64] LI, P, PEI, Y & LI, J 2023 A comprehensive survey on design and application of autoencoder in deep learning. *Applied Soft Computing* p. 110176.
- [65] JAEGER, H 2001 The “echo state” approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* **148** (34), 13.
- [66] JANOCHA, K & CZARNECKI, W. M 2017 On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659* .
- [67] KINGMA, D. P & BA, J 2014 Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .



- [68] GLOT, X & BENGIO, Y 2010 Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings.
- [69] HE, K, ZHANG, X, REN, S & SUN, J 2015 Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.
- [70] YANG, G & HU, E. J 2020 Feature learning in infinite-width neural networks. *arXiv preprint arXiv:2011.14522*.
- [71] JACOT, A, GABRIEL, F & HONGLER, C 2018 Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems* **31**.
- [72] CHIZAT, L, OYALLON, E & BACH, F 2019 On lazy training in differentiable programming. *Advances in neural information processing systems* **32**.
- [73] CHIZAT, L & BACH, F 2018 On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in neural information processing systems* **31**.
- [74] OZTURK, M. C, XU, D & PRINCIPE, J. C 2007 Analysis and design of echo state networks. *Neural computation* **19** (1), 111–138.
- [75] SUN, C, SONG, M, HONG, S & LI, H 2020 A review of designs and applications of echo state networks. *arXiv preprint arXiv:2012.02974*.
- [76] LUKOŠEVIČIUS, M & JAEGER, H 2009 Reservoir computing approaches to recurrent neural network training. *Computer Science Review* **3** (3), 127–149.
- [77] TANAKA, G, YAMANE, T, HÉROUX, J. B, NAKANE, R, KANAZAWA, N, TAKEDA, S, NUMATA, H, NAKANO, D & HIROSE, A 2019 Recent advances in physical reservoir computing: A review. *Neural Networks* **115**, 100–123.
- [78] FERNANDO, C & SOJAKKA, S 2003 Pattern recognition in a bucket. In *European conference on artificial life*, pp. 588–597. Springer.

- [79] LU, Z, PATHAK, J, HUNT, B, GIRVAN, M, BROCKETT, R & OTT, E 2017 Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **27** (4), 041102.
- [80] TIKHONOV, A. N, ARSENIN, V. I, ARSENIN, V *et al.* 1977 *Solutions of ill-posed problems*. Vh Winston.
- [81] VERSTRAETEN, D, SCHRAUWEN, B, D'HAENE, M & STROOBANDT, D 2007 An experimental unification of reservoir computing methods. *Neural networks* **20** (3), 391–403.
- [82] GRIFFITH, A, POMERANCE, A & GAUTHIER, D. J 2019 Forecasting chaotic systems with very low connectivity reservoir computers. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **29** (12), 123108.
- [83] PATHAK, J, LU, Z, HUNT, B. R, GIRVAN, M & OTT, E 2017 Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **27** (12), 121102.
- [84] QIU, J, MOUSAVI, N, ZHAO, L & GUSTAVSSON, K 2022 Active gyrotactic stability of microswimmers using hydromechanical signals. *Physical Review Fluids* **7** (1), 014311.
- [85] ELGETI, J, WINKLER, R. G & GOMPPER, G 2015 Physics of microswimmers—single particle motion and collective behavior: a review. *Reports on progress in physics* **78** (5), 056601.
- [86] HU, J, ZHOU, S, SUN, Y, FANG, X & WU, L 2012 Fabrication, properties and applications of janus particles. *Chemical Society Reviews* **41** (11), 4356–4378.
- [87] RAO, K. J, LI, F, MENG, L, ZHENG, H, CAI, F & WANG, W 2015 A force to be reckoned with: a review of synthetic microswimmers powered by ultrasound. *Small* **11** (24), 2836–2846.
- [88] QIU, J, MOUSAVI, N, GUSTAVSSON, K, XU, C, MEHLIG, B & ZHAO, L 2022 Navigation of micro-swimmers in steady flow: The importance of symmetries. *Journal of Fluid Mechanics* **932**, A10.

- [89] BORRA, F, BIFERALE, L, CENCINI, M & CELANI, A 2022 Reinforcement learning for pursuit and evasion of microswimmers at low reynolds number. *Physical Review Fluids* **7** (2), 023103.
- [90] BERMAN, S. A, BUGGELN, J, BRANTLEY, D. A, MITCHELL, K. A & SOLOMON, T. H 2021 Transport barriers to self-propelled particles in fluid flows. *Physical Review Fluids* **6** (1), L012501.
- [91] GUSTAVSSON, K, BERGLUND, E, JONSSON, P & MEHLIG, B 2016 Preferential sampling and small-scale clustering of gyrotactic microswimmers in turbulence. *Physical review letters* **116** (10), 108104.
- [92] DURHAM, W. M, CLIMENT, E, BARRY, M, DE LILLO, F, BOFFETTA, G, CENCINI, M & STOCKER, R 2013 Turbulence drives microscale patches of motile phytoplankton. *Nature communications* **4** (1), 2148.
- [93] BORGNINO, M, GUSTAVSSON, K, DE LILLO, F, BOFFETTA, G, CENCINI, M & MEHLIG, B 2019 Alignment of nonspherical active particles in chaotic flows. *Physical review letters* **123** (13), 138003.
- [94] JEFFERY, G. B 1922 The motion of ellipsoidal particles immersed in a viscous fluid. *Proceedings of the Royal Society of London. Series A, Containing papers of a mathematical and physical character* **102** (715), 161–179.
- [95] PEDLEY, T. J & KESSLER, J 1987 The orientation of spheroidal microorganisms swimming in a flow field. *Proceedings of the Royal Society of London. Series B. Biological Sciences* **231** (1262), 47–70.
- [96] TAYLOR, G. I & GREEN, A. E 1937 Mechanism of the production of small eddies from large ones. *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences* **158** (895), 499–521.
- [97] DURHAM, W. M, CLIMENT, E & STOCKER, R 2011 Gyrotaxis in a steady vortical flow. *Physical Review Letters* **106** (23), 238102.
- [98] COLABRESE, S, GUSTAVSSON, K, CELANI, A & BIFERALE, L 2017 Flow navigation by smart microswimmers via reinforcement learning. *Physical review letters* **118** (15), 158004.

- [99] RICHARDSON, S. H, BAGGALEY, A & HILL, N 2018 Gyrotactic suppression and emergence of chaotic trajectories of swimming particles in three-dimensional flows. *Physical Review Fluids* **3** (2), 023102.
- [100] DENG, L 2012 The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine* **29** (6), 141–142.
- [101] COVER, T. M 1965 Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers* (3), 326–334.
- [102] NEWMAN, C. M 1986 The distribution of Lyapunov exponents: Exact results for random matrices. *Communications In Mathematical Physics* **103** (1), 121–126.
- [103] DUBEY, A 2021 Personal communication .
- [104] PATHAK, J, HUNT, B, GIRVAN, M, LU, Z & OTT, E 2018 Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical review letters* **120** (2), 024102.
- [105] JAEGER, H 2002 Short term memory in echo state networks. gmd-report 152. In *GMD-German National Research Institute for Computer Science (2002)*, <http://www.faculty.jacobs-university.de/hjaeger/pubs/STMEchoStatesTechRep.pdf>. Citeseer.
- [106] SPROTT, J. C 2003 *Chaos and time-series analysis*. Oxford university press.
- [107] HART, A, HOOK, J & DAWES, J 2020 Embedding and approximation theorems for echo state networks. *Neural Networks* **128**, 234–247.
- [108] IKEDA, K 1979 Multiple-valued stationary state and its instability of the transmitted light by a ring cavity system. *Optics communications* **30** (2), 257–261.
- [109] PYLE, R, JOVANOVIC, N, SUBRAMANIAN, D, PALEM, K. V & PATEL, A. B 2021 Domain-driven models yield better predictions at lower cost than reservoir computers in lorenz systems. *Philosophical Transactions of the Royal Society A* **379** (2194), 20200246.

- [110] FETTE, G & EGGERT, J 2005 Short term memory and pattern matching with simple echo state networks. In *International Conference on Artificial Neural Networks*, pp. 13–18. Springer.
- [111] BON, C, DELLA PENNA, A, D’OVIDIO, F, YP ARNOULD, J, POUPART, T & BOST, C.-A 2015 Influence of oceanographic structures on foraging strategies: Macaroni penguins at crozet islands. *Movement ecology* **3**, 1–11.
- [112] TEW KAI, E, ROSSI, V, SUDRE, J, WEIMERSKIRCH, H, LOPEZ, C, HERNANDEZ-GARCIA, E, MARSAC, F & GARÇON, V 2009 Top marine predators track lagrangian coherent structures. *Proceedings of the National Academy of Sciences* **106** (20), 8245–8250.
- [113] PENG, J & DABIRI, J 2009 Transport of inertial particles by lagrangian coherent structures: application to predator–prey interaction in jellyfish feeding. *Journal of Fluid Mechanics* **623**, 75–84.
- [114] BIFERALE, L, BONACCORSO, F, BUZZICOTTI, M, CLARK DI LEONI, P & GUSTAVSSON, K 2019 Zermelo’s problem: optimal point-to-point navigation in 2d turbulent flows using reinforcement learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **29** (10).
- [115] BERG, H. C 2004 *Escherichia coli in motion. Biological, and Medical Physics Biomedical Engineering. Springer Verlag AIP: Press New York* .
- [116] HALLER, G, KARRASCH, D & KOGELBAUER, F 2018 Material barriers to diffusive and stochastic transport. *Proceedings of the National Academy of Sciences* **115** (37), 9074–9079.
- [117] TORNEY, C & NEUFELD, Z 2007 Transport and aggregation of self-propelled particles in fluid flows. *Physical review letters* **99** (7), 078101.
- [118] XUE, Y, ZHANG, Q & SLOWIK, A 2023 Automatic topology optimization of echo state network based on particle swarm optimization. *Engineering Applications of Artificial Intelligence* **117**, 105574.
- [119] KAWAI, Y, PARK, J & ASADA, M 2019 A small-world topology enhances the echo state property and signal propagation in reservoir computing. *Neural Networks* **112**, 15–23.

- [120] SOLON, A. P, CATES, M. E & TAILLEUR, J 2015 Active brownian particles and run-and-tumble particles: A comparative study. *The European Physical Journal Special Topics* **224** (7), 1231–1262.
- [121] BENNETT, R & BENNETT, R 2015 Physics of microorganism behaviour: Motility, synchronisation, run-and-tumble, phototaxis. PhD thesis, Oxford University, UK.

## PART III

### APPENDIX

#### A Maximal Lyapunov exponent of a randomly initialised neural network

Here, we show the calculation of the maximal Lyapunov exponent of a feed-forward neural network, starting from the expression in Eq. 6.7.

$$\lambda_1 = \frac{1}{2} \left\langle \log \left[ \sum_{i=1}^N \left( \sum_{j=1}^N D_{ij} W_{j1} \right)^2 \right] \right\rangle. \quad (\text{A.1})$$

As the  $D$  matrix is diagonal, we may simplify Eq. A.1 to obtain

$$\lambda_1 = \frac{1}{2} \left\langle \log \left[ \sum_{i=1}^N (D_{ii} W_{i1})^2 \right] \right\rangle. \quad (\text{A.2})$$

We wish to know the PDF of the random variable  $Z_N = \sum_{i=1}^N (D_{ii} W_{i1})^2$  and start by finding the PDF of  $X = D_{ii} W_{i1}$ :

$$\begin{aligned} F_X(x) &= \Pr[X \leq x] = \Pr[D_{ii} W_{i1} \leq x] = p \Pr[W_{i1} \leq x] + (1-p) \Pr[0 \leq x] \\ &= p \Phi(x) + (1-p) \theta(x) \end{aligned} \quad (\text{A.3})$$

where  $\Phi(x)$  is the Gaussian CDF and  $\theta(x)$  is the Heaviside function. The PDF of  $X$  is then obtained by taking the derivative

$$f_X(x) = \frac{d}{dx} F_X(x) = p \phi(x) + (1-p) \delta(x) = \frac{p}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} + (1-p) \delta(x), \quad (\text{A.4})$$

where  $\delta(x)$  is the delta function. Next, we find the PDF of  $Y = X^2$  through a change of variables, as  $g(x) = x^2$  is a monotonic function for  $x \geq 0$ :

$$f_Y(y) = \left| \frac{d}{dy} g^{-1}(y) \right| f_X(g^{-1}(y)) = \frac{1}{y^{1/2}} \left( \frac{p}{\sqrt{2\pi\sigma^2}} e^{-\frac{y}{2\sigma^2}} + (1-p) \delta(y^{1/2}) \right), \quad y \geq 0. \quad (\text{A.5})$$

The multiplication by 2 comes from the fact that the PDF of  $X$  is symmetric around zero and is not constrained to positive values. The first term in the sum in Eq. A.5 is a scaled  $\chi^2$ -distribution of order 1, and we rewrite the expression as

$$f_Y(y) = \frac{p}{\sigma} \frac{1}{2^{1/2}\Gamma(1/2)} y^{1/2-1} e^{-\frac{y}{2\sigma^2}} + (1-p)\delta(y^{1/2})y^{-1/2}, \quad y \geq 0. \quad (\text{A.6})$$

The latter term of the sum in Eq. A.6 can be rewritten using the following relation

$$\delta(g(y)) = \frac{1}{|g'(y)|} \delta(y)$$

which leads to a cancellation of  $y^{-1/2}$ . We now proceed to calculate the distribution of  $Z_N = \sum_{i=1}^N (D_{ii} W_{i1})^2$  as the  $N$  convolutions of Eq. A.6. Starting with the case where  $N = 2$ , we have

$$\begin{aligned} (f_Y * g_Y)(z) &= \int_0^z f_Y(y)g_Y(z-y)dy \\ &= \left(\frac{p}{\sigma} \frac{1}{2^{1/2}\Gamma(1/2)}\right)^2 e^{-\frac{z}{2\sigma^2}} \int_0^z y^{1/2-1}(z-y)^{1/2-1}dy \\ &\quad + (1-p)\left(\frac{p}{\sigma} \frac{1}{2^{1/2}\Gamma(1/2)}\right) \int_0^z y^{1/2-1} e^{-\frac{y}{2\sigma^2}} \delta(z-y)dy \\ &\quad + (1-p)\left(\frac{p}{\sigma} \frac{1}{2^{1/2}\Gamma(1/2)}\right) \int_0^z (z-y)^{1/2-1} e^{-\frac{z-y}{2\sigma^2}} \delta(y)dy \\ &\quad + (1-p)^2 \int_0^z \delta(y)\delta(z-y)dy, \end{aligned} \quad (\text{A.7})$$

which simplifies to

$$\begin{aligned} (f_Y * g_Y)(z) &= \int_0^z f_Y(y)g_Y(z-y)dy \\ &= \left(\frac{p}{\sigma} \frac{1}{2^{1/2}\Gamma(1/2)}\right)^2 e^{-\frac{z}{2\sigma^2}} \int_0^z y^{1/2-1}(z-y)^{1/2-1}dy \\ &\quad + 2(1-p)\left(\frac{p}{\sigma} \frac{1}{2^{1/2}\Gamma(1/2)}\right) z^{1/2-1} e^{-\frac{z}{2\sigma^2}} \\ &\quad + (1-p)^2 \delta(z), \end{aligned} \quad (\text{A.8})$$



Performing the change of variables in the integral  $t = y/z$  we have

$$\left(\frac{p}{\sigma} \frac{1}{2^{1/2}\Gamma(1/2)}\right)^2 e^{-\frac{z}{2\sigma^2}} \int_0^1 t^{1/2-1}(1-t)^{1/2-1} dt \quad (\text{A.9})$$

where the integral is equal to the Beta function  $B(1/2, 1/2) = \frac{\Gamma(1/2)\Gamma(1/2)}{\Gamma(1/2+1/2)}$  and hence

$$\left(\frac{p}{\sigma} \frac{1}{2^{1/2}\Gamma(1/2)}\right)^2 e^{-\frac{z}{2\sigma^2}} \frac{\Gamma(1/2)\Gamma(1/2)}{\Gamma(1/2+1/2)} = \left(\frac{p}{\sigma}\right)^2 \frac{1}{2\Gamma(1)} e^{-\frac{z}{2\sigma^2}}. \quad (\text{A.10})$$

Thus, the result becomes

$$f_{Z_2}(z) = \left(\frac{p}{\sigma}\right)^2 \frac{1}{2\Gamma(1)} e^{-\frac{z}{2\sigma^2}} + 2(1-p) \left(\frac{p}{\sigma} \frac{1}{2^{1/2}\Gamma(1/2)}\right) z^{1/2-1} e^{-\frac{z}{2\sigma^2}} + (1-p)^2 \delta(z). \quad (\text{A.11})$$

For any  $N$  we find

$$f_{Z_N}(z) = \sum_{n=1}^N \binom{N}{n} p^n (1-p)^{N-n} \frac{z^{n/2-1}}{\sigma^n 2^{n/2}\Gamma(n/2)} e^{-\frac{z}{2\sigma^2}} + (1-p)^N \delta(z). \quad (\text{A.12})$$

We now calculate the expression for the maximal Lyapunov exponent as

$$\begin{aligned} \lambda_1 &= \frac{1}{2} \int_0^\infty \log(z) f_{Z_N}(z) dz \\ &= \frac{1}{2} \sum_{n=1}^N \binom{N}{n} p^n (1-p)^{N-n} \int_0^\infty \frac{\log(z) z^{n/2-1}}{\sigma^n 2^{n/2}\Gamma(n/2)} e^{-\frac{z}{2\sigma^2}} dz \\ &\quad + \frac{1}{2} (1-p)^N \int_0^\infty \log(z) \delta(z) dz \end{aligned} \quad (\text{A.13})$$

The first integral is calculated in e.g. [102], yielding an expression containing the digamma function  $\psi(x)$ . The second integral equals negative infinity. The probability of the monodromy matrix  $\mathbb{M}(0, L)$  to have zero rank was calculated by A. Dubey [103] to be

$$\Pr[\text{rank}(\mathbb{M}) = 0] = 1 - \left(1 - (1-p)^N\right)^L. \quad (\text{A.14})$$

Since our derivation is based on the expected logarithm of the magnitude of a single matrix  $\mathbb{D}^{(k)}\mathbb{W}^{(k)}$  (Eq. 6.5), the pre-factor in Eq. A.13 corresponds to the case where  $L = 1$  in Eq. A.14. Conditional on that the rank of  $\mathbb{D}^{(k)}$  is non-zero, we ignore this term and obtain the final result

$$\tilde{\lambda}_1 = \frac{1}{2} \sum_{n=1}^N \binom{N}{n} p^n (1-p)^{N-n} (\log 2 + \log \sigma^2 + \psi(n/2)). \quad (\text{A.15})$$

PART IV  
RESEARCH PAPERS



# Paper A

<https://journals.aps.org/prl/pdf/10.1103/PhysRevLett.132.057301>



## Finite-Time Lyapunov Exponents of Deep Neural Networks


L. Storm,<sup>1</sup> H. Linander,<sup>2</sup> J. Bec,<sup>3,4</sup> K. Gustavsson,<sup>1</sup> and B. Mehlig<sup>1</sup>

<sup>1</sup>*Department of Physics, University of Gothenburg, 41296 Gothenburg, Sweden*

<sup>2</sup>*Department of Mathematical Sciences, Chalmers Technical University and University of Gothenburg, Gothenburg, Sweden*

<sup>3</sup>*MINES Paris, PSL Research University, CNRS, Cemef, Valbonne, F-06900, France*

<sup>4</sup>*Université Côte d'Azur, Inria, CNRS, Cemef, Valbonne, F-06900, France*

 (Received 16 June 2023; revised 5 November 2023; accepted 3 January 2024; published 1 February 2024)

We compute how small input perturbations affect the output of deep neural networks, exploring an analogy between deep feed-forward networks and dynamical systems, where the growth or decay of local perturbations is characterized by finite-time Lyapunov exponents. We show that the maximal exponent forms geometrical structures in input space, akin to coherent structures in dynamical systems. Ridges of large positive exponents divide input space into different regions that the network associates with different classes. These ridges visualize the geometry that deep networks construct in input space, shedding light on the fundamental mechanisms underlying their learning capabilities.

DOI: 10.1103/PhysRevLett.132.057301

Deep neural networks can be trained to model complex functional relationships. The expressivity of such neural networks—their ability to unfold intricate data structures—increases exponentially as the number of layers increases [1]. Recent breakthrough applications of neural networks [2] use deep feed-forward layouts with many layers of neurons [3]. These are hard to train due to the multiplicative amplification of signals propagating through the layers, causing signals to either explode or vanish in magnitude if the number of layers is too large. Mathematical analysis in the asymptotic limit of infinitely wide layers reveals how deep networks can nevertheless learn to solve classification tasks [4–7]. Recent results indicate that finite-width networks may learn in different ways [8–10]. It is not understood, however, when and how such networks use their exponential expressivity to represent data features needed for a classification task, how the representation affects prediction accuracy and uncertainty, and how it depends on the network layout.

Here we use dynamical-systems theory to answer these questions. Deep feed-forward networks [Fig. 1(a)] are discrete dynamical systems. Inputs  $\mathbf{x}^{(0)}$  are mapped iteratively through  $x_i^{(\ell)} = g(\sum_{j=1}^{N_{\ell}} w_{ij}^{(\ell)} x_j^{(\ell-1)} - \theta_i^{(\ell)})$ , where  $g(\cdot)$  is a nonlinear activation function [11], the layer index  $\ell = 0, \dots, L + 1$  plays the role of time,  $L$  is the number of hidden layers,  $N_{\ell}$  is the number of neurons in layer  $\ell$ , and the weights  $w_{ij}^{(\ell)}$  and thresholds  $\theta_i^{(\ell)}$  are parameters.

Sensitivity of  $\mathbf{x}^{(\ell)}$  to small changes in the inputs  $\mathbf{x}^{(0)} = \mathbf{x}$  corresponds to exponentially growing perturbations in a chaotic system with positive maximal Lyapunov exponent [12,13]  $\lim_{\ell \rightarrow \infty} \lambda_1^{(\ell)}(\mathbf{x})$ , with growth rate  $\lambda_1^{(\ell)}(\mathbf{x}) = \ell^{-1} \log |\delta \mathbf{x}^{(\ell)}| / |\delta \mathbf{x}|$ . The latter is called maximal finite-time Lyapunov exponent (FTLE).

The network weights  $w_{ij}^{(\ell)}$  are usually initialized as random numbers, independently Gaussian distributed with zero mean and variance  $\sigma_{\ell}^2$ . In this case, the Lyapunov exponents are initially determined by a product of random matrices, and the multiplicative ergodic theorem guarantees that  $\lambda_1^{(L)}(\mathbf{x})$  converges as  $L \rightarrow \infty$ , to a limit that is independent of  $\mathbf{x}$  [14]. In the limit  $N_{\ell} = N \rightarrow \infty$ , one finds  $\lambda_1^{(L)} \sim \log(GN\sigma^2)$ , explaining why the initial weight variance should be chosen so that  $GN\sigma^2 = 1$ , because then signals neither contract nor expand [15–17], stabilizing initial stages of the learning (the constant  $G$  depends on the choice of activation function [15]).

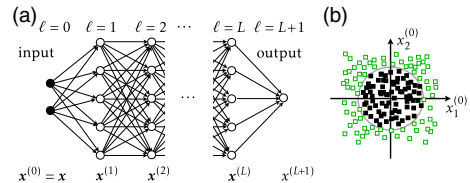


FIG. 1. Classification with a fully connected feed-forward network. (a) Layout with two input components  $x_1^{(0)}$  and  $x_2^{(0)}$ ,  $L$  hidden layers with five neurons each, and one output  $x^{(L+1)}$ . (b) Input plane (schematic) for a classification problem with a circular decision boundary that separates input patterns with targets  $t = +1$  (empty green square) from those with  $t = -1$  (filled black square).

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI. Funded by Bibsam.

In the limit  $N \rightarrow \infty$ , the hidden weights of deep networks barely change under training [4,18,19]. For finite width, by contrast, the weights tend to change [5]. How the network output changes in response to the changing weights indicates how the network learns.

To understand how the trained network expresses the input-data features needed for classification, we ask how the network output depends on small input changes, encoded in the  $\mathbf{x}$  dependence of the maximal-FTLE field  $\lambda_1(\mathbf{x})$  of the trained network. For a classification problem with two-dimensional inputs  $\mathbf{x}$ , divided into two classes with targets  $t(\mathbf{x}) = \pm 1$  [Fig. 1(b)], we determine how the  $\mathbf{x}$  dependence of the maximal FTLE changes when changing  $N$  and  $L$ . For narrow networks, we find that the maximal-FTLE field forms ridges in the input plane, much like Lagrangian coherent structures in dynamical systems [20–22]. These ridges provide insight into the learning process, illustrating how the network learns to change its output by order unity in response to a small shift of the input pattern across the decision boundary. The ridges disappear as the network width grows, suggesting a different learning mechanism. For more complex classification problems (MNIST [23] and CIFAR-10 [24]), we show that FTLE structures in input space explain variations in classification accuracy and predictive uncertainty.

*Finite-time Lyapunov exponents.*—Figure 1(a) shows a fully connected feed-forward network with  $L$  hidden layers,  $N_0$  input components,  $N$  neurons per hidden layer, and  $N_{L+1} = 1$  output neuron. The network maps every input  $\mathbf{x}^{(0)} = \mathbf{x}$  to an output  $x^{(L+1)}$ . Weights and thresholds are varied to minimize the output error  $[x^{(L+1)} - t(\mathbf{x})]^2$ , so that the network predicts the correct target  $t(\mathbf{x})$  for each input  $\mathbf{x}$ . The sensitivity of  $\mathbf{x}^{(\ell)}$  to small changes  $\delta\mathbf{x}$  is determined by linearization,

$$\delta\mathbf{x}^{(\ell)} = \mathbb{D}^{(\ell)}\mathbb{W}^{(\ell)} \dots \mathbb{D}^{(2)}\mathbb{W}^{(2)}\mathbb{D}^{(1)}\mathbb{W}^{(1)}\delta\mathbf{x} \equiv \mathbb{J}_\ell\delta\mathbf{x}. \quad (1)$$

Here,  $\mathbb{W}^{(\ell)}$  are the weight matrices with elements  $w_{ij}^{(\ell)}$ , and  $\mathbb{D}^{(\ell)}$  are diagonal matrices with elements  $D_{ij}^{(\ell)} = g'(b_i^{(\ell)})\delta_{ij}$ , with  $b_i^{(\ell)} = \sum_{j=1}^{N_\ell} w_{ij}^{(\ell)} x_j^{(\ell-1)} - \theta_i^{(\ell)}$  and  $g'(b_i^{(\ell)}) = (d/db)g(b)|_{b=b_i^{(\ell)}}$ . The Jacobian  $\mathbb{J}_\ell(\mathbf{x})$  characterizes the growth or decay of small perturbations to  $\mathbf{x}$  [12,13]. Its maximal singular value  $\Lambda_1^{(\ell)}(\mathbf{x})$  increases or decreases exponentially as a function of  $\ell$ , with rate  $\lambda_1^{(\ell)}(\mathbf{x}) \equiv \ell^{-1} \log \Lambda_1^{(\ell)}(\mathbf{x})$ . The singular values  $\Lambda_1^{(\ell)}(\mathbf{x}) > \Lambda_2^{(\ell)}(\mathbf{x}) > \dots$  are the square roots of the non-negative eigenvalues of the right Cauchy-Green tensor  $\mathbb{J}_\ell^T(\mathbf{x})\mathbb{J}_\ell(\mathbf{x})$ . The maximal eigenvector of  $\mathbb{J}_\ell^T(\mathbf{x})\mathbb{J}_\ell(\mathbf{x})$  determines the direction of maximal stretching, i.e. in which input direction the output changes the most.

FTLEs and Cauchy-Green tensors are used in solid mechanics to identify elastic deformation patterns [25], and to find regions of instability in plastic deformation [26]

and crack initiation [27]. More generally, FTLEs help to characterize the sensitivity of complex dynamics to initial conditions [28–31]. In fluid mechanics, they explain the alignment of particles transported by the fluid [32,33], providing valuable insight into the stretching and rotation of fluid elements over time and space [34]. FTLEs allow us to identify Lagrangian coherent structures [20–22]; fluid-velocity structures that help to organize and understand complex spatiotemporal flow patterns [35]. These geometrical structures appear as ridges of large maximal FTLEs, orthogonal to the maximal stretching direction.

In applying these methods to neural networks, one should recognize several facts. First, in deep neural networks, the weights change from layer to layer. Therefore the corresponding dynamical system is not autonomous. Second, the number  $N_\ell$  of neurons per layer may change as a function of  $\ell$ , corresponding to a changing phase-space dimension. Third, the neural-network weights are trained. This limits the exponential growth of the maximal singular value  $\Lambda_1^{(L)}$ , and it causes the FTLE  $\lambda_1^{(L)}(\mathbf{x})$  to remain  $\mathbf{x}$  dependent, even in the limit of large  $L$  (discussed in more detail below). Fourth, one can use different activation functions, such as the piecewise linear ReLU function [11], or the smooth tanh function [15]. Here we use  $g(b) = \tanh(b)$ , so that the network map is continuously differentiable just like the dynamical systems for which Lagrangian coherent structures were found and analyzed.

*Two-dimensional dataset.*—To illustrate the geometrical structures formed by the maximal FTLE, we first consider a toy problem. The dataset [Fig. 1(b)] comprises  $4 \times 10^4$  input patterns, with 90% used for training, the rest for testing. We trained fully connected feed-forward networks on this dataset by stochastic gradient descent, minimizing the output error  $[x^{(L+1)} - t(\mathbf{x})]^2$ . In this way we obtained classification accuracies of at least 98%. We used different layouts, changing the numbers of layers and hidden neurons per layer. The weights were initialized as independent Gaussian random numbers with zero mean and variance  $\sigma_\ell^2 \sim N_\ell^{-1}$ . The thresholds were initialized to zero (see the Supplemental Material [36] for details). After training, we computed the maximal FTLE in layer  $L$  and the associated stretching direction from Eq. (1) as described in Refs. [38,39].

The results are summarized in Fig. 2, which shows maximal-FTLE fields for trained networks with different layouts. We see that the ridges of large positive  $\lambda_1^{(L)}(\mathbf{x})$  align with the decision boundary between the two classes [Fig. 1(b)]. The network learns by grouping the inputs into two different basins of attraction for  $t = \pm 1$ , separated by a ridge of positive  $\lambda_1^{(L)}(\mathbf{x})$ . A small shift of the input across the decision boundary leads to a substantial change in the output. In other words, a large maximal FTLE quantifies exponential expressivity of the network near the ridge. This is consistent with the observation that the output is



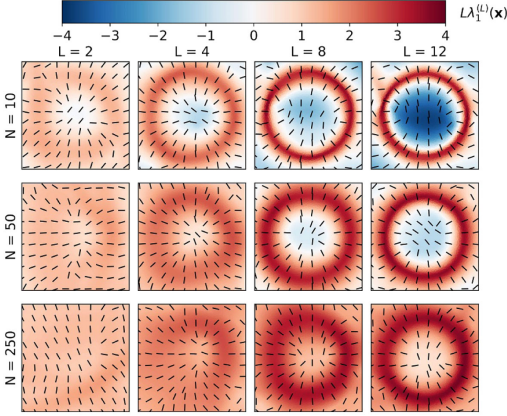


FIG. 2. Geometrical FTLE structures in input space for different widths  $N$  and depths  $L$  of fully connected feed-forward neural networks trained on the dataset from Fig. 1(b). Shown is the magnitude of  $L\lambda_1^{(L)}(\mathbf{x})$ , and the maximal stretching directions (black lines).

particularly sensitive to weight changes near decision boundaries [8]. The ridges are most prominent for small  $N$  and large  $L$ . The contrast between ridge and background increases as  $L$  becomes larger, quantifying the exponential expressivity of deep networks [1]. For larger  $L$ , the network can resolve smaller input distances  $\delta\mathbf{x}$  because the singular values increase or decrease exponentially from layer to layer.

The ridges of large maximal FTLE are Lagrangian coherent structures, because the maximal stretching directions (solid lines in Fig. 2) become orthogonal to the ridges for large  $L$ . This demonstrates that there is a stringent analogy between the FTLE ridges of deep neural networks and Lagrangian coherent structures.

The ridges gradually disappear as the number  $N$  of hidden neurons per layer increases, because the maximal singular value of  $\mathbb{J}_L(\mathbf{x})$  approaches a definite  $\mathbf{x}$ -independent limit as  $N \rightarrow \infty$  at fixed  $L$ . But how can the network distinguish inputs with different targets in this case, without ridges indicating decision boundaries? One possibility is that the large number of hidden neurons allows the network to embed the inputs into a high-dimensional space where they can be separated thanks to the universal approximation theorem [40]. In this case, training only the output weights (and threshold) suffices, as demonstrated by Fig. 3(a). That the classification error with random hidden weights is larger than that of the fully trained network is not surprising, since different random embeddings have different classification errors when the number of patterns exceeds twice the embedding dimension [11,41]. In summary, large- $N$  networks can classify with random hidden weights. This implies that the hidden weights do not need to change

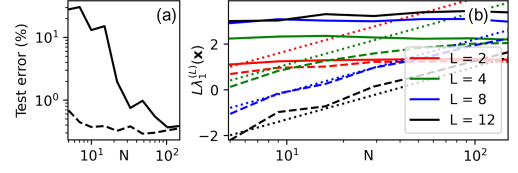


FIG. 3. (a) Classification error for a fully connected feed-forward network with  $L = 2$  hidden layers with untrained, random hidden weights, and trained output weights, as a function of  $N$  (solid black line). Also shown is the classification error for the fully trained network (dashed line). Both curves were obtained for the dataset shown schematically in Fig. 1. (b) Quantification of the crossover seen in Fig. 2, for fully trained networks. Shown are the averages  $\langle \lambda_1^{(L)}(\mathbf{x}) \rangle_d$  (solid) and  $\langle \lambda_1^{(L)}(\mathbf{x}) \rangle_c$  (dashed), see text. The data was obtained by averaging over independent initial-weight realizations. Also shown is a fit to  $L\langle \lambda_1^{(L)}(\mathbf{x}) \rangle_c \approx -C_c L + \log N$  (dotted).

during training, just as in kernel regression with the neural-tangent kernel [4,18]. In other words, the learning in this regime is lazy [19].

Figure 3(b) describes in more detail the crossover between the two learning regimes in Fig. 2. Shown are local averages of the maximal FTLE on the decision boundary,  $\langle \lambda_1^{(L)}(\mathbf{x}) \rangle_d$ , and in the center of the input plane,  $\langle \lambda_1^{(L)}(\mathbf{x}) \rangle_c$ , as functions of  $N$  for different values of  $L$ . The results were obtained by training for 200 epochs to reach classification accuracies  $\geq 99\%$ . Although the details of the maximal-FTLE field change upon training further, Fig. 3(b) allows us to draw the following conclusions about the transition. First,  $L\langle \lambda_1^{(L)}(\mathbf{x}) \rangle_d$  tends to an  $N$ -independent constant as  $L$  increases,  $L\langle \lambda_1^{(L)}(\mathbf{x}) \rangle_d \rightarrow C_d$ . This saturation is due to training: the network learns to produce output differences of the order of  $\delta x^{(L+1)} \sim 1$ , and to resolve input differences  $\delta\mathbf{x}$  on the scale of the mean distance  $|\delta\mathbf{x}^{(0)}|$  between neighboring inputs over the decision boundary. Second, the data shown in Fig. 3(b) suggest that  $L\langle \lambda_1^{(L)}(\mathbf{x}) \rangle_c \approx -C_c L + \log N$  for large enough  $L$ . Third, the contrast between ridges and background disappears upon increasing  $N$ , when the background reaches the ridge level,  $\langle \lambda_1^{(L)}(\mathbf{x}) \rangle_c \approx \langle \lambda_1^{(L)}(\mathbf{x}) \rangle_d$ . At this point the learning mechanism transitions from learning by ridges to random embedding. In Fig. 3(b) this happens around  $N_c \sim \exp(C_d + C_c L)$ . While the precise form of the law may depend on the training details, the general conclusion is that  $N_c$  depends very sensitively on  $L$ , because the  $N$  dependence of the relation is logarithmic, just as the  $N \rightarrow \infty$  result for the Lyapunov exponent quoted above. We remark that the  $L$  scaling discussed above implies that  $\lambda_1^{(L)}(\mathbf{x})$  remains  $\mathbf{x}$  dependent for large  $L$ .

One may wonder how the FTLE fields in Fig. 2 depend on the initial weight variance  $\sigma^2$ . For  $GN\sigma^2 < 1$ , the FTLEs are negative on average, initially. This implies a slowing down of the initial training (vanishing-gradient problem). To see this, consider the fundamental forward-backward dichotomy of deep neural networks [11]: weight updates in the stochastic-gradient algorithm are given by  $\delta w_{mn}^{(\ell)} \propto \Delta_m^{(\ell)} x_n^{(\ell-1)}$ , where

$$[\Delta^{(\ell)}]^T = [\Delta^{(L)}]^T \mathbb{D}^{(L)} \mathbb{W}^{(L)} \dots \mathbb{D}^{(\ell+1)} \mathbb{W}^{(\ell+1)} \mathbb{D}^{(\ell)}, \quad (2)$$

and  $\Delta_j^{(L)} = g'(b^{(L+1)})[x^{(L+1)} - t(x)]g'(b_j^{(L)})w_j^{(L+1)}$ . It follows from Eq. (2) that negative FTLEs cause small weight increments  $\delta w_{mn}^{(\ell)}$ . Conversely, when the maximal FTLE is positive and too large, the weights grow rapidly, leading to training instabilities.

Remarkably, training has a self-organizing effect. After training, the maximal-FTLE distribution becomes independent of the initial  $\sigma^2$ , provided that the initial maximal FTLE is not too large (Fig. S1 in [36]). For small enough  $N$ , in particular, the distribution centers around zero. This is explained by the fact that the network learns by creating maximal-FTLE ridges in input space: in order to accommodate positive and negative  $\lambda_1^{(L)}(x)$ , the distribution must center around zero, alleviating the unstable-gradient problem. We remark that the shape of the distribution, the tails in particular, continues to evolve as one trains further.

**MNIST dataset.**—This dataset consists of 60 000 images of handwritten digits 0 to 9. Each grayscale image has  $28 \times 28$  pixels and was pre-processed to facilitate machine learning [23]. Deep neural networks can achieve high precision in classifying this data, with accuracies of up to 99.77% on a test set of 10 000 digits [42].

We determined the maximal-FTLE field for this dataset for a network with  $L = 16$  hidden layers, each containing  $N = 20$  neurons, and a standard softmax layer with ten outputs [11]. To visualize the geometrical structures in the  $28^2$ -dimensional input space, we projected it onto two dimensions as follows [43]. We added a bottleneck layer with two neurons to the fully trained network, just before the softmax-output layer. We retrained only the weights and thresholds of this additional layer and the output layer, keeping all other hidden neurons unchanged. The local fields  $b_1$  and  $b_2$  of the two bottleneck neurons are the coordinates of the two-dimensional representation shown in Fig. 4(a). We see that the input data separate into ten distinct clusters corresponding to the ten digits. The maximal FTLEs at the center of these clusters are very small or even negative, indicating that the output is not sensitive to small input changes. These regions are delineated by areas with significantly larger positive FTLEs [see  $3\times$  enlargement in panel (a)]. Figure 2 leads us to expect that patterns with large  $\lambda_1^{(L)}(x)$  are located near the decision boundaries in high-dimensional input space. This is verified

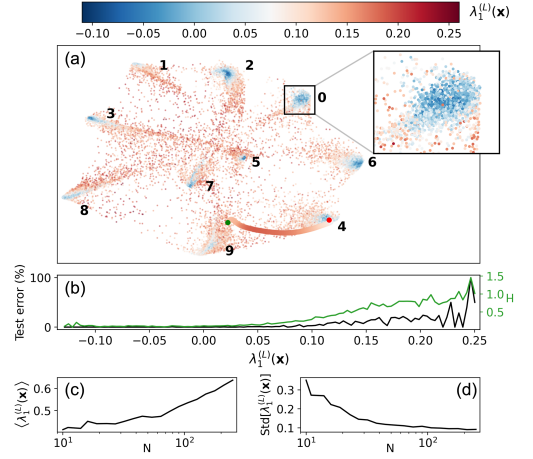


FIG. 4. Maximal-FTLE field for MNIST [23]. A fully connected feed-forward network with  $N = 20$  neurons per hidden layer,  $L = 16$  hidden layers, and a softmax layer with ten outputs was trained to a classification accuracy of 98.88%. The maximal FTLE was calculated for each of the  $28^2$ -dimensional inputs and projected onto two dimensions (see text). (a) Training data in the nonlinear projection. For each input, the maximal FTLE  $\lambda_1^{(L)}$  is shown color coded. The box contains 93% of the recognized digits 0. A  $3\times$  zoom of this box is also shown. The colored line represents an adversarial attack from 9 to 4 (see text), with  $\lambda_1^{(L)}(x)$  color coded. (b) Classification error on the test set and predictive uncertainty  $H$  (see text) as functions of  $\lambda_1^{(L)}(x)$ . (c),(d) Mean and standard deviation of maximal-FTLE distribution versus  $N$ .

by strong correlations between  $\lambda_1^{(L)}(x)$  and both the classification error and the predictive uncertainty. Figure 4(b) shows that the classification error on the test set is larger for inputs  $x$  with larger  $\lambda_1^{(L)}(x)$ , and that large values of  $\lambda_1^{(L)}(x)$  correlate with high predictive uncertainty, measured by the entropy  $H$  of the posterior predictive distribution [44]. For softmax outputs, where  $x_i^{(L+1)}$  can be interpreted as probabilities,  $H = -\sum_i \langle x_i^{(L+1)} \rangle \log \langle x_i^{(L+1)} \rangle$ , where  $\langle \cdot \rangle$  denotes an average over an ensemble of networks with the same layout but different weight initializations [45,46]. These observations confirm that ridges of maximal FTLEs localize the decision boundaries in input space. Similar conclusions hold for other architectures (Fig. S2 in [36]), and for the more complex CIFAR-10 dataset (Fig. S3).

Figure 4(a) also shows  $\lambda_1^{(L)}(x)$  along a path generated by an adversarial attack. The attack begins from a sample within the cluster corresponding to the digit 9 and aims to transform it into a digit 4 by making small perturbations to the input data [47] toward class 4. We see that the maximal FTLE is small at first, then increases as the path approaches the decision boundary, and eventually decreases again. This

indicates, first, that our conclusions regarding the correlations between large maximal FTLEs and decision boundaries extend to neighborhoods of the MNIST training set that contain patterns the network has not encountered during training. Second, the maximal stretching direction correlates with the direction in input space found by the adversarial attack (Fig. S4 [36]). Third, since it is hard to locate the FTLE ridges in the high-dimensional input space, we characterized the transition between the two learning regimes in terms of the mean and the standard deviation of the FTLE distribution. Figure 4(c) shows that the mean becomes positive and that the variance tends to zero as  $N$  grows, leading to the uniform FTLE field characteristic of learning by random embedding.

**Conclusions.**—For deep neural networks trained on different classification problems, we explored geometrical structures of finite-time Lyapunov exponents in input space. In fluid mechanics, such Lagrangian coherent structures appear as ridges of large exponents, and they are used with great success to organize the phase space of complex spatiotemporal flow patterns. The same is true for deep neural networks: FTLE ridges partition input space into different regions associated with different classes. Our analysis shows how the network exploits its exponential expressivity to form the ridges. Their sharpness determines how quickly classification errors and prediction uncertainty decreases as one moves away from the ridge. As the width of the network increases, the contrast between ridge and background disappears, leading to a different learning mechanism, random embedding, with qualitative differences regarding classification errors and predictive uncertainties. The transition to this lazy-learning regime [4,18,19] occurs for very wide networks, with widths  $\sim \exp(L)$ . The transition may explain why wider networks are more robust against adversarial attacks [47]: the less important the ridges are for representing the relevant data structures, the harder it is to realize adversarial attacks. The geometrical method presented here may extend to other network architectures, such as resnets [48] or transformers [49], and could help to visualize and understand the mechanisms that allow such neural networks to learn, but this remains a question for the future.

We thank A. Dubey and J. Meibohm for discussions concerning FTLE distributions for neural networks with random weights. L. S. was supported by grants from the Knut and Alice Wallenberg (KAW) Foundation (No. 2019.0079) and Vetenskapsrådet (VR), No. 2021-4452. J. B. received support from UCA-JEDI Future Investments (Grant No. ANR-15-IDEX-01). H. L. was supported by a grant from the KAW Foundation. K. G. was supported by VR Grant No. 2018-03974, and B. M. by VR Grant No. 2021-4452. Some of the numerical computations for this project were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC).

- [1] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli, Exponential expressivity in deep neural networks through transient chaos, *Adv. Neural Inf. Process. Syst.* **29**, 3360 (2016), [https://proceedings.neurips.cc/paper\\_files/paper/2016/file/148510031349642de5ca0c544f31b2ef-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2016/file/148510031349642de5ca0c544f31b2ef-Paper.pdf).
- [2] J. Jumper *et al.*, Highly accurate protein structure prediction with AlphaFold, *Nature (London)* **596**, 583 (2021).
- [3] M. Tan and Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in *International Conference on Machine Learning* (PMLR, Cambridge, 2019), pp. 6105–6114.
- [4] A. Jacot, F. Gabriel, and C. Hongler, Neural tangent kernel: Convergence and generalization in neural networks, *Adv. Neural Inf. Process. Syst.* **31**, 8580 (2018), [https://proceedings.neurips.cc/paper\\_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf).
- [5] M. Geiger, L. Petri, and M. Wyart, Landscape and training regimes in deep learning, *Phys. Rep.* **924**, 1 (2021).
- [6] P. M. Nguyen and H. T. Pham, A rigorous framework for the mean field limit of multilayer neural networks, *arXiv*: 2001.11443.
- [7] I. Seroussi, G. Naveh, and R. Zohar, Separation of scales and a thermodynamic description of feature learning in some CNNs, *Nat. Commun.* **14**, 908 (2023).
- [8] A. Baratin, T. George, C. Laurent, R. D. Hjelm, G. Lajoie, P. Vincent, and S. Lacoste-Julien, Implicit regularization via neural feature alignment, in *International Conference on Artificial Intelligence and Statistics* (PMLR, Cambridge, 2021), pp. 2269–2277.
- [9] J. Paccolat, L. Petri, M. Geiger, K. Tyloo, and M. Wyart, Geometric compression of invariant manifolds in neural networks, *J. Stat. Mech.* (2021) 044001.
- [10] R. Pacelli, S. Ariosto, M. Pastore, F. Ginelli, M. Gherardi, and P. Rotondo, A statistical mechanics framework for Bayesian deep neural networks beyond the infinite-width limit, *Nat. Mach. Intell.* **5**, 1497 (2023).
- [11] B. Mehlig, *Machine Learning with Neural Networks: An Introduction for Scientists and Engineers* (Cambridge University Press, Cambridge, England, 2021).
- [12] E. Ott, *Chaos in Dynamical Systems*, 2nd ed. (Cambridge University Press, Cambridge, England, 2002).
- [13] P. Cvitanović, R. Artuso, R. Mainieri, G. Tanner, and G. Vattay, *Chaos: Classical and Quantum* (Niels Bohr Institute, Copenhagen, 2020).
- [14] A. Crisanti, A. Vulpiani, and G. Paladin, *Products of Random Matrices in Statistical Physics* (Springer, Berlin, 1993).
- [15] J. Pennington, S. Schoenholz, and S. Ganguli, Resurrecting the sigmoid in deep learning through dynamical isometry: Theory and practice, *Adv. Neural Inf. Process. Syst.* **30**, 4785 (2017), [https://proceedings.neurips.cc/paper\\_files/paper/2017/file/d9fc0cdb67638d50f411432d0d41d0ba-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2017/file/d9fc0cdb67638d50f411432d0d41d0ba-Paper.pdf).
- [16] I. Sutskever, J. Martens, G. Dahl, and G. E. Hinton, On the importance of initialization and momentum in deep learning, in *Proceedings of the 30th International Conference on Machine Learning—Volume 28* (PMLR, Cambridge, 2013), pp. III–1139–III–1147.
- [17] X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in *Proceedings of the Thirteenth International Conference on Artificial*

- Intelligence and Statistics*, Proceedings of Machine Learning Research Vol. 9, edited by Y. W. Teh and M. Titterton (JMLR Workshop and Conference Proceedings, Chia Laguna Resort, Sardinia, Italy, 2010), pp. 249–256.
- [18] S. Chen, H. He, and W. Su, Label-aware neural tangent kernel: Toward better generalization and local elasticity, *Adv. Neural Inf. Process. Syst.* **33**, 15847 (2020), [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/b6b90237b3ebd1e462a5d11dbc5c4dae-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/b6b90237b3ebd1e462a5d11dbc5c4dae-Paper.pdf).
- [19] L. Chizat, E. Oyallon, and F. Bach, On lazy training in differentiable programming, *Adv. Neural Inf. Process. Syst.* **32**, 2933 (2019), [https://proceedings.neurips.cc/paper\\_files/paper/2019/file/ae614c557843b1df326cb29c57225459-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2019/file/ae614c557843b1df326cb29c57225459-Paper.pdf).
- [20] G. Haller and G. Yuan, Lagrangian coherent structures and mixing in two-dimensional turbulence, *Physica (Amsterdam)* **147D**, 352 (2000).
- [21] V. Lucarini and T. Bódi, Edge states in the climate system: Exploring global instabilities and critical transitions, *Nonlinearity* **30**, R32 (2017).
- [22] M. Benítez, Y. Duguet, P. Schlatter, and D. S. Henningson, Edge manifold as a Lagrangian coherent structure in a high-dimensional state space, *Phys. Rev. Res.* **2**, 033258 (2020).
- [23] Y. LeCun, C. Cortes, and C. J. C. Burges, The MNIST database of handwritten digits, [yann.lecun.com/exdb/mnist](http://yann.lecun.com/exdb/mnist) (2018).
- [24] A. Krizhevsky, G. Hinton *et al.*, Learning multiple layers of features from tiny images, Technical report, Computer Science University of Toronto, Canada, (2009).
- [25] C. Truesdell and W. Noll, in *The Non-Linear Field Theories of Mechanics*, Encyclopedia of Physics Vol. III, edited by S. Flügge (Springer, Berlin, 1965), pp. 1–579.
- [26] J. L. Ren, C. Chen, Z. Y. Liu, R. Li, and G. Wang, Plastic dynamics transition between chaotic and self-organized critical states in a glassy metal via a multifractal intermediate, *Phys. Rev. B* **86**, 134303 (2012).
- [27] X. Jin, Y. Chen, L. Wang, H. Han, and P. Chen, Failure prediction, monitoring and diagnosis methods for slewing bearings of large-scale wind turbine: A review, *Measurement* **172**, 108855 (2021).
- [28] G. Haller, Lagrangian coherent structures, *Annu. Rev. Fluid Mech.* **47**, 137 (2015).
- [29] S. Vannitsem, Predictability of large-scale atmospheric motions: Lyapunov exponents and error dynamics, *Chaos* **27**, 032101 (2017).
- [30] A. Uthamacumaran, A review of dynamical systems approaches for the detection of chaotic attractors in cancer networks, *Patterns* **2**, 100226 (2021).
- [31] A. Morozov, K. Abbott, K. Cuddington, T. Francis, G. Gellner, A. Hastings, Y.-C. Lai, S. Petrovskii, K. Scranton, and M. L. Zeeman, Long transients in ecology: Theory and applications, *Phys. Life Rev.* **32**, 1 (2020).
- [32] R. Ni, N. T. Ouellette, and G. A. Voth, Alignment of vorticity and rods with Lagrangian fluid stretching in turbulence, *J. Fluid Mech.* **743**, R3 (2014).
- [33] V. Bezuglyy, M. Wilkinson, and B. Mehlig, Poincaré indices of rheoscopic visualisations, *Europhys. Lett.* **89**, 34003 (2009).
- [34] P. L. Johnson, S. S. Hamilton, R. Burns, and C. Meneveau, Analysis of geometrical and statistical features of Lagrangian stretching in turbulent channel flow using a database task-parallel particle tracking algorithm, *Phys. Rev. Fluids* **2**, 014605 (2017).
- [35] J. F. Gibson, J. Halcrow, and P. Cvitanovic, Visualizing the geometry of state space in plane Couette flow, *J. Fluid Mech.* **611**, 107 (2008).
- [36] See Supplemental Material at <http://link.aps.org/supplemental/10.1103/PhysRevLett.132.057301> for detailed information regarding the experimental setup and the theoretical models. It also contains Ref. [37].
- [37] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, Automatic differentiation in PyTorch, Technical report, (2017), <https://openreview.net/forum?id=BJJsrnfCZ>.
- [38] J. P. Eckmann and D. Ruelle, Ergodic theory of chaos and strange attractors, *Rev. Mod. Phys.* **57**, 617 (1985).
- [39] T. Okushima, New method for computing finite-time Lyapunov exponents, *Phys. Rev. Lett.* **91**, 254101 (2003).
- [40] P. Kidger and T. Lyons, Universal approximation with deep narrow networks, [arXiv:1905.08539](https://arxiv.org/abs/1905.08539).
- [41] T. M. Cover, Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition, *IEEE Trans. Electron. Comput.* **EC-14**, 326 (1965).
- [42] D. Ciregan, U. Meier, and J. Schmidhuber, Multi-column deep neural networks for image classification, in *2012 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE Computer Society, Silver Spring, 2012), pp. 3642–3649.
- [43] Y. Wang, H. Yao, and S. Zhao, Auto-encoder based dimensionality reduction, *Neurocomputing* **184**, 232 (2016).
- [44] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher *et al.*, A survey of uncertainty in deep neural networks, [arXiv:2107.03342](https://arxiv.org/abs/2107.03342).
- [45] B. Lakshminarayanan, A. Pritzel, and C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, *Adv. Neural Inf. Process. Syst.* **30**, 6402 (2017), [https://proceedings.neurips.cc/paper\\_files/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf).
- [46] L. Hoffmann and C. Elster, Deep ensembles from a Bayesian perspective, [arXiv:2105.13283](https://arxiv.org/abs/2105.13283).
- [47] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, Towards deep learning models resistant to adversarial attacks, in *International Conference on Learning Representations* (International Conference on Learning Representations, Vancouver, 2018).
- [48] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.
- [49] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, An image is worth 16 × 16 words: Transformers for image recognition at scale, [arXiv:2010.11929](https://arxiv.org/abs/2010.11929).

# Paper B

<https://iopscience.iop.org/article/10.1088/2632-2153/aca1f6/pdf>





## PAPER

## OPEN ACCESS

RECEIVED  
3 June 2022REVISED  
27 September 2022ACCEPTED FOR PUBLICATION  
10 November 2022PUBLISHED  
5 December 2022

Original Content from  
this work may be used  
under the terms of the  
[Creative Commons  
Attribution 4.0 licence](#).

Any further distribution  
of this work must  
maintain attribution to  
the author(s) and the title  
of the work, journal  
citation and DOI.



# Constraints on parameter choices for successful time-series prediction with echo-state networks

L Storm\*, K Gustavsson and B Mehlig\*

Department of Physics, Gothenburg University, 41296 Gothenburg, Sweden

\* Authors to whom any correspondence should be addressed.

E-mail: [ludvig.storm@physics.gu.se](mailto:ludvig.storm@physics.gu.se) and [bernhard.mehlig@physics.gu.se](mailto:bernhard.mehlig@physics.gu.se)**Keywords:** echo-state networks, reservoir computing, dynamical systems, Lyapunov exponent

## Abstract

Echo-state networks are simple models of discrete dynamical systems driven by a time series. By selecting network parameters such that the dynamics of the network is contractive, characterized by a negative maximal Lyapunov exponent, the network may synchronize with the driving signal. Exploiting this synchronization, the echo-state network may be trained to autonomously reproduce the input dynamics, enabling time-series prediction. However, while synchronization is a necessary condition for prediction, it is not sufficient. Here, we study what other conditions are necessary for successful time-series prediction. We identify two key parameters for prediction performance, and conduct a parameter sweep to find regions where prediction is successful. These regions differ significantly depending on whether full or partial phase space information about the input is provided to the network during training. We explain how these regions emerge.

## 1. Introduction

Many driven dynamical systems can be found in nature and engineering. Reservoir computing has recently become popular to study in this context, as it yields simple models of such dynamical systems. By exploiting signal-driven synchronization, where the dynamics of the reservoir neurons synchronizes with the input time series, a reservoir computer can be trained to reproduce a time series autonomously [1–4]. A necessary condition for the synchronization to occur is that the dynamics of the reservoir neurons is contractive; a property ensured by the reservoir dynamics having a negative maximal Lyapunov exponent. In reservoir computing literature, the ability to synchronize is referred to as the *echo-state property*, a term coined by Jaeger in his original paper on echo-state networks (ESNs) [5], which is the most common realisation of reservoir networks. The maximal Lyapunov exponent has been the focus of study in several papers due to its close connection to the echo-state property [6–8]. There is some variation in how the maximal Lyapunov exponent has been defined. In [6], the Lyapunov exponent is defined in the absence of input. However, as the input has been shown to have a contractive effect on the reservoir dynamics when using the commonly employed tanh activation function [7], the maximal Lyapunov exponent defined in the presence of input is more naturally connected to the echo-state property.

While the echo-state property is a necessary condition for the reproduction of a time series, it is not sufficient. The ability for a reservoir network to reproduce a time series has recently been formally connected to time-delay embedding [9]. The result states that the embedding is possible because the dependence on previous inputs decays at different rates for different neurons in the reservoir, creating an internal representation that captures different time scales of the input time series. The rate at which dependence on previous inputs of a given neuron decays is controlled by the strength of the input and recurrent connections, as these control the strength of the driving and the time scale of the recurrent dynamics of that neuron. In fact, using time delay embedding, it is possible to reproduce a time series with only partial phase space information. By partial phase space information is meant that only a subset of the components of the time series is used when making the prediction of the time series. The connection between the ability to represent several time scales and prediction performance was first observed in [5, 10] and has inspired the

design heuristic that the reservoir dynamics should be ‘rich’ in the sense that the different neurons should display a wide range of dynamics that captures different time scales of the time series. However, other results show that the reservoir connections, which allow the reservoir to represent temporal information, can be removed while still maintaining good prediction performance [11, 12]. In this case, time delay embedding is not possible. It is clear that such networks cannot reproduce dynamics with only partial phase space information. The distinction between full and partial-information tasks in reservoir computing was made in [13], labelled as non-temporal and temporal tasks respectively, but distinctions between how the reservoir should be designed in the two cases were not discussed.

In this paper, we investigate the differences in parameter dependence when full or partial phase space information is provided to an ESN. We begin by showing that, in the limit of large network dimension, and for a given input time series, the maximal Lyapunov exponent depends only on two parameters that combine several tuning parameters, namely the reservoir dimension, the scale of the reservoir connections (here quantified as the variance of the connection weights), the sparsity of the reservoir connectivity matrix, and the dimension and scale of the input. Sweeping the two parameters identified, we study the difference between the regions where reservoir computing is successful for the cases of full and partial information, and explain the shape of these regions. This includes showing why the maximal Lyapunov exponent has a lower boundary in the case of partial information, and how the commonly employed ridge parameter introduces a lower boundary of the input scale for successful reservoir computing. A condition for successful prediction in the partial-information case is shown to imply that the commonly employed metric for linear information, *memory capacity* [5], must be low, implying that maximizing this metric is counterproductive when optimizing performance. Additionally, we show that results concerning the sampling rate in time-delay embedding theory [14] can be applied to the case of partial information to improve performance.

The paper is structured as follows: First, we provide some background on the theory of ESNs and how their predictive performance is evaluated. In the following section, we derive a mean-field expression for the maximal Lyapunov exponent using random-matrix theory, arriving at the same result as in [7], but extending it to more general input time series rather than Gaussian noise. This is followed by a section where we describe the methods we use. We then present the results for the case of full and partial phase space information. We conclude with a discussion of the results.

## 2. Background

### 2.1. Echo-state networks

The ESN training dynamics for a reservoir with  $N$  neurons and an input signal with  $n$  components are given by [15]

$$r_i(t+1) = g \left( \sum_{j=1}^N A_{ij} r_j(t) + \sum_{\alpha=1}^n W_{i\alpha}^{(in)} u_\alpha(t) \right), \tag{1a}$$

$$v_i(t+1) = \sum_{j=1}^N W_{ij}^{(out)} f(r_j(t+1)). \tag{1b}$$

Here  $r_i(t)$  is the state of the  $i$ :th reservoir neuron at time  $t$ , and  $u_\alpha(t)$  is the  $\alpha$ :th component of the input signal. The matrix  $\mathbf{A}$  is the reservoir connection matrix whose entries  $A_{ij}$  represent the connection strength between the reservoir nodes, while  $\mathbf{W}^{(in)}$  are the connections between the input and the reservoir.  $g(\cdot)$  is the activation function, and  $f(\cdot)$  is applied to the reservoir states before it is projected to the output space with the output weight matrix  $\mathbf{W}^{(out)}$ . The argument of the activation function is referred to as the local field.  $f(\cdot)$  is often set to be the identity function. In this work, to break the inherent symmetry of the reservoir dynamics which causes the ESN to learn the reflected input series  $\mathbf{u} \rightarrow -\mathbf{u}$  as well as the original, we employ the Lu readout [3].

During prediction, we follow the standard procedure introduced in [5] and replace the input  $u_\alpha(t)$  by the output  $v_\alpha(t)$  of the reservoir to form an autonomous system,

$$r_i(t+1) = g \left( \sum_{j=1}^N A_{ij} r_j(t) + \sum_{\alpha=1}^n W_{i\alpha}^{(in)} v_\alpha(t) \right), \tag{2a}$$



$$v_i(t+1) = \sum_{j=1}^N W_{ij}^{(out)} f(r_j(t+1)). \tag{2b}$$

This is the prediction dynamics.

**2.2. Training and evaluation**

In order to train the ESN, the training dynamics (1) is run for some time using the input time series to ensure that the reservoir dynamics has synchronized with the input. Then, at time  $t = 0$ , an  $2N \times T_{max}$  matrix  $\mathbf{R}$  is formed where each column is the reservoir state vectors  $\mathbf{r}(t)$  and  $\mathbf{r}^2(t)$  concatenated (due to the Lu readout) at each time  $t = 0, 1, \dots, T_{max} - 1$ . We wish to minimize the quadratic error between the output  $\mathbf{v}(t)$  and the target  $\mathbf{y}(t) = \mathbf{u}(t)$  and achieve this by employing ridge regression [16] to obtain

$$\mathbf{W}^{(out)} = \mathbf{Y}\mathbf{R}^T (\mathbf{R}\mathbf{R}^T + k\mathbf{I})^{-1}. \tag{3}$$

Here,  $\mathbf{Y}$  is a matrix whose columns are given by  $\mathbf{y}(t)$ , and  $k \geq 0$  is the ridge parameter which is introduced to reduce overfitting. An additional effect of the ridge parameter is that the magnitude of the entries in  $\mathbf{W}^{(out)}$  decreases as  $k$  increases.

Once  $\mathbf{W}^{(out)}$  has been determined, the prediction dynamics (2) is used to autonomously predict how the time series continues. We now define an error function which will be used to measure the prediction performance of the trained network. In order to evaluate the prediction performance of the ESN, we monitor

$$\varepsilon_\alpha(t) = \sqrt{\frac{(y_\alpha(t) - v_\alpha(t))^2}{\sigma_{y_\alpha}^2}}, \tag{4}$$

where  $\sigma_{y_\alpha}^2$  is the variance of the  $\alpha$ :th component of the time series. The quantity  $\varepsilon_\alpha(t)$  quantifies how many standard deviations the  $\alpha$ :th component of the prediction deviates from the target time series. When any of the predicted components deviates more than some threshold value, the time is recorded as the successful prediction time. We set the threshold value to 0.5. Decreasing this value does not qualitatively affect the obtained results. As this quantity fluctuates depending on the random initialisation of the ESN and from where in the time series the prediction started, the final performance score is determined by an average over several random initialisations of both the ESN and initial value of the time series. As the quantity is standardized, the metric is comparable for different time series.

**2.3. Parameters**

In designing an ESN, several parameters must be selected. As they are central to this work, we summarise the relevant parameters here. The parameters that are mainly discussed in literature are the reservoir dimension  $N$ , the scale of the reservoir connectivity matrix  $\sigma_A^2$ , which is the variance of the entries in  $\mathbf{A}$  (the spectral radius is sometimes used instead as a scale metric), the sparsity of the connections in the reservoir  $s$ , which takes the value  $s = 1$  if all neurons are connected and  $s = 0$  if no neurons are connected, the input dimension  $n$ , and the scale of the input  $\sigma_{in}^2$ , which is the variance of the entries of  $\mathbf{W}^{(in)}$ . These are parameters pertaining to the architecture of the ESN. In addition, the ridge parameter  $k$  used during training and the sampling rate  $\delta t$  of the time series are important tuning parameters. In this work, we assume that  $N$  is sufficiently large so that the sum over reservoir states in (1a) and (2a) can be approximated as a random variable with a Gaussian distribution with mean zero (due to the distribution of the reservoir connections  $A_{ij}$ ) and variance  $sN\sigma_A^2$ . In this limit, it is unnecessary to vary  $s$ ,  $N$ , and  $\sigma_A^2$  independently when selecting reservoir parameters, which is often done in literature, see for example [5, 13]. For a given input series, the reservoir dynamics thus only depends on two parameters, namely  $sN\sigma_A^2$  and  $n\sigma_{in}^2$ . In the remainder of the article, these two parameters are used to investigate parameter regions where reservoir computing is successful.

**3. Maximal Lyapunov exponent**

The maximal Lyapunov exponent of a dynamical system describes the long term fate of the separation of two initially nearby trajectories [17]. The quantity is computed under the assumption that the separation remains small within the time frame of interest, and as such, we can consider the linearised dynamics of the system to describe the evolution of the separation. For ESNs, it is possible to define three different Lyapunov exponents by considering different dynamical systems: (i) system (1a) with  $\sigma_{in}^2 = 0$ , (ii) system (1a with  $\sigma_{in}^2 > 0$ , and (iii) system (2) for a trained ESN. In [6], definition (i) was employed. However, definition (ii) must be used if one wants to quantify the echo-state property, because the input has a contracting effect

on the reservoir dynamics when the tanh activation function is employed [7]. It is therefore more natural to study the latter definition. Finally, if an ESN has been trained successfully, the third definition of the exponent approximate the maximal Lyapunov exponent of the input dynamics, as shown in [1]. We mainly focus on definition (ii) and refer to this as the training Lyapunov exponent  $\lambda_T$ .

For an ESN employing the tanh activation function, we may compute the linearised separation of reservoir states  $\delta\mathbf{r}(t)$  in the presence of input as

$$\delta\mathbf{r}(t+1) = \mathbf{D}(t)\mathbf{A}\delta\mathbf{r}(t), \tag{5}$$

where  $\mathbf{D}(t)$  is a diagonal matrix with entries  $D_{ii}(t) = 1 - \tanh^2(b_i(t))$ , where  $b_i(t) = \sum_{j=1}^N A_{ij}r_j(t) + \sum_{\alpha} W_{i\alpha}^{(in)} u_{\alpha}(t)$ . The training Lyapunov exponent is obtained by computing [17]

$$\lambda_T = \lim_{t \rightarrow \infty} \frac{1}{t} \log \frac{|\mathbf{D}(t-1)\mathbf{A}\mathbf{D}(t-2)\mathbf{A}\dots\mathbf{D}(0)\mathbf{A}\delta\mathbf{r}(0)|}{|\delta\mathbf{r}(0)|}. \tag{6}$$

Numerically, the product in (6) can be computed employing the QR method [18] and computing the average maximal expansion of  $\delta\mathbf{r}(t)$  per time step until the average has converged to some fixed value.

The training Lyapunov exponent has previously been derived in the limit of large  $N$  using mean-field theory [7]. It was assumed that the reservoir dimension  $N$  is sufficiently large so that the sum  $\sum_{j=1}^N A_{ij}r_j(t)$  is distributed according to a normal distribution due to the central limit theorem. We employ the same assumption and derive a similar result for the training Lyapunov exponent using random matrix theory. We do not assume that the input is Gaussian random noise, but that it is a general, stationary time series with a rapid decay of time correlations. We therefore do not require an i.i.d. input series. Using these assumptions, we obtain an expression for the training Lyapunov exponent (see [appendix](#)):

$$\lambda_T = \frac{1}{2} \left[ \ln(sN\sigma_A^2) + \ln \left( N^{-1} \sum_i \langle D_{ii}^2(t) \rangle \right) \right]. \tag{7}$$

Here,  $\langle \cdot \rangle$  is the average taken over input samples and ensembles of  $\mathbf{A}$  and  $\mathbf{W}^{(in)}$ . This is the same result as [7], for relaxed assumptions on the input time series. To obtain  $\langle D_{ii}^2 \rangle$ , we use the same procedure as [7] and construct an iterative map for the variance of the reservoir states  $r_i(t)$ . Assuming that  $N$  is large enough so that the sum  $\sum_{j=1}^N A_{ij}r_j$  is normally distributed, we can compute the probability density function  $f_b(x)$  of the local field by using the convolution of the probability mass function of a normal distribution with zero mean and variance  $sN\sigma_A^2\sigma_r^2$ , and the empirical probability mass function of the normalized input time series, given an ensemble of input trajectories initialized with random initial values, scaled by  $\sigma_{in}^2$ , to construct an iterative map of the variance of  $r_i(t)$  taken over input samples and ensembles of  $\mathbf{A}$  and  $\mathbf{W}^{(in)}$ ,

$$\sigma_r^2(t+1) = \int_{-\infty}^{\infty} db (g(b))^2 f_b(b; sN\sigma_A^2, \sigma_r^2(t), \sigma_{in}^2). \tag{8}$$

In [7], it was shown that this map converges to a fixed point when the input is a Gaussian random variable. A similar result was derived by Poole *et al* [19] for feed-forward neural networks, where the map was also shown to rapidly converge. Our numerical results show that  $\sigma_r^2(t)$  converges for non-Gaussian inputs. Assuming  $t$  is large enough for the map to have converged, and denoting the converged variance by  $(\sigma_r^*)^2$ , one finds

$$\langle D_{ii}^2 \rangle = \langle (1 - r_i^2(t))^2 \rangle = 1 - 2(\sigma_r^*)^2 + \langle r_i^4 \rangle, \tag{9}$$

where the fourth moment of  $r_i(t)$ , which also converges as the distribution only depends the first and second moments, can be computed as

$$\langle r_i^4 \rangle = \int_{-\infty}^{\infty} db (g(b))^4 f_b(b; sN\sigma_A^2, (\sigma_r^*)^2, \sigma_{in}^2). \tag{10}$$

Combining (7) and (9), we find that the predicted training Lyapunov exponent agrees very well with the result obtained using the QR method when the reservoir dimension  $N$  is large. The result shows that  $\lambda_T$ , for a given input time series, depends on  $sN\sigma_A^2$  and  $n\sigma_{in}^2$ . This agrees with the discussion in section 2.3.

### 4. Method

To evaluate the prediction performance of ESNs when full and partial information is provided, we use the ESN to predict a chaotic time series where we either input the ESN with the time series of all the components of the time series, or only a single component. In the latter case, we use the ESN to predict the input component. As the ESN has incomplete information for this case, it must construct a time-delay embedding to reproduce the dynamics correctly. As examples of chaotic time series, we use the Lorenz63 system [20], given by

$$\frac{d}{dt}x = \sigma(y - x), \tag{11a}$$

$$\frac{d}{dt}y = \rho x - y - xz, \tag{11b}$$

$$\frac{d}{dt}z = xy - \beta z, \tag{11c}$$

with  $\sigma = 10$ ,  $\rho = 28$ , and  $\beta = 8/3$ , which results in that the dynamical system has a Lyapunov spectrum of  $\lambda_1 = 0.901$ ,  $\lambda_2 = 0$ , and  $\lambda_3 = -14.6$  [21], and the Halvorsen system [21]

$$\frac{d}{dt}x = -ax - 4(y + z) - y^2 \tag{12a}$$

$$\frac{d}{dt}y = -ay - 4(x + z) - z^2 \tag{12b}$$

$$\frac{d}{dt}z = -az - 4(x + y) - x^2, \tag{12c}$$

with  $a = 1.3$ . The Lyapunov spectrum of the Halvorsen system is  $\lambda_1 = 0.69$ ,  $\lambda_2 = 0$ , and  $\lambda_3 = -4.9$  when the considered parameters are used [21].

We obtain a time series by discretizing the dynamical systems (11) and (12) with a sampling rate  $\delta t = 0.1$ . This choice is informed by the work of Kantz and Schreiber (see p 151 in [14]) where the information theoretical concept of mutual information is used to find an optimal step size for time delay embedding of the Lorenz63 system. We use the same sampling rate for the Halvorsen time series. The effect of changing the sampling rate is investigated in section 5.2. The ESN is trained on the Lorenz63 or Halvorsen system for roughly 200 Lyapunov times, where one Lyapunov time is defined as  $\lambda_1^{-1}$  and  $\lambda_1$  is the maximal Lyapunov exponent of the dynamical system. Before feeding the time series to the reservoir, the time series is normalized such that the largest variance of any variable of the dynamical system over time equals unity. This is to ensure that the dependence on  $n\sigma_{in}^2$  is comparable for the different time series.

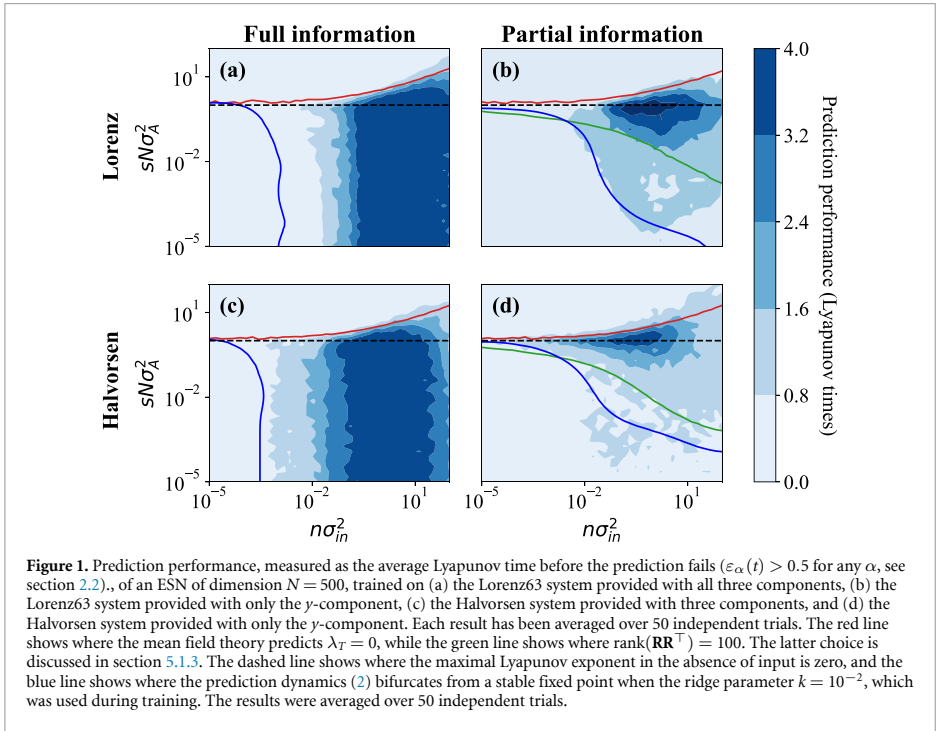
### 5. Results and discussion

#### 5.1. Parameter dependence for full and partial information

We characterize the prediction performance in a phase diagram with axes  $sN\sigma_A^2$  and  $n\sigma_{in}^2$  (see figure 1), for two cases: (i) Providing full phase space information to the reservoir (panels (a) and (c) in figure 1) and (ii) providing only partial phase space information to the reservoir (panels (b) and (d) in figure 1). Different aspects of the phase diagram in figure 1 are discussed below.

##### 5.1.1. Maximal Lyapunov exponent

We first observe that the reservoir dynamics must contract ( $\lambda_T < 0$ ) for successful prediction. This is demonstrated by the red line in the phase diagrams. In [22], the transition between the successful and failed prediction is shown to be smooth. However, we find that the transition becomes sharper as  $N$  increases. We also note that the maximal Lyapunov exponent computed in the absence of input (dashed black line in figure 1), used in [6], works well as long as  $n\sigma_{in}^2$  is small. As  $n\sigma_{in}^2$  becomes larger, the input variance has an increasingly contractive effect on  $\lambda_T$ . It is clear from figure 1 that  $\lambda_T < 0$  is a necessary but not sufficient condition for successful prediction.

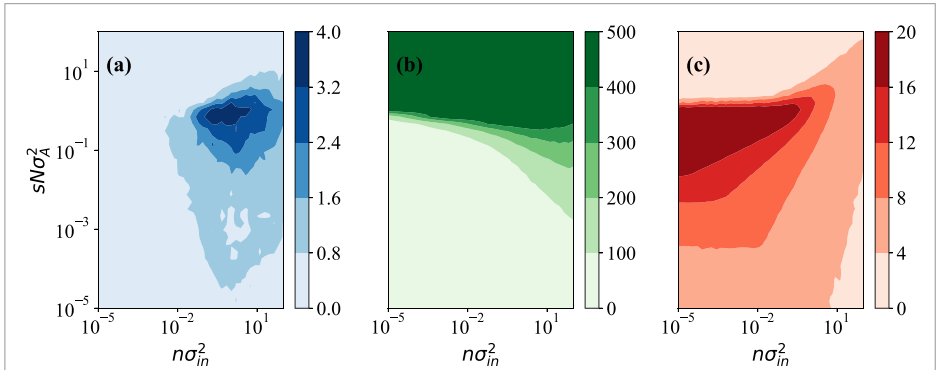


### 5.1.2. Full and partial information

A qualitative difference exists in the parameter dependence on prediction performance when full or partial information is provided to the network. In the full information case, as long as  $\lambda_T < 0$ , the performance is roughly independent of  $sN\sigma_A^2$ . This is consistent with the result of [11, 12], where it was shown that the connections between the reservoir neurons can be removed (setting  $\mathbf{A}$  to zero) and still the reservoir allows successful prediction. Removing the connections renders the ESN memory-less, and the algorithm simply projects the input series nonlinearly to a high dimensional space and performs a function fitting. This is possible because full phase space information is provided; only the current phase space coordinate is necessary to determine the evolution of the dynamics. This is not the case for partial information. In [9], it was shown that the reservoir computer employs time delay embedding to predict a time series. It is possible, according to Takens' embedding theorem, to embed a high dimensional time series using the history of a single observable. The theorem states that, given at least  $2d_f + 1$  delays, where  $d_f$  is the box-counting dimension of the attractor of the time series, the embedding is possible. In our case, this corresponds to having at least  $2d_f + 1$  neurons representing different time scales of the input time series. The box-counting dimension of the Lorenz63 system is 2.06 [21], implying that approximately five neurons are required. However, as was pointed out in [9], while the embedding is possible, projecting the embedding back to the original space *linearly* (2b) is not necessarily accurate. To resolve this, the universal approximation theorem was evoked in [9], stating that with a sufficiently large sum of weighted nonlinear activation functions, any functional relationship can be approximated. Hence, we need sufficiently many neurons representing different time scales of the input time series to be able to predict the time series when only partial information is provided. The different time scales are sampled by choosing reservoir and input weights such that the dependence on previous inputs decays at different rates for different neurons.

### 5.1.3. Rank of $\mathbf{R}\mathbf{R}^\top$

In panels (b) and (d) in figure 1, the ESN must use time-delay embedding to reconstruct the input dynamics. When  $sN\sigma_A^2\sigma_r^2 \ll n\sigma_m^2$ , all reservoir states are highly correlated because they are all strongly driven by the input signal. As  $sN\sigma_A^2\sigma_r^2 \sim n\sigma_m^2$ , the reservoir states may develop different dynamics due to the randomly sampled connections in  $\mathbf{A}$ . This can be quantified using the rank of the matrix  $\mathbf{R}\mathbf{R}^\top$ , i.e. the number of linearly independent (over time) reservoir neurons. We remind the reader that  $\mathbf{R}$  is the matrix whose columns are the reservoir states  $\mathbf{r}(t)$  throughout the training sequence (see section 2.2). The rank of  $\mathbf{R}\mathbf{R}^\top$



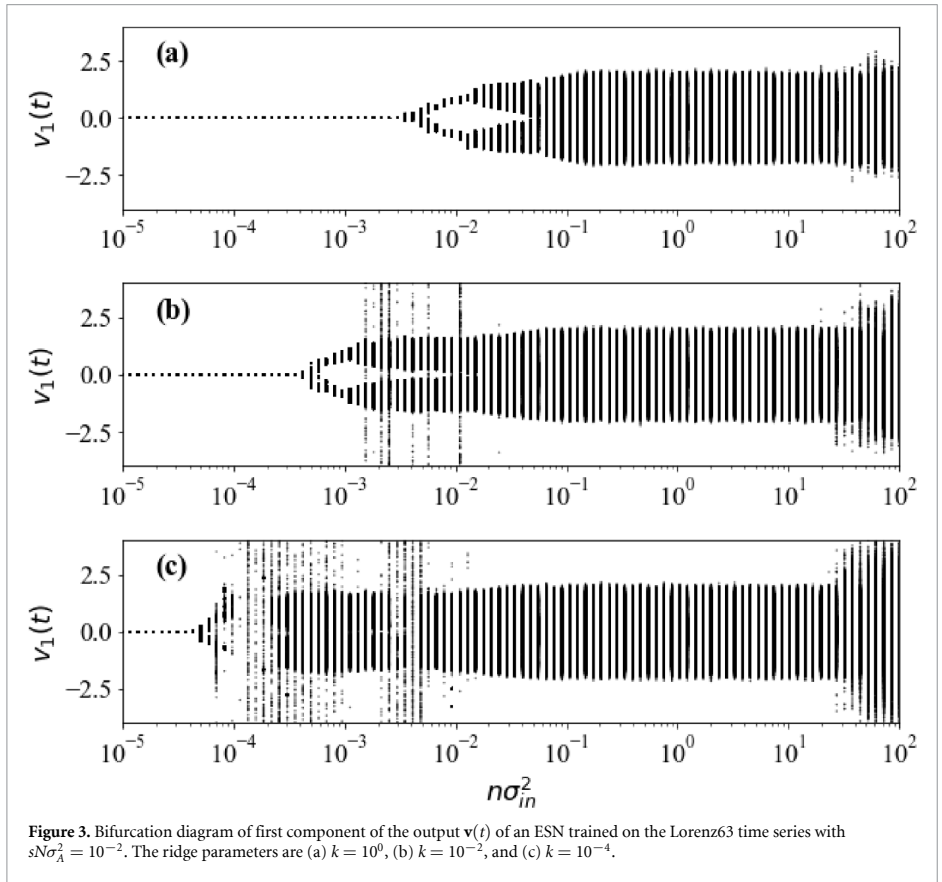
**Figure 2.** (a) Performance of an ESN with dimension  $N = 500$  predicting the  $y$ -component of the Lorenz63 attractor (same data as in figures 1(a) and (b)) rank of the  $\mathbf{R}\mathbf{R}^T$  matrix computed for the same reservoir computer, and (c) the memory capacity, computed for the same reservoir dimension. The result has been averaged over 100 independent trials.

quantifies the ‘richness’ described by Jaeger in his original paper on ESNs. This is the effective number of activation functions that the ESN can use to approximate the functional relationship between the reservoir embedding and the original space. In figure 1, the green line shows where the rank is equal to 100. Along this contour, the ESN can effectively employ 100 reservoir states to approximate the functional relationship between the time-delay embedding performed by the reservoir and the output. Above the green line, the rank increases gradually, making the approximation more accurate. As shown in figure 1, it is only once the rank begins to increase that the reservoir is able to predict. The gradual increase of rank is reflected in a gradual increase of performance. In panels (a) and (c), the rank of  $\mathbf{R}\mathbf{R}^T$  does not affect performance, because the ESN does not need to perform a time-delay embedding to reconstruct the input dynamics.

That predictive performance depends on the rank of  $\mathbf{R}\mathbf{R}^T$  has several consequences. Firstly, the lower bound depends on the effective number of reservoir states required to approximate the relation between the reservoir embedding and the original space, and is independent of any time scale of the predicted time series. Thus, it is incorrect to state that the scale of  $\mathbf{A}$  (often the spectral radius is used) must be adjusted in accordance with the time scale of the predicted time series [5]. In fact, as long as sufficiently many neurons are uncorrelated and each neuron is an echo of the input, prediction is possible. Secondly, the result has consequences for the linear memory capacity of a reservoir [5]. The memory capacity  $MC$  measures the maximal achievable linear correlation between current reservoir states and previous inputs and is defined as

$$MC = \sum_{\tau=1}^{\infty} \max_{\mathbf{w}^{(out)}} \frac{\text{cov}^2(\mathbf{v}(t), \mathbf{u}(t - \tau))}{\sigma_v^2 \sigma_u^2}, \tag{13}$$

where the input is a series of i.i.d. Gaussian random variables. A high memory capacity means that the reservoir state  $\mathbf{r}(t)$  contains linear information about an input  $\mathbf{u}(t - \tau)$  for some large  $\tau$ . Hence, all reservoir states between  $t - \tau$  and  $t$  should be highly correlated. The rank of  $\mathbf{R}\mathbf{R}^T$  is equal to its number of non-zero singular values. This is equivalent to the number of non-zero singular values of  $\mathbf{R}^T \mathbf{R}$ , which represents the correlations between reservoir states at different times. Since a high rank reflects that the reservoir effectively has a large number of reservoir states to use in its functional approximation, and a low rank reflects a high memory capacity, maximizing linear memory capacity and functional approximation accuracy appear to be mutually exclusive tasks. This is related to the well-known memory-nonlinearity trade-off [23]; the more nonlinear the reservoir dynamics are, the shorter the memory becomes. This prediction is verified by figure 2. Comparing panels (b) and (c), we see that when the memory capacity peaks, the rank is low. Comparing panels (a) and (c), we conclude that high linear memory capacity is not indicative of high prediction performance. This means that prediction performance does not rely on being able to reconstruct the time series far back in time, but rather on the ability to represent several time scales of the input. Two important points should be made: Firstly, the defined memory capacity only measures linear information, and so the result does not imply that the reservoir does not need memory to perform a prediction. Indeed, when only partial information is presented to the network, memory is necessary to construct a time-delay embedding. When linear memory capacity is low, the reservoir can still retain nonlinear information about the input. In [24], information processing capacity was introduced as a metric that extends memory capacity to nonlinear cases. However, as shown in [23], nonlinearity inherently degrades memory of the input, and



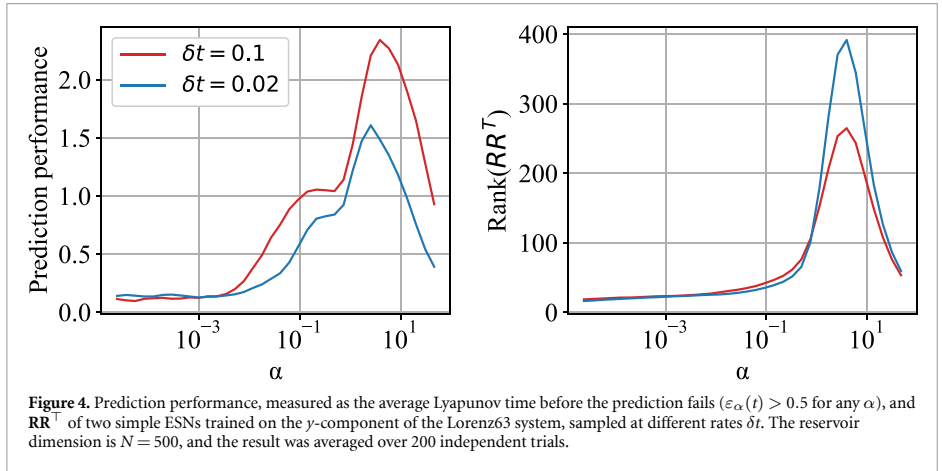
thus, long memory, which is only afforded by linear dynamics, and ‘rich’ reservoir dynamics, cannot be achieved simultaneously. Secondly, memory capacity is computed using an i.i.d. input, meaning there are no time correlations in the input series. In general, time correlations exist for input series, and so each input carries with it information about previous inputs. This can affect the amount of linear correlation the current reservoir state has with previous inputs, and so panel (c) cannot be used to directly infer the linear memory of the reservoir in panel (a). However, we expect the parameter regions with high correlation with previous inputs to be similar for the case of inputs with time correlations.

5.1.4. Saturation of activation function

The performance drops once  $n\sigma_{in}^2$  becomes too large. In this limit, the local fields of the reservoir neurons become so large that the activation function saturates and information about the input time series is lost.

5.1.5. Ridge parameter

When  $n\sigma_{in}^2$  is small, prediction fails the full information case (see panels (a) and (c) in figure 1). To see what causes this, consider that in order for the ESN to predict a time series, it must be able to reproduce the Lyapunov spectrum of the input time series [1]. This means that the norm of the matrix  $\mathbf{A} + \mathbf{W}^{(in)}\mathbf{W}^{(out)}$  relevant for the prediction dynamics (2), must be sufficiently large. However, the ridge parameter  $k$  sets a limit for how large the norm of  $\mathbf{W}^{(out)}$  can be. Consider, for example, a chaotic time series. To predict the chaotic time series,  $n\sigma_{in}^2$  must exceed a threshold value so that the prediction dynamics can be chaotic. The same line of arguments hold for the case when partial information is provided (panels (b) and (d)). To observe the effect of changing the ridge parameter, we compute a bifurcation diagram of the reservoir neurons in an ESN trained on the Lorenz63 system. In figure 3, we see how the ridge parameter changes at what value of  $n\sigma_{in}^2$  the prediction dynamics bifurcates from having a stable fixed point at zero. Beyond this bifurcation, the prediction dynamics eventually becomes that of the Lorenz63 system. For smaller ridge



parameters, the dynamics is more prone to become unstable. Indeed, the effect of the ridge parameter is to regularize  $\mathbf{W}^{(out)}$  such that its entries do not diverge to infinity due to  $\mathbf{RR}^T$  having an undefined inverse (see (3)). Thus, this instability is expected as  $k$  decreases. The bifurcation is shown in figure 1 as a blue line and corresponds to the second panel in figure 3. In figure 1, the contour where the bifurcation occurs looks different for the full and partial information case because, for the case when only partial information is provided, the reservoir fails to embed the input dynamics and the prediction dynamics does not become chaotic.

**5.2. Independence of  $\delta t$**

To study the dependence on changing  $\delta t$ , we employ the ‘simple ESN’ architecture [25], where  $\mathbf{A}$  is a diagonal matrix. This is done because it allows us to control the time scale of the reservoir neurons explicitly. In the result below, we deterministically set the diagonal elements of  $\mathbf{A}$  to  $A_{ii} = \alpha \frac{i}{N}$  for a positive parameter  $\alpha$ . The time scale of each neuron is simply determined by the magnitude of its corresponding weight in  $\mathbf{A}$ . If the ESN depends on  $\delta t$ , and by extension, the memory requirements of the time series to be predicted, the parameter region where prediction works should change when the sampling rate  $\delta t$  is changed. As seen in figure 4, apart from decreasing the performance, decreasing  $\delta t$  does not shift the parameter region where prediction works significantly, despite being altered by one order of magnitude. This is consistent with the previous observation, that the performance depends on the number of uncorrelated reservoir states, as measured by the rank of  $\mathbf{RR}^T$ . What changes is instead the prediction performance. This is consistent with the result from [14], where  $\delta t = 0.1$  is closer to the optimal sampling rate for time delay embedding of the Lorenz63 system. We note that the rank is larger when  $\delta t$  is smaller.

**6. Conclusions**

Correctly selecting tuning parameters is crucial for successful reservoir computing. However, no clear understanding of how the parameters should be selected exists, and the choice largely comes down to heuristics. In this article, we explain how prediction performance depends on parameter selection when full phase space information or partial phase space information is provided to the network.

We find that there is a qualitative difference between the two cases. When partial phase space information is provided, the reservoir must construct a time-delay embedding of the input time series. To approximate the functional relationship between the embedding and the original space of the time series, the reservoir network uses a weighted sum of reservoir states; the more states, the more accurate the approximation. We show that the effective number of available reservoir states used for the approximation is equal to the number of independent states, quantifies by the number of non-zero singular values of the matrix  $\mathbf{RR}^T$ . This imposes a condition on the relationship between the strength of the recurrent connections of the reservoir and the strength of the input signal. If the input signal dominates the dynamics, the reservoir states are strongly correlated, making the approximation of the functional relationship poor. On the other hand, no such condition is found when full phase space information is provided. This is because all the information required to predict the next time step is provided in the current time step. Hence, the reservoir network can simply perform function fitting to model the input time series.

That the approximation of the functional relationship between the reservoir embedding and the original space becomes more accurate, thus improving the network’s prediction performance, when reservoir states become uncorrelated has a consequence for the role of linear memory capacity. As memory capacity increases when the linear correlation between the reservoir states at times  $t$  and  $t - \tau$  increases, maximizing memory capacity and predictive performance are mutually exclusive tasks. Memory capacity should therefore not be used as a metric associated with predictive performance.

Our results also show that tuning the time scale of the reservoir in accordance with the time scale of the input time series is unnecessary. In fact, the lower bound of the reservoir time scale for successful time-series prediction is independent on the sampling rate of the input time series. Instead, it depends on when the reservoir states start to become uncorrelated. However, we find that predictive performance can be improved by tuning the sampling rate in the same way it can be optimized in time-delay embedding literature.

Finally, we find that a lower limit for the strength of the input exists for both the full and partial information case due to that the ridge parameter limits the norm of the output connection strength. Limiting the norm constrains the maximum achievable maximal Lyapunov exponent of the reservoir dynamics during prediction. Hence, if this exponent is smaller than that of the input time series, prediction is impossible.

In conclusion, we have studied the parameter regions where reservoir computing is successful in the case of full and partial information, and found they differ qualitatively. The result is a step in the direction of clarifying how parameters should be selected in an informed way, instead of relying on heuristics. More research is needed to understand how the reservoir can be optimally designed to develop uncorrelated reservoir states to improve predictive performance.

### Data availability statement

The data that supports the findings of this study are available upon reasonable request from the authors.

### Acknowledgment

B M was supported by grants from the Knut and Alice Wallenberg Foundation, Grant No. 2019.0079, and Vetenskapsrådet, Grant No. 2021-4452.

### Ethical statement

This manuscript does not involve any human or animal participants.

### Conflict of interest

All authors declare that they have no conflicts of interest.

### Appendix

The training Lyapunov exponent  $\lambda_T$  is defined as

$$\lambda_T = \lim_{t \rightarrow \infty} \frac{1}{t} \log \frac{|\mathbf{D}(t-1)\mathbf{A}\mathbf{D}(t-2)\mathbf{A} \dots \mathbf{D}(0)\mathbf{A}\delta\mathbf{r}(0)|}{|\delta\mathbf{r}(0)|}, \tag{A1}$$

where  $\mathbf{D}(t)$  is a diagonal matrix with entries

$$D_{ii}(t) = 1 - \tanh^2 \left( \sum_j^N A_{ij}r_j(t) + \sum_\alpha^n W_{i\alpha}^{(in)} u_\alpha(t) \right), \tag{A2}$$

and  $\delta r(t)$  is the separation between two initially infinitesimally nearby reservoir states. To derive (7), we start from (A1) by writing  $\delta\mathbf{r}(0) = \delta r_0 \mathbf{n}$ , where  $\mathbf{n}$  is the unit vector pointing in the direction of  $\delta\mathbf{r}(0)$ , and denote the matrix product as  $\mathbf{J}_t = \mathbf{D}(t-1)\mathbf{A}\mathbf{D}(t-2)\mathbf{A} \dots \mathbf{D}(0)\mathbf{A}$ . Using this, we write (6) as

$$\lambda_T = \lim_{t \rightarrow \infty} \frac{1}{2t} \ln (\mathbf{n}^\top \mathbf{J}_t^\top \mathbf{J}_t \mathbf{n}). \tag{A3}$$



Assuming the decay of correlation between consecutive  $\mathbf{D}(t)\mathbf{A}$  matrices is exponential, and that the distribution of the elements  $D_{ii}(t)$  converge rapidly, we approximate the matrices  $\mathbf{D}(t)\mathbf{A}$  as independent and identically distributed and use the Furstenberg theorem to obtain [26]

$$\lambda_T = \lim_{t \rightarrow \infty} \frac{1}{2t} \langle \ln(\mathbf{n}^\top \mathbf{J}_t^\top \mathbf{J}_t \mathbf{n}) \rangle, \tag{A4}$$

where the average is taken over samples of inputs and ensembles of  $\mathbf{A}$  and  $\mathbf{W}^{(in)}$  matrices. We assume that the average over samples is equal to the time average of the input time series. The theorem states that in the limit of large  $t$ , the Lyapunov exponent is a non-random quantity. If the entries of  $\mathbf{J}_t$  reach a stationary distribution, then the product  $\mathbf{n}^\top \mathbf{J}_t^\top \mathbf{J}_t \mathbf{n}$  has a negligible variance in the limit of large  $N$ . In this limit, one obtains

$$\lambda_T = \lim_{t \rightarrow \infty} \frac{1}{2t} \ln \langle \mathbf{n}^\top \mathbf{J}_t^\top \mathbf{J}_t \mathbf{n} \rangle. \tag{A5}$$

We use the result derived by Newman for products of i.i.d. random matrices [26, 27] to simplify the expression to

$$\lambda_T = \frac{1}{2} \ln \langle \mathbf{n}^\top (\mathbf{D}(t)\mathbf{A})^\top \mathbf{D}(t)\mathbf{A} \mathbf{n} \rangle. \tag{A6}$$

The proof of this equivalence requires the distribution of the random variable  $\frac{|\mathbf{D}(t)\mathbf{A}\mathbf{z}(t)|}{|\mathbf{z}(t)|}$ , where  $\mathbf{z}(t)$  is a random  $N$ -dimensional vector, to be independent on  $\mathbf{z}(t)$ . Using the Euclidian norm, we have

$$\frac{|\mathbf{D}(t)\mathbf{A}\mathbf{z}(t)|^2}{|\mathbf{z}(t)|^2} = \frac{\mathbf{z}^\top(t) \mathbf{A}^\top \mathbf{D}^2(t) \mathbf{A} \mathbf{z}(t)}{\mathbf{z}^\top(t) \mathbf{z}(t)}. \tag{A7}$$

The elements of the matrix  $\mathbf{A}^\top \mathbf{D}^2(t) \mathbf{A}$  are sums of all the diagonal entries of  $\mathbf{D}^2(t)$ , each weighted by the product of two entries of  $\mathbf{A}$ . As the elements of  $\mathbf{A}$  are i.i.d. when  $N$  is large, this sum approaches a mean value that is independent of the direction of  $\mathbf{z}(t)$ . The proof then proceeds by stating that, if the random variable  $\frac{|\mathbf{D}(t)\mathbf{A}\mathbf{z}(t)|}{|\mathbf{z}(t)|}$  is independent of  $\mathbf{z}(t)$ , then

$$\ln |\mathbf{J}_t \mathbf{z}(0)| = \sum_{k=0}^{t-1} \ln \frac{|\mathbf{D}(k)\mathbf{A}\mathbf{z}(k)|}{|\mathbf{z}(k)|} \tag{A8}$$

is a sum of uncorrelated variables. The result in (A6) follows by employing the law of large numbers. Proceeding by using the assumption that the entries of  $\mathbf{D}(t)\mathbf{A}$  are approximately i.i.d. (A6) can be evaluated to be

$$\lambda_T = \frac{1}{2} \ln N^{-1} \langle \text{tr} [(\mathbf{D}(t)\mathbf{A})^\top \mathbf{D}(t)\mathbf{A}] \rangle. \tag{A9}$$

The argument of the logarithm can be rewritten as

$$\begin{aligned} N^{-1} \langle \text{tr} [\mathbf{A}^\top \mathbf{D}^2(t) \mathbf{A}] \rangle &= N^{-1} \sum_i^N \left\langle D_{ii}^2(t) \left( \sum_j^N A_{ij}^2 \right) \right\rangle \\ &= N^{-1} \sum_i^N \langle D_{ii}^2(t) sN\sigma_A^2 \rangle = s\sigma_A^2 \sum_i^N \langle D_{ii}^2(t) \rangle. \end{aligned} \tag{A10}$$

Thus, we finally obtain

$$\lambda_T = \frac{1}{2} \left[ \ln(sN\sigma_A^2) + \ln \left( N^{-1} \sum_i^N \langle D_{ii}^2(t) \rangle \right) \right]. \tag{A11}$$

This result is equivalent to the logarithm of the square root of (10) in [7], derived there for Gaussian white-noise inputs. Our derivation shows that (A11) is valid for general, stationary time series with rapid decay of time correlations.

**ORCID iD**

B Mehlig  <https://orcid.org/0000-0002-3672-6538>

## References

- [1] Pathak J, Lu Z, Hunt B R, Girvan M and Ott E 2017 *Chaos* **27** 121102
- [2] Lim S H, Theo Giorgini L, Moon W and Wettlaufer J S 2020 *Chaos* **30** 123126
- [3] Lu Z, Pathak J, Hunt B, Girvan M, Brockett R and Ott E 2017 *Chaos* **27** 041102
- [4] Kim J Z, Lu Z, Nozari E, Pappas G J and Bassett D S 2021 *Nat. Mach. Intell.* **3** 316–23
- [5] Jaeger H 2001 *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148 p 13
- [6] Verstraeten D, Schrauwen B, d'Haene M and Stroobandt D 2007 *Neural Netw.* **20** 391–403
- [7] Massar M and Massar S 2013 *Phys. Rev. E* **87** 042809
- [8] Wainrib G and Galtier M N 2016 *Neural Netw.* **76** 39–45
- [9] Hart A, Hook J and Dawes J 2020 *Neural Netw.* **128** 234–47
- [10] Ozturk M C, Xu D and Principe J C 2007 *Neural Comput.* **19** 111–38
- [11] Pyle R, Jovanovic N, Subramanian D, Palem K V and Patel A B 2021 *Phil. Trans. R. Soc. A* **379** 20200246
- [12] Griffith A 2021 *Essential reservoir computing PhD Thesis* The Ohio State University
- [13] Lukoševičius M and Jaeger H 2009 *Comput. Sci. Rev.* **3** 127–49
- [14] Kantz H and Schreiber T 2004 *Nonlinear Time Series Analysis* vol 7 (Cambridge: Cambridge University Press)
- [15] Mehlig B 2021 *Machine Learning with Neural Networks: An Introduction for Scientists and Engineers* (Cambridge: Cambridge University Press) (<https://doi.org/10.1017/9781108860604>)
- [16] Tikhonov A N et al 1977 *Solutions of Ill-Posed Problems* (New York: V.H. Winston)
- [17] Ott E 2002 *Chaos in Dynamical Systems* (Cambridge: Cambridge University Press) (<https://doi.org/10.1017/CBO9780511803260>)
- [18] Geist K, Parlitz U and Lauterborn W 1990 *Prog. Theor. Phys.* **83** 875–93
- [19] Poole B, Lahiri S, Raghu M, Sohl-Dickstein J and Ganguli S 2016 *Adv. Neural Inf. Process. Syst.* **29** 3369–77
- [20] Lorenz E N 1963 *J. Atmos. Sci.* **20** 130–41
- [21] Sprott J C 2010 *Elegant Chaos: Algebraically Simple Chaotic Flows* (Singapore: World Scientific) (<https://doi.org/10.1142/7183>)
- [22] Schrauwen B, Buesing L and Legenstein R 2008 *Adv. Neural Inf. Process. Syst.* **21** 1425–32
- [23] Inubushi M and Yoshimura K 2017 *Sci. Rep.* **7** 1–10
- [24] Dambre J, Verstraeten D, Schrauwen B and Massar S 2012 *Sci. Rep.* **2** 1–7
- [25] Fette G and Eggert J 2005 Short term memory and pattern matching with simple echo state networks *Int. Conf. on Artificial Neural Networks* (Springer) pp 13–18
- [26] Crisanti A, Paladin G and Vulpiani A 1993 *Products of Random Matrices: In Statistical Physics* vol 104 (Berlin: Springer)
- [27] Newman C M 1986 *Commun. Math. Phys.* **103** 121–6