Supervised machine learning to estimate instabilities in chaotic systems: estimation of local Lyapunov exponents

Received: 7 February 2022 | Revised: 20 February 2023 | Accepted: 21 March 2023

DOI: 10.1002/qj.4450

Quarterly Journal of the Royal Meteorological Society RMetS

RESEARCH ARTICLE

Supervised machine learning to estimate instabilities in chaotic systems: Estimation of local Lyapunov exponents

Daniel Ayers[1,2] | **Jack Lau**[3] | **Javier Amezcua**[1,4] | **Alberto Carrassi**[1,5] | **Varun Ojha**[3,6]

[1]Department of Meteorology, University of Reading, Reading, UK

[2]National Centre for Earth Observation, Reading, UK

[3]Department of Computer Science, University of Reading, Reading, UK

[4]School of Science and Engineering, Tecnológico de Monterrey, Mexico City, Mexico

[5]Department of Physics and Astronomy "Augusto Righi", University of Bologna, Bologna, Italy

[6]School of Computing, Newcastle University, Newcastle upon Tyne, UK

Correspondence
Daniel Ayers, Department of Meteorology, University of Reading, Reading, UK.
Email: d.ayers@pgr.reading.ac.uk

Funding information
Engineering and Physical Sciences Research Council, Grant/Award Number: EP/N509723/1; National Centre for Earth Observation, Grant/Award Number: NCEO02004; Schmidt Futures, Grant/Award Number: 353

Abstract

In chaotic dynamical systems such as the weather, prediction errors grow faster in some situations than in others. Real-time knowledge about the error growth could enable strategies to adjust the modelling and forecasting infrastructure on the fly to increase accuracy and/or reduce computation time. For example, one could change the ensemble size, the distribution and type of target observations, and so forth. Local Lyapunov exponents are known indicators of the rate at which very small prediction errors grow over a finite time interval. However, their computation is very expensive: it requires maintaining and evolving a tangent linear model, orthogonalisation algorithms and storing large matrices. In this feasibility study, we investigate the accuracy of supervised machine learning in estimating the current local Lyapunov exponents, from input of current and recent time steps of the system trajectory, as an alternative to the classical method. Thus machine learning is not used here to emulate a physical model or some of its components, but "nonintrusively" as a complementary tool. We test four popular supervised learning algorithms: regression trees, multilayer perceptrons, convolutional neural networks, and long short-term memory networks. Experiments are conducted on two low-dimensional chaotic systems of ordinary differential equations, the Rössler and Lorenz 63 models. We find that on average the machine learning algorithms predict the stable local Lyapunov exponent accurately, the unstable exponent reasonably accurately, and the neutral exponent only somewhat accurately. We show that greater prediction accuracy is associated with local homogeneity of the local Lyapunov exponents on the system attractor. Importantly, the situations in which (forecast) errors grow fastest are not necessarily the same as those in which it is more difficult to predict local Lyapunov exponents with machine learning.

KEYWORDS

chaos, local Lyapunov exponents, numerical modelling, supervised machine learning

1 | INTRODUCTION

Weather and climate are well-known exemplars of chaotic dynamical systems. These systems exhibit extreme sensitivity to initial conditions, meaning that initial condition errors are subject to (on average) exponential growth until they reach saturation (Lorenz, 1963; Kalnay, 2002). The rate and characteristics of such growth, however, are highly state dependent (Lighthill *et al.*, 1986; Vannitsem, 2017). As a consequence, although chaotic systems have a finite predictability horizon (about two weeks for the atmosphere: see, e.g., Holton and Hakim, 2013), the best estimate of prediction-error growth fluctuates in size along with the system's evolution, as the system goes through periods of lower or higher predictability. For example, the short-term predictability of the atmosphere depends on the weather regime present at a given time (see, *e.g.* Palmer, 1996). Understanding the nature of error growth is essential to characterising a system, and to enable better prediction. The present work is motivated by the idea that, if the degree of predictability of the system is known in real time, it may be possible and beneficial to take adaptive measures. For instance, we speculate that a local decrease of predictability might be counteracted by increasing the ensemble size in the context of ensemble-based data assimilation or probabilistic forecasting, or the distribution and type of target observations. Conversely, in areas of high predictability, one might save computational resources (and thus energy consumption) via the opposite actions. Understanding the impact of such actions would require experimentation. In this study we investigate the potential of machine learning (ML) methods (Bishop, 1995; Hastie *et al.*, 2009) to provide a real-time estimation of the system's local predictability.

The mathematical theory of dynamical systems has long been the backbone to understanding and quantifying predictability in deterministic chaotic systems. This is commonly done by studying the instability properties of the solution, that is, by analysing the linearised dynamics of small perturbations: the tangent space evolution of these "small" perturbations is taken as proxy of the dynamics of unknown initial condition errors (Ott, 2002). In this context, Lyapunov exponents (LEs) are well-established quantities that measure the asymptotic rates of error growth for a set of infinitesimally small errors that capture all directions of phase space (Pikovsky and Politi, 2016). In practice, they measure the average growth of small finite errors over long periods of time. The spectrum of LEs is characteristic of each given dynamical system. Lyapunov exponents, and their corresponding Lyapunov vectors (LVs), have been exploited in geosciences for more efficient uncertainty quantification in data assimilation (*e.g.*, Palatella *et al.*, 2013; Quinn *et al.*, 2020; Albarakati *et al.*, 2021; Carrassi *et al.*, 2022), or for initialising probabilistic predictions (*e.g.*, Toth and Kalnay, 1997; Buizza, 2019; Vannitsem and Duan, 2020). The LE spectrum can also be used to calculate other characteristic properties of a system, such as the Kolmogorov–Sinai entropy, which measures the rate of information loss (Sinai, 2009), or the Kaplan–Yorke attractor dimension (Kaplan and Yorke, 1979).

We note that LEs are associated with directions known as covariant Lyapunov vectors (CLVs). CLVs also provide useful information and can be calculated numerically (see Ginelli *et al.*, 2007; Wolfe and Samelson, 2007; Froyland *et al.*, 2013). However, in this work we focus on the exponents only.

The LEs are calculated as an average of finite-time Lyapunov exponents, which are here referred to as *local Lyapunov exponents* (LLEs: (Benettin *et al.*, 1980a; Benettin *et al.*, 1980b; Kuptsov and Parlitz, 2012). Whereas LEs provide "global" information about the average growth of small perturbations in the system, the LLEs describe "local" growth rates along a finite-time section of the trajectory. Notably, the LLEs show the heterogeneity of the instabilities in phase space: the fluctuation of the local dynamical stability around the asymptotic value as the system state varies (Sandri, 1996; Pikovsky and Politi, 2016). This makes the LLEs ideal quantities to measure the local degree of predictability, yet a bottleneck for their real-time use in operational scenarios is the huge computational cost. Computational cost grows quickly with the system's dimension, making it prohibitive even for moderate-size models, let alone for models as large as those currently used in numerical weather predictions ($\mathcal{O}(10^9)$ dimensions). Using the standard method (Benettin *et al.*, 1980a; Benettin *et al.*, 1980b; Kuptsov and Parlitz, 2012; Pikovsky and Politi, 2016), calculating the LLEs and LEs involves computing a long trajectory of the system (including a spin-up needed to ensure the solution has reached the model attractor), propagating perturbations (as many as the number of desired LLEs) with the tangent linear model (i.e., the resolvent of the model Jacobian), and then repeatedly performing a process of orthogonalisation (e.g., using a QR decomposition algorithm).

Despite the computational bottleneck, Lyapunov methods (i.e., computing the local and global LE spectrum and aforementioned associated properties or the Lyapunov vectors) have been used for dynamical analysis of geophysical models of intermediate order ($\mathcal{O}(10^3)$ – $\mathcal{O}(10^5)$ variables): for example, see Vannitsem and Lucarini (2016), Vannitsem (2017), and De Cruz *et al.* (2018). Additionally, Lyapunov methods have been applied to weather reanalysis data to analyse the dynamics of the North Atlantic Oscillation (Quinn *et al.*, 2021) and of persistent states of atmospheric pressure over the European and western Asian continents (Quinn *et al.*, 2022). In

these works, the bottleneck was overcome by reducing the data dimension (using empirical orthogonal functions) and constructing a reduced model. Whilst these works demonstrate the utility of Lyapunov methods, they do not provide a means of calculating LLEs that is cheap enough to be carried out regularly during a forecasting cycle.

Avoiding the need for such a costly computation whilst attaining an estimate of the LEs or LLEs thus has great relevance. In their recent work, Chen *et al.* (2021) show how the outcomes of properly tuned data assimilation experiments can reveal the first LE as well as the Kolmogorov–Sinai entropy of the underlying dynamical model. The present work also seeks to avoid the cost of classical calculation methods, albeit only when time is critical. We investigate the feasibility of using certain ML methods (Bishop, 1995; Hastie *et al.*, 2009) to estimate the LLEs based only on information from the system's solution. Our focus is on supervised learning, which uses a data set of input–output pairs. The targets, that is, the desired outputs, are LLEs calculated using the classical method of evolving perturbations via the tangent linear model and orthogonalising. In this way, the cost of such methods is paid during the training phase of the ML method, and is avoided when making predictions.

In the area of weather and climate forecasting, supervised learning has been used for various purposes (see e.g., Reichstein *et al.*, 2019; Rasp *et al.*, 2020; Chantry *et al.*, 2021; Düben *et al.*, 2021, and references therein). These include (i) emulating the full dynamics of a system (Pathak *et al.*, 2017; Fablet *et al.*, 2018; Pathak *et al.*, 2018; Nguyen *et al.*, 2019; Brajard *et al.*, 2020; Patel *et al.*, 2021; Schultz *et al.*, 2021; Sonnewald *et al.*, 2021) and (ii) improving a physics-based model with data-driven correction or parameterisation (O'Gorman and Dwyer, 2018; Rasp *et al.*, 2018; Bolton and Zanna, 2019; Bonavita and Laloyaux, 2020; Rasp, 2020; Brajard *et al.*, 2021; Gottwald and Reich, 2021; Nguyen *et al.*, 2021). Both approaches imply an intervention in the original model: the first approach yields surrogate data-driven models of the full original system, while the second approach builds hybrid models with data-driven and physics-based components. In either case the spectrum of the LEs of these new models can be computed using the standard approach (Benettin *et al.*, 1980a; Benettin *et al.*, 1980b; Kuptsov and Parlitz, 2012) and can be compared with that of the original model as a way to quantify the goodness of the ML-reconstructed dynamics (Pathak *et al.*, 2017; Brajard *et al.*, 2020).

In contrast to these two families of methods, this study aims towards improving prediction skills by equipping the model with an external tool to quantify the local degree of predictability in real time, and thus guide "nonintrusive" adaptations, whereby the model equations are left unaltered. More specifically, the goal is to use ML to predict the current LLE spectrum given the input of the system state at the current and (possibly) most recent time steps. We envisage that the trained ML algorithm could then be interrogated for information about the local dynamical instability whilst performing the numerical model forward integration. We speculate that such information could drive a decision process for adaptive modelling: for example, adjusting the ensemble size when performing ensemble-based data assimilation or probabilistic predictions, changing the distribution and type of target observations, or adapting the numerical integration scheme. Such adaptations could mitigate error, improve uncertainty quantification, or reduce computational cost.

In this feasibility study, we test the accuracy of some popular supervised ML algorithms in this task in two prototypical low-dimensional chaotic dynamical systems. This study is concerned solely with the predictive capability of ML methods: the task of optimising the computational cost of making predictions is left for future work. We anticipate that the latter task will be largely dependent on the specific use case and computing hardware. The ML algorithms we test are regression trees (RTs: Breiman *et al.*, 1984), multilayer perceptrons (MLPs: *e.g.* see Goodfellow *et al.*, 2016, Chapter 6), convolutional neural networks (CNNs: LeCun *et al.*, 1989), and long short-term memory networks (LSTMs: Hochreiter and Schmidhuber, 1997; Graves, 2012). These algorithms encompass three approaches to exploiting the temporal structure of the input. We evaluate both their pointwise accuracy and their statistical performance, measured in this case by the closeness of the distribution of predictions to the distribution of the target values. We find that on average the machine learning algorithms predict the stable local Lyapunov exponent accurately, the unstable exponent reasonably accurately, and the neutral exponent only somewhat accurately. Each exponent is predicted more accurately in the Lorenz 63 system than in the Rössler system . We show that greater prediction accuracy is associated with local homogeneity of the local Lyapunov exponents on the system attractor. Importantly, the situations in which (forecast) errors grow fastest are not necessarily the same as those in which it is more difficult to predict local Lyapunov exponents with machine learning.

The rest of this article is organised as follows. In Section 2, we briefly review the theory of LEs and detail the method used to compute them. In Section 3, we pose and conceptualise the ML problem we intend to solve, motivate the choice of algorithms, and detail the input and target data and the evaluation metrics. In Section 4, we present the two systems under consideration: the Rössler and Lorenz 63 models, and discuss the characteristics of their Lyapunov spectra. In Section 5 we present the results and Section 6 concludes with a discussion.

2 | LYAPUNOV EXPONENTS

2.1 | Overview of the theory

We review briefly the theory of LEs, with the aim of providing an intuitive explanation of what they are. The section follows Legras and Vautard (1996), Benettin *et al.* (1980a); Benettin *et al.* (1980b), Kuptsov and Parlitz (2012), Pikovsky and Politi (2016) and Strogatz, 2018, Chapter 9, Section 3, to which we refer the reader for a more rigorous and comprehensive treatment.

Consider a deterministic autonomous dynamical system

$$\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state of the system, $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^n$ is the evolution function, and $\dot{\mathbf{x}}$ denotes the derivative of $\mathbf{x}$ with respect to time. A trajectory of the dynamical system starting at initial condition $\mathbf{x}(0)$ is a set $\{\mathbf{x}(t) : t \in A\}$, where A is a connected subset of $\mathbb{R}_{\geq 0}$ containing 0. Consider the difference $\mathbf{v}(t)$ between a trajectory started from the "true" initial condition $\mathbf{x}(0)$ and a trajectory started from the perturbed initial condition $\mathbf{x}(0) + \mathbf{v}(0)$, where $\mathbf{v}(0)$ is infinitesimally small. The idea behind LEs is to find $\lambda(t)$, where

$$e^{t\lambda(t)} = \frac{\|\mathbf{v}\,(t)\|}{\|\mathbf{v}\,(0)\|} \tag{2a}$$

$$\Leftrightarrow \lambda(t) = t^{-1} \ln\left(\frac{\|\mathbf{v}\,(t)\|}{\|\mathbf{v}\,(0)\|}\right). \tag{2b}$$

In other words, $\lambda(t)$ is the exponential growth rate of the initial error. In this setting, $\lambda(t)$ is specific to the initial condition $\mathbf{x}(0)$ and perturbation $\mathbf{v}(0)$, and $\lambda(t)$ varies with time.

Lyapunov exponents generalise this notion to describe (a) average exponential growth rates of the system, regardless of the initial condition and (b) the exponential growth rates for infinitesimal perturbations in all directions. To account for all directions, we consider perturbations contained in an n-sphere of infinitesimal radius. As time progresses, perturbations within the sphere are mapped into an ellipsoid. The LEs are the time average of the exponential growth rate of the ratios between the axes of the sphere and ellipsoid (Legras and Vautard, 1996).

Fix an initial condition $\mathbf{x}(0)$ and let $\mathbf{v}(0)$ be an infinitesimally small perturbation, as above. Then the dynamics of the perturbation are given by

$$\dot{\mathbf{v}} = \mathbf{J}_\mathbf{g}\,\mathbf{v}, \tag{3}$$

where $\mathbf{J}_\mathbf{g}$ is the Jacobian of $\mathbf{g}$ evaluated at $\mathbf{x}(t)$, that is, the linearisation of the evolution function at $\mathbf{x}(t)$. The solutions to Equation 3 can be found using a fundamental matrix, that is, any matrix-valued function $\mathbf{M}(t)$ satisfying

$$\dot{\mathbf{M}} = \mathbf{J}_\mathbf{g}\mathbf{M}, \tag{4}$$

such that $\mathbf{M}(t)$ is nonsingular for all t. Focusing on a single perturbation, it follows from Equation 2a that

$$e^{\lambda(t)} = \left(\frac{\|\mathbf{v}(t)\|}{\|\mathbf{v}(0)\|}\right)^{1/t} = \left(\frac{\sqrt{\mathbf{v}(0)^{\mathrm{T}}\mathbf{M}(t)^{\mathrm{T}}\mathbf{M}(t)\mathbf{v}(0)}}{\sqrt{\mathbf{v}(0)^{\mathrm{T}}\mathbf{v}(0)}}\right)^{1/t}. \tag{5}$$

We are interested in the value of $\lambda(t)$ as $t \to \infty$. Rearranging Equation 5 and taking the limit, we have

$$\lambda = \lim_{t\to\infty} \ln\left[\left(\mathbf{v}(0)^{\mathrm{T}}\mathbf{M}(t)^{\mathrm{T}}\mathbf{M}(t)\mathbf{v}(0)\right)^{1/(2t)}\right]. \tag{6}$$

For almost all choices of $\mathbf{v}(0)$, the λ given by Equation 6 is the largest LE.

We consider now the full spectra of LEs $\lambda_i(t)$, $i = 1, \ldots, n$ that arise when one considers a sphere of perturbations. The growth of a sphere of perturbations depends only on $\mathbf{M}(t)^{\mathrm{T}}\mathbf{M}(t)$. Thus we consider the limit $\mathbf{W}(\mathbf{x}(0))$ defined by

$$\mathbf{W}(\mathbf{x}(0)) = \lim_{t\to\infty}\left[\mathbf{M}(t)^{\mathrm{T}}\mathbf{M}(t)\right]^{1/(2t)}. \tag{7}$$

By the multiplicative ergodic theorem (Oseledets, 1968; Ruelle, 1979), the limit exists, depends on the initial condition $\mathbf{x}$ (0), and, importantly, the eigendecomposition of $\mathbf{M}(t)^{\mathrm{T}}\mathbf{M}(t)$ in the limit exists, which gives

$$\mathbf{W}(\mathbf{x}(0)) = \mathbf{P}(\mathbf{x}(0))\mathbf{D}\mathbf{P}^{\mathrm{T}}(\mathbf{x}(0)), \tag{8}$$

where the eigenvector matrix $\mathbf{P}(\mathbf{x}\,(0))$ is orthonormal. The matrix of eigenvalues $\mathbf{D}$ is unique and depends neither on $\mathbf{x}(0)$ nor on the norm of the vector space containing the perturbations (Kuptsov and Parlitz, 2012). The LEs, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$, are the natural logarithm of the diagonal elements of $\mathbf{D}$.

We finish with some remarks on the significance of LEs. A chaotic system is a system with at least one positive LE. The LEs, defined above in terms of the growth of axes of a sphere of perturbations, are linked to the growth of the volume of the n-parallelepiped defined by the principle axes of the resulting ellipsoid (see Wolf *et al.*, 1985). Also, the sum of the LEs is equal to the average divergence of the flow (see Pikovsky and Politi, 2016, section 2.5.4). Thus, in dissipative systems, the sum of the LEs is negative. Finally, continuous chaotic systems have at least one

LE equal to zero. This is due to there being zero growth of an infinitesimal perturbation in the direction of the flow.

2.2 | Computation of local and global Lyapunov exponents

The theory does not translate directly to a method for calculation of the LEs, since, in order to approach the limit in Equation 7, one must integrate Equation 4 to find $\mathbf{M}(t)$ for very large t. This both accumulates numerical errors and results in a range of eigenvalues of $\mathbf{M}^{\mathrm{T}}(t)\mathbf{M}(t)$ that is too large for accurate numerical calculation (Pikovsky and Politi, 2016). Instead, we measure the growth of perturbations over (finitely) many small time intervals and compute the average. Specifically, for each time interval we calculate the LLEs: the natural logarithm of the growth ratios, divided by the length of the time interval, as shown in Equation 2b. If the system is ergodic, the arithmetic mean of the LLEs converges to the LEs as the number of time intervals increases. Crucially, the perturbation vectors are orthogonalized and resized between each time interval. Orthogonalising the propagated perturbations is necessary to keep the perturbations distinct, since perturbations will tend to be attracted towards the direction of largest growth. The resizing is necessary to prevent perturbations becoming too small or large to be represented by floating-point numbers. We now present the algorithm used to calculate LEs and LLEs in this work, which is based on methods presented in Benettin *et al.* (1980a); Benettin *et al.* (1980b) and Kuptsov and Parlitz (2012).

1 Calculate and store a long trajectory $\{\mathbf{x}(t) : t \in [0, T_{\mathrm{end}}]\}$. Discard an initial transient period to ensure the trajectory is in the attractor. One can alternatively calculate the trajectory at the same time as integrating Equation 4 in Step 3a below, which avoids the need to store a long trajectory.
2 Initialize a matrix of perturbations $\mathbf{Q}_0 = [\mathbf{q}_0^1, \mathbf{q}_0^2, \dots, \mathbf{q}_0^n]$, such that the $\mathbf{q}_0^i \in \mathbb{R}^n$ are orthogonal and of unit length, that is, orthonormal.
3 Repeat the following iteration m times, where m is large enough to achieve convergence of the LEs. In iteration j, starting with $j = 1$, perturbations are propagated along the trajectory $\{\mathbf{x}(t) : t \in [(j-1)\tau, j\tau]\}$, where τ is typically small. Each iteration results in n LLEs: λ_j^i, $i = 1, \dots, n$. Henceforth we notate LEs with hatted lambdas to distinguish the asymptotic LE $\hat{\lambda}_i$ from the LLE λ_j^i.
   (a) Propagate the perturbations: $\mathbf{V}_j = \mathbf{M}(j\tau)\mathbf{Q}_{j-1}$, where $\mathbf{Q}_{j-1}$ is from iteration $j-1$, and $\mathbf{M}(j\tau)$ is computed by integrating Equation 4.
   (b) Orthonormalize the propagated perturbations $\mathbf{V}_j$ to get $\mathbf{Q}_j$ using QR decomposition (Golub and Van Loan, 2013; Strang, 2016):

$$\mathbf{Q}_j\mathbf{R}_j = \mathbf{V}_j. \tag{9}$$

   (c) The diagonal elements r_j^i of $\mathbf{R}_j$ are the desired ratios. The LLEs at time $j\tau$ are calculated as

$$\lambda_j^i = \tau^{-1}\ln(r_j^{\alpha(i)}), \quad i = 1, \dots, n. \tag{10}$$

   In Equation 10, the diagonal element $r_j^{\alpha(i)}$ is indexed by labelling function $\alpha(i)$, where $\alpha(i)$ is determined in Step 4.
4 Calculate the LEs:

$$\hat{\lambda}_i = (m-k)^{-1}\sum_{j=k+1}^{m}\lambda_j^i, \qquad i = 1, \dots, n, \tag{11}$$

   where the LLEs from the first k iterations are discarded. The bijective function α (in Equation 10) takes inputs and values $1, \dots, n$, and is chosen such that the global LEs are numbered in descending order, that is, such that $\hat{\lambda}_i \geq \hat{\lambda}_{i+1}$. We say "$i$th LLE" to refer to any set of LLEs $\{\lambda_j^i : j = k, \dots, m\}$ that are associated with the ith LE.

In Step 4, a transient period of length k iterations is required to let the initial perturbations $\mathbf{Q}_0$ converge to the dynamics of the trajectory so that the leading perturbation $\mathbf{q}_j^1$ is oriented in the direction of the largest growth. As discussed above, the LLEs are defined in terms of ratios of the axes of the n-sphere and the ellipsoid. In practice, it is unlikely that the chosen initial perturbation $\mathbf{q}_0^1$ will be mapped by $\mathbf{M}(\tau)$ onto the leading axis of the ellipsoid, which will lead to a poor estimation of the LLE. However, with sufficiently many iterations k, $\mathbf{q}_k^1$ will be attracted to the direction of largest growth, leading to more accurate estimates.

Computational cost

The algorithm for computing the LLEs and LEs does not scale well. The computational costs of the steps of the algorithm are as follows. The length of the required transient period, that is, the number of "spin-up" iterations, depends on the system dynamics and, in the worst case, grows proportionally to the system dimension n (Kuptsov and Parlitz, 2012, p. 754). For computing LEs, the total number of iterations m depends on the complexity of the attractor and the precision required. Each

iteration (Step 3) requires the calculation of a trajectory of length τ, which involves at least $\mathcal{O}(n)$ floating-point operations (flops). The cost of integrating the matrix differential Equation 4 involves at least one evaluation of the Jacobian matrix, and at least one multiplication of the Jacobian by another matrix, per time step. For a dense, nontrivial Jacobian, it is reasonable to assume that Step b involves $\mathcal{O}(n^2)$ flops in the simplest case, namely where $\tau = \Delta t$ and a minimal numerical integration scheme is used. In a less simple case, Step b will require at least $\mathcal{O}(n^{2.3})$ flops: the cost of multiplying two dense $n \times n$ matrices (Alman and Williams, 2021). Step b is by far the most expensive step, since computing eigenvectors and eigenvalues via QR decomposition requires $25n^3$ flops (Golub and Van Loan, 2013; Arbenz, 2016). The overall theoretical time complexity of the LE algorithm is thus $25n^3 + \mathcal{O}(n^2)$ flops. In practice, $n \times n$ matrix multiplication can be much slower (at least $\mathcal{O}(n^3)$) due to memory access latency (Albrecht *et al.*, 2010). Consequently, computing the full LLE spectrum for a modern weather prediction system (where $n \approx 10^9$) is too expensive to be done during the forecast cycle.

Obviously, generating a subset of the LLE spectrum costs less. When using the tangent linear model, one must compute consecutive leading LLEs: it is not possible to calculate LLE λ^i without also calculating $\lambda^1, \ldots, \lambda^{i-1}$. The cost of QR decomposition of an $n \times i$ matrix scales at $\mathcal{O}(ni^2)$ (Boyd and Vandenberghe, 2018). The cost of multiplying an $n \times n$ matrix by an $n \times i$ matrix is $\mathcal{O}(n^2)$ when i is sufficiently smaller than n (in particular, at least when $\log_n(i) < 0.31389$, see Huang and Pan, 1998; Christandl *et al.*, 2020). In such cases, where $i \ll n$, the cost of computing the LLEs scales as $\mathcal{O}(n^2)$, as it is dominated by matrix multiplication rather than QR decomposition. Use cases where it suffices to know a subset of the LLE spectrum include assimilation in the unstable subspace (e.g., see Carrassi *et al.*, 2022) and computing the local Kaplan–Yorke dimension, which can also be exploited for better data assimilation (Quinn *et al.*, 2020).

3 | USING SUPERVISED MACHINE LEARNING TO ESTIMATE LYAPUNOV EXPONENTS

3.1 | Problem statement and evaluation metrics

Supervised learning refers to ML algorithms that use data sets formed of input–target pairs, whereby the goal is to construct a statistical model that emulates the idealised function that maps from the input to the target. The input and target are multidimensional arrays of data, not necessarily of the same dimensions. A single input–target pair is known as an example; the size of a ML data set refers to the number of examples it contains.

Supervised learning algorithms construct statistical models by optimising the model's parameters using the data. In the problem of this study, the input is the system state at a set of consecutive recent time steps including the current time t. The target is the vector containing the full spectrum of n LLEs calculated using the method described in Section 2.2 by integrating perturbations from time $t - \tau$ to t. We choose to estimate the full LLE spectrum; however, we note that the ML approaches we use can easily be adapted to the subproblem of estimating a subset of the LLE spectrum (henceforth "the subproblem"), such as unstable and near-neutral LLEs. See also the remarks in Sections 2.2 and 5.3. Generally, we have

$$(\text{input}, \text{target}) = \left((\mathbf{x}_{k-r}, \ldots, \mathbf{x}_{k-1}, \mathbf{x}_k), \quad (\lambda_k^1, \ldots, \lambda_k^n) \right), \tag{12}$$

where we recall that $\mathbf{x}_k \in \mathbb{R}^n$ denotes the system state at time $k\Delta t$ and λ_k^i is the ith LLE computed from the interval $[k\Delta t - \tau, k\Delta t]$.

Pointwise accuracy

By pointwise accuracy we refer to the ability of a ML algorithm to predict a specific LLE at an arbitrary time t. For its evaluation, we calculate a separate R^2 score for each LLE in the spectrum, from a set of d predictions and targets. The R^2 score, also known as the coefficient of determination, is given by

$$R^2\left(\left\{ \left(y_j, \hat{y}_j \right) | j = 1, \ldots, d \right\} \right) = 1 - \frac{\sum_{j=1}^{d} (y_j - \hat{y}_j)^2}{\sum_{j=1}^{d} (y_j - \bar{y})^2} \in (-\infty, 1], \tag{13}$$

where, for each j, $y_j \in \mathbb{R}$ is the target output (e.g., the ith LLE), $\hat{y}_j \in \mathbb{R}$ is the model's prediction, and $\bar{y}$ is the mean of the target outputs. In Equation 13, the numerator is known as the sum of squares of residuals and the denominator is known as the total sum of squares. An R^2 score of 1 is optimal and an R^2 score of 0 is as good as guessing the mean of the target values every time.

Similarity of prediction and target distributions

In addition to the pointwise accuracy, we evaluate the statistical accuracy of the ML models with quantile–quantile

TABLE 1 The four supervised learning algorithms used in this study.

Algorithm	Architecture
Multilayer perceptron (MLP)	One or more dense layers and one dense output layer
Regression tree (RT)	One tree per target LLE
Convolutional neural network (CNN)	One 1D convolution layer, max-pooling layer, flatten layer, one or more dense layers
Long-short term memory network (LSTM)	One or more LSTM layers, one dense output layer

Note: The hyperparameter values (e.g., number of dense layers) used in the experiment are described in Section 5.2.

(QQ) plots. QQ plots provide a simple nonparametric tool to compare the empirical probability distributions generated by two samples (Wilk and Gnanadesikan, 1968). In our case, these are the predicted and target values. To generate the plot, a set of quantiles (the 1000 quantiles in our experiments) is computed for both samples. These quantiles are then plotted against each other in a scatter plot. If the two samples have the same empirical distributions, the scatter plot renders a 45° diagonal line (of course this is subject to sampling error, which diminishes as the sample size grows). Departures from this ideal result show differences in the location and scale parameters of the empirical distributions, as well as possible linear and nonlinear relationships between the variables: see, for example, National Institute of Standards and Technology (U.S.) (2012) for a more detailed discussion. In our case, the QQ plots are useful to show which parts of the target distribution are represented well by the predictions.

3.2 | Supervised learning algorithms

We test four algorithms, summarised in Table 1, all well known in the ML community. They are chosen to represent commonly used, proven successful supervised learning algorithms. In this section we detail the algorithms, their structure, and their relative characteristics. The final details of the algorithms, including the number of parameters, are determined by hyperparameter tuning and described in Section 5.2. We note that superior performance in supervised learning tasks has been achieved by conducting a neural architecture search (NAS): an extensive (and costly) optimisation of neural network (NN) architecture from a vast and highly flexible search space (Zoph *et al.*, 2018). Here we stop short of conducting such a NAS. Instead, we choose established architectures for four distinct algorithms and carry out hyperparameter optimisation for each, where the hyperparameters include key architectural choices such as the number of layers and the number of neurons in each layer. We expect that the results from our selection will give a good indication of the possible performance of supervised learning in this task.

As we discuss in the following paragraphs, the chosen algorithms take different approaches to using the temporal structure of the input (when there are multiple time steps in the input). Here, by temporal structure we mean the temporal sequence of elements in each input vector, as opposed to the pairwise relation between the inputs and outputs while segmenting the time series for data-set preparation. The relative success of each algorithm gives insight into the nature of the problem from the ML perspective.

The first algorithm is the RT (Breiman *et al.*, 1984). The RT is the only non-neural network algorithm we test; RTs function by evaluating a finite chain of comparisons on the input features, such as "$x_k > 0$". The chain of comparisons forms a tree graph; each leaf node corresponds to an output value. Thus RTs have finitely many possible output values. The key advantage of a RT is the low computational cost of making predictions. Other advantages include the implicit feature selection process and potentially greater explainability of predictions compared with NNs. In fact, the RT may make use of only a few features from the set available in the input vector (Breiman *et al.*, 1984). By setting two hyperparameters (the maximum number of leaf nodes and the maximum depth, i.e., number of consecutive comparisons before a leaf node), it is possible to constrain the size of the resulting tree greatly. For simplicity, we opt to train a separate RT for each target LLE: that is, the prediction of the target vector $(\lambda_k^1, \ldots, \lambda_k^n)$ is made by n RTs, where each RT predicts a different λ_k^i.

The second algorithm is the MLP, the most basic type of feedforward artificial NN (as described in, *e.g.*, Goodfellow *et al.*, 2016, Chapter 6). The MLP is comprised of several hidden layers and an output layer, each of which is comprised of many neurons, where each neuron receives as input the outputs from all neurons in the previous layer (i.e., each layer is densely/fully connected). Each hidden layer has the same number of neurons; this value is optimised as a hyperparameter (see Section 5.2). When making a prediction, the entire input vector is passed to every

neuron in the first hidden layer. These neurons compute their outputs according to their weights and activation function, and their outputs are passed on to the neurons of the second layer, and so forth, until the neurons in the final layer produce outputs that are taken to be the final output of the algorithm. The MLP can theoretically approximate any continuous function (Hornik, 1991); however, in practise it has been found that in many problems accurate predictions are achieved more easily with more sophisticated architectures.

With regards to using temporal structure, the RT and the MLP take the same approach: to treat each element of the input vector (each feature) as an independent and identically distributed variable. Both algorithms have no inherent preference with respect to the temporal structure of the data, since they are invariant to the choice of order of the input features (a choice that is made in the preprocessing stage before training the algorithm).

The third algorithm is a CNN (LeCun *et al.*, 1989). In our experiments, the CNN is comprised of one 1D convolution layer with a kernel of size two and a stride of one, a max-pooling layer (if the number of time steps in the input is greater than 2), a flattening layer, and finally a set of dense layers (the number of which is optimised as described in Section 5.2). The 1D convolution layer is "1D" in that the layer convolves only over the time dimension. The kernel size of two means that, if the input is comprised of system state vectors at $r+1$ consecutive time steps, the convolution layer is only sensitive to patterns that occur in any time window of length two within the $r+1$ time steps. The max-pooling layer has a pool size and stride of two, which results in an invariance to translation of patterns by a single time step. Thus the CNN takes a different approach to using the temporal structure of the input, by only being sensitive to patterns that appear in small time windows within the input. In other words, the convolution layer predisposes the CNN to be aware which system states in the input are adjacent to each other in time. CNNs have been shown to be very successful in a range of applications, including in image-related tasks (LeCun *et al.*, 2015). The CNN architecture in our experiments is equivalent to the MLP with an additional 1D convolution layer at the beginning. We choose a 1D convolution (i.e., a convolution along the time dimension only) because, in the ordinary differential equation (ODE) systems in our experiments, there are only three state variables and there is no spatial locality that would motivate a focus on two of the variables at one time.

The fourth algorithm is the LSTM (Hochreiter and Schmidhuber, 1997; Graves, 2012). An LSTM is a form of recurrent neural network (RNN): the NN processes data in a sequence and, after each term in the sequence is processed, information is stored in a hidden state. In an LSTM, the flow of information in and out of the NN's hidden state is controlled by learned gates. LSTMs have been successful in various tasks where there are long-term dependences, that is, where the correct output at a later element in the sequence requires information from elements further back in the sequence. Such tasks include speech recognition (Graves *et al.*, 2013) and machine translation (Wu *et al.*, 2016). In machine translation, for instance, it is useful to retain information about words early on in the sentence to predict best how to translate words at the end of the sentence. However, the sophistication of the LSTM architecture comes at a computational cost, which we noticed particularly during training. In the experiments of this study, the LSTMs are composed of one to three LSTM layers and a dense output layer. The number of LSTM layers and the number of units in each is optimised as described in Section 5.2. The LSTM layers process the input one time step at a time, in chronological order. The estimation of the LLEs is made after the last time step has been processed. Thus, the LSTM takes a third approach to using the temporal structure by using information from past time steps when processing future time steps.

The NN algorithms (MLP, CNN and LSTM) were implemented using Tensorflow (Martín Abadi *et al.*, 2015). Additionally, all NN algorithms standardise the input and target data by subtracting the mean and dividing by the standard deviation, where the mean and standard deviation are calculated from the training set. Each component of the input and target is standardised independently. This is to make the data more amenable to learning. The scaling pipeline was implemented using Scikit-learn (Pedregosa *et al.*, 2011). For the RT, we use the implementation in the Scikit-learn Python module (Pedregosa *et al.*, 2011).

4 | RÖSSLER, LORENZ 63, AND THEIR LOCAL LYAPUNOV SPECTRA

In this section we present the two dynamical systems used in this work. We discuss the characteristics of their attractors and their LLEs, as this forms the data for the ML models and will be important in understanding their performance. The Rössler (Rössler, 1976) and Lorenz 63 (Lorenz, 1963) systems are both three-dimensional, continuous-time, ODE dynamical systems, given respectively by Equations 14 and 15. We use the parameters $(a, b, c) = (0.37, 0.2, 5.7)$ and $(\sigma, \rho, \beta) = (10, 28, 8/3)$. These are commonly chosen values for which the systems exhibit chaotic behaviour (see *e.g.* Ott, 2002). Under these settings, both systems are

dissipative, and they possess strange attractors of fractal dimension.

Rössler

$$\begin{aligned}\dot{x} &= -y - z,\\ \dot{y} &= x + ay,\\ \dot{z} &= b + z(x - c),\end{aligned} \tag{14}$$

Lorenz 63

$$\begin{aligned}\dot{x} &= \sigma(y - x),\\ \dot{y} &= x(\rho - z) - y,\\ \dot{z} &= xy - \beta z.\end{aligned} \tag{15}$$

The Lorenz 63 system is famous amongst weather and climate scientists: initially derived as a truncated model of Rayleigh–Bérnard convection in a two-dimensional fluid flow, it is the archetypal chaotic system and continues to be used in weather and climate science: for example, in data assimilation experiments (Carrassi *et al.*, 2018). For the chosen system parameters, the attractor of the Lorenz 63 system is formed of two wings, each centred around a nonstable fixed point. There is a third nonstable fixed point at the origin (Sparrow, 1982). The Rössler system was introduced as a simpler version of the Lorenz 63 system, having only one nonlinear term (zx) instead of the two in the Lorenz 63 system (xz and xy). Equation 14 was derived as a simplification of a system that combined two chemical reactions: a slow, two-variable oscillator (x and y) and a faster "switching-type" reaction (z) (Rössler, 1976). For a historical review of the development of the Rössler system , see Letellier and Messager (2010). The resulting system attractor is composed of a "disc" in the xy plane (z close to 0) and a "loop" in which trajectories rise (z increases rapidly) out of the disc, before folding over and back into the disc. Consequently there is an imbalance: any infinite, nonperiodic trajectory spends more time in the disc than in the loop. The attractor has two fixed points: an unstable fixed point in the centre of the disc, around which system trajectories spiral outwards, and a stable fixed point located outside the attractor. We shall see that the difference in the systems' dynamics provides a useful comparison.

It is easy to compute the analytic Jacobian matrices from Equations 14 and 15. Furthermore, the small size of both systems allows us to perform exhaustive experiments with long time series. The data used in our results are generated following Section 2.2. The evolution Equations 14 and 15, as well as the corresponding fundamental matrix Equation 4, are integrated using a fourth-order Runge–Kutta scheme with time step $\Delta t = 0.01$ for both systems. The LLEs are calculated over time windows of length $\tau = 0.04$ (i.e., four time steps); the LLEs at time t are calculated by integrating perturbations from time t to $t + \tau$. We choose $\tau = 0.04$, rather than a smaller value, so that the resulting set of LLEs provides better coverage of the attractor. Table 2 shows the resulting LEs. Further details of the data used in the results are given in Section 5.1.

In Figures 1 and 2, the top row shows the values of the LLEs along the systems' trajectories, specifically for the points $\mathbf{x}(\tau j)$ where $1200 \leq j \leq 26{,}199$; the values of the LLEs are given in color. We show only 25,000 data points to avoid saturating the figures: the local heterogeneity of the LLE values in phase space is evident. The bottom row displays the distribution of LLE values via histograms, for the points $\mathbf{x}(\tau j)$ where $1200 \leq j \leq 719{,}999$. The ith column shows the LLEs associated with the ith LE.

Let us first consider the statistical distribution of the LLEs. In the Rössler system (Figure 1) there is a marked difference between the first two LLE distributions and the third. The first two have a single, tall, thin mode with long tails and are roughly symmetric. The distribution of the third LLE has a lower-density mode with thicker tails and is negatively skewed. The range of values of the third distribution is also much smaller than those of the first two.

The Lorenz 63 system LLE distributions differ from those of the Rössler system. All three LLE distributions (Figure 2) have thicker tails and are positively skewed. The

TABLE 2 The LEs of the Rössler and Lorenz 63 systems as calculated following Section 2.

System	$\hat{\lambda}_1$	$\hat{\lambda}_2$	$\hat{\lambda}_3$
Rössler	0.19597 ± 0.00011	0.0000075 ± 0.0001275	−4.30097 ± 0.00068
Lorenz 63	0.90495 ± 0.000145	0.001975 ± 0.000055	−14.571345 ± 0.000105

Note: These values are computed from 718,800 LLE iterations, following a transient of $k = 1200$ iterations. As described in Section 2, the arithmetic mean of the LLEs converges to the LEs as one includes LLEs from more iterations, that is, from a longer trajectory. The convergence is not monotonic: the series of arithmetic means fluctuates as the number of iterations increases. To give an indication of the precision of the numerical calculation, we therefore calculate the LE as the midpoint of the range of the series of arithmetic means acquired from the first j iterations, where $j = 716{,}801, \ldots, 718{,}800$. The extent of the range above and below this value is also given. The proximity of the second LE to zero is a test of the accuracy of the numerical algorithm, since the middle LE is theoretically known to be zero in chaotic autonomous continuous-time systems (Pikovsky and Politi, 2016).

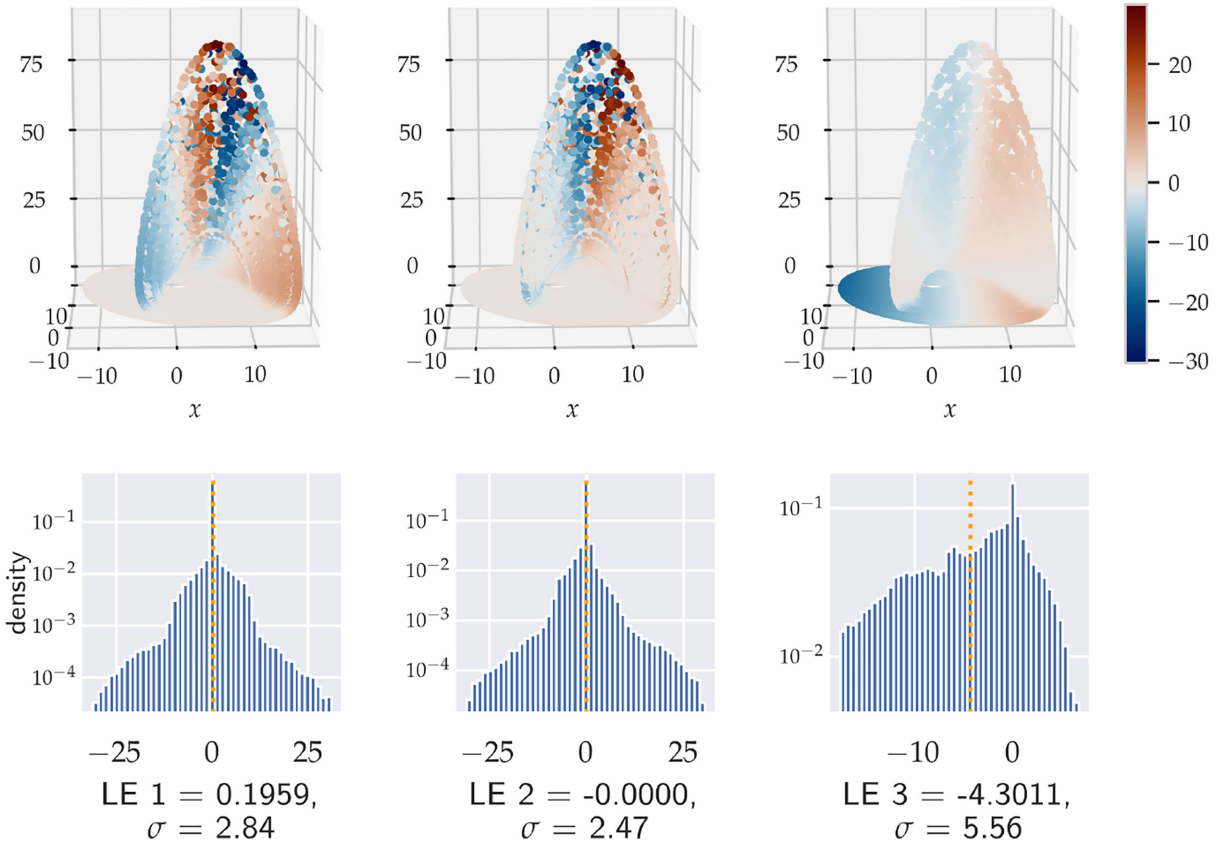

FIGURE 1 A long-time trajectory of the Rössler system coloured by LLE values (top row) shows how LLE values tend to be arranged in the system's attractor. The bottom row shows the corresponding statistical distribution of the LLEs via histograms; the mean of the LLE values is shown by the dotted (orange) line. The top panels show the same collection of 25,000 points. On the other hand, the histograms are generated from the full set of 718,800 LLEs; note that the vertical axis is plotted on a logarithmic scale. The mean (i.e., the corresponding LE) and standard deviation of the LLE values in each column are shown underneath. [Colour figure can be viewed at wileyonlinelibrary.com]

first and third are weakly bimodal, whilst the second is unimodal.

We now introduce a property that will be important to understanding the performance of the ML. In the context of the distribution of LLEs in the attractor of a system, we say that a region U of the attractor A is *(locally) heterogeneous* if the function $f : A \to \mathbb{R}$ mapping from A to the ith LLE value is nonsmooth in U. The best definition in more simple terms is that the LLE values in U vary in a nonsmooth fashion across U. For example, an alternating lattice (such as a chess board) would be locally heterogeneous. If U is not heterogeneous, we say it is homogeneous. In this work we use homogeneous and heterogeneous only in the sense of *locally across a small region of phase space* U, and not in the sense of across the entire attractor (i.e., globally), or along a trajectory (i.e., across time), as is the case in, for example, Vannitsem (2017) and Lucarini and Gritsun (2020).

For all three Rössler LLEs, in the disc of the attractor that sits in the xy-plane, the LLE values are homogeneous. In the loop that jumps out of the disc with positive z-values, there are bands of similar values for all three LLEs. For the first two LLEs, the most extreme LLE values are in the loop and there is significant local heterogeneity within the bands. For example, there are some small regions of the attractor where the first LLE is 10 or above for the majority of points, yet for some other points it is as low as −20. This high degree of heterogeneity is reflected in the longer tails of the distributions of the first and second LLEs (cf. Figures 1 and 2). In contrast, the third LLE is locally homogeneous: the colour change in the graph is smooth. Note that the trajectory of the Rössler system spends more time in the disc than in the loop. Consequently, data points in the loop are considerably sparser.

In the Lorenz 63 system, the first two LLEs have distinct regions of local heterogeneity and homogeneity. Unlike with the Rössler system, the regions of greater mixing are along the boundaries between regions of greater local homogeneity. The most extreme values of the first LLE are at the bottom of the attractor (z close to 0) and on the top edge of the wings: not in the regions of local heterogeneity. The third LLE, as with the Rössler system, is

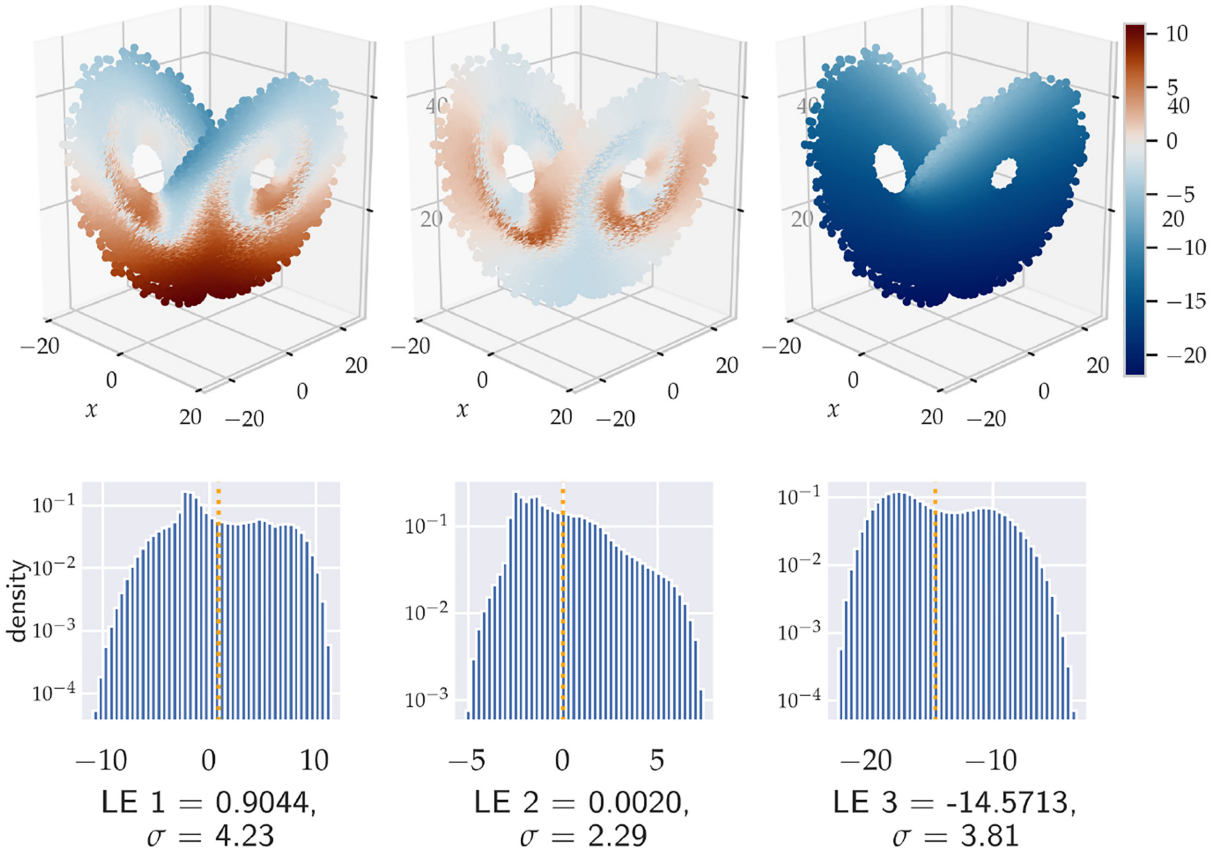

FIGURE 2 As Figure 1, but for Lorenz 63. [Colour figure can be viewed at wileyonlinelibrary.com]

locally homogeneous everywhere. As we will see, the characteristics of the LLEs we have discussed have implications for the performance of the ML methods trying to estimate them.

5 | EXPERIMENTS AND RESULTS

5.1 | Experiment setup

Having described the ML algorithms (Section 3.2) and dynamical systems (Section 4) used in this work, this section details the experiments and results. The experiments test a total of 16 configurations, which consist of each combination of two dynamical systems, four ML algorithms, and two input types:

$$\begin{aligned} &\text{Input Type 1:} && (\mathbf{x}_k), \\ &\text{Input Type 2:} && (\mathbf{x}_{k-5},\ \mathbf{x}_{k-4},\ \mathbf{x}_{k-3},\ \mathbf{x}_{k-2},\ \mathbf{x}_{k-1},\ \mathbf{x}_k). \end{aligned} \tag{16}$$

The first input type consists of the current time step only, whereas the second has the current time step and the five preceding time steps. In the envisaged operational application of this approach, storing multiple time steps of the entire model state poses a severe computational challenge. Since we envisage making LLE predictions every time step, and we have assumed that the pattern of input time steps remains fixed, the furthest-back input time step dictates the number of time steps that must be stored. We choose five time steps into the past as a balance between testing whether previous time steps can enable more accurate predictions and not requiring huge amounts of steps to be stored. Ignoring the constraints of feasibility as regards our approach, we expect that delays of more than five time steps might enable more accurate predictions. Given the choice of a maximum of five time steps into the past, we include all six time steps in the input and rely on the ML algorithms to extract useful features thereof. Note that, if there is only one time step in the input, the CNN is equivalent to an MLP where the number of units in the first hidden layer of the MLP is twice the number of filters in the CNN.

The main results were attained with data sets of 10^5 examples. The data sets were created using the method described in Section 2.2 and the parameters given in Section 4. For our chosen value of $\tau = 0.04$, the 10^5 examples are generated from a trajectory of 4×10^5 time steps, equating to 784 and 3604 Lyapunov times for Rössler and Lorenz 63, respectively. We note that another

study that estimated the global LEs of the Lorenz 63 system by emulating the dynamics with reservoir computers used a far smaller data set of 91 Lyapunov times (Pathak *et al.*, 2017). By inspecting plots of the trajectory (not shown here), we anticipated that 10^5 examples provide sufficient coverage of the system attractors. The Kullback–Leibler divergences of input and target variables (not included here) show that 10^5 examples should provides a good representation of the variables' statistical distributions (when compared with a data set of 10^6). To assess the impact of the data-set size, we repeated experiments with data sets of 5×10^4 and 10^6 examples. We found that the much larger data set resulted in only slightly more accurate predictions, with the exception of the LSTM. Thus the initial choice of 10^5 does not limit performance substantially. These results are discussed further in Section 5.3.

From each system's data set we generate 30 data-set instances, where each instance is a unique random shuffle of the original. Each data-set instance is partitioned into training, validation, and test sets with a ratio of 0.6 : 0.2 : 0.2. The resulting data setup is summarised in Table 3. The training, validation, and test sets of different data-set instances are therefore distinct. Given that the Rössler system and Lorenz 63 system are both ergodic, and the size of the training, validation, and test data sets is large, the coverage of the attractors by examples in each data-set instance is similar to what would be achieved if we had instead generated 30 sets of new data. Each configuration is tested on 30 data-set instances, that is, each configuration is tested in 30 trials. This provides an estimate of the variability of performance of each algorithm.

The NNs (MLP, CNN, LSTM) are implemented with two methods for preventing overfitting: activity regularisation on each layer and early stopping. Regularisation penalises large weight values; early stopping selects the model weights that score optimally on the validation data set, rather than on the training data set.

TABLE 3 A summary of key values in the data setup for the 10^5 data sets.

Number of data-set instances	30
Number of examples:	
In each data-set instance	100,000
In the training partition	60,000
In the validation partition	20,000
In the test partition	20,000

Note: Each data-set instance contains the same examples in a unique, shuffled order.

5.2 | Hyperparameter optimisation

In ML, *hyperparameter* refers to any parameter that has to do with the form of, or method of optimising, the statistical model, as opposed to the trainable parameters of the model itself (often referred to as weights). Typically, hyperparameters are fixed before the model is fitted to the data, that is, before the weights are optimised. For example, one hyperparameter for a NN is the learning rate: the amount by which model weights are changed at each step in the optimisation. Selecting the right hyperparameters is essential for effective use of ML algorithms (Goodfellow *et al.*, 2016).

We use a Bayesian optimisation method, implemented by Scikit-Optimize (Head *et al.*, 2021), to optimise the hyperparameters for each ML algorithm used in this study. Conceptually, a Bayesian optimisation method is an informed hyperparameter search that generates a probabilistic model (e.g., using a surrogate Gaussian process regression) of the true ML model (e.g., CNN) to select a set of hyperparameters that maximises the true ML model's performance (Snoek *et al.*, 2012).

We perform a separate hyperparameter optimisation for each configuration. Hyperparameter optimisation was carried out on Google's tensor processing units (TPUs) using Google Colaboratory. The search domain for each hyperparameter was chosen based on users' knowledge of the algorithms, the nature of the problem, and common practice in the ML community (Hastie *et al.*, 2009; Goodfellow *et al.*, 2016). The search domains are shown in Table 4. The chosen search space permits the NNs to be reasonably large (up to 200 neurons per layer and up to 10 layers for MLP and CNN, and up to 100 LSTM units per layer and up to 3 LSTM layers for LSTM) given the low dimensionality of the task: mapping from three (3 variables $\times$ 1 time steps) or 18 (3×6) features to three outputs. On the other hand, we opted to restrict the maximum number of layers for the LSTM due to the greater complexity of the algorithm. In fact, the first attempt to perform hyperparameter optimisation with a six-layer LSTM exceeded the 24 h runtime limit for Google Colaboratory. In contrast to the NNs, we forced the RTs to remain computationally light by limiting *maximum leaf nodes* (i.e., the maximum number of possible output values) to 100. The entire hyperparameter optimisation process had a combined runtime of 108 h. Although the hyperparameter optimisation was computationally expensive, it was affordable in the low-dimensional problems at hand. It greatly increases the chance that we attain maximal performance from each ML algorithm, thus providing useful insights into the problem from a ML perspective.

Table 5 shows the optimised hyperparameter values resulting from 50 iterations of the optimisation algorithm. Some optimal values are at the boundary of the search

TABLE 4 The search domain for hyperparameters of the ML algorithms.

Algorithm	Hyperparameter	Search domain
RT	Maximum depth	[1, 100]
	Maximum leaf nodes	[5, 100]
	Maximum features	{None, $\log_2$, square root}
	Splitter	{best, random}
	Min. cost–complexity pruning parameter	$[1 \times 10^{-6}, 100]$
	Minimum examples per leaf	[1, 20]
	Minimum weight fraction per leaf	$\{0, 0.5\}$
MLP	Learning rate	$[1 \times 10^{-6}, 0.9]$
	Number of layers	[1, 10]
	Number of neurons per layer	[1, 200]
	Activity regularisation on each layer	$\{L1(\alpha = 0.001), L2(\alpha = 0.001)$, None$\}$
CNN	Learning rate	$[1 \times 10^{-6}, 0.9]$
	Number of filters	[1, 100]
	Number of dense layers	[1, 10]
	Number of neurons per dense layer	[10, 200]
	Activity regularisation on each layer	$\{L1(\alpha = 0.001), L2(\alpha = 0.001)$, None$\}$
LSTM	Learning rate	$[1 \times 10^{-6}, 0.9]$
	Number of LSTM layers	[1, 3]
	Number of LSTM units per layer	[1, 100]
	Activity regularisation on each layer	$\{L1(\alpha = 0.001), L2(\alpha = 0.001)$, None$\}$

Note: For the MLP, CNN, and LSTM, the number of (dense) layers excludes the final densely connected output layer with three units.

domain (Table 4). This suggests that the algorithms might have performed better with hyperparameter values beyond the chosen search domain. For instance, in every experiment configuration, the optimised value of the RT hyperparameter *maximum leaf nodes* is 100, the maximum value in the search domain. Other examples of hyperparameters with optimised values at the boundary of the search domain for some configurations are the number of neurons per layer for the MLP and the CNN, and the number of LSTM units per layer. Nevertheless, the number of dense layers in the MLP and the CNN, as well as the number of LSTM layers, are mostly not at the boundary. Overall, from the results of the hyperparameter optimisation, we can argue that, although greater prediction accuracy might be achieved using NNs with larger layers (more neurons or LSTM units), the number of layers (NN depth) in the experiments is adequate.

To give an idea of the complexity of each of the four optimised ML algorithms, Table 6 shows the size of the ML models, measured by the maximum number of comparisons for the RT and the number of trainable parameters in the case of the NNs. The model size is a function of the optimised hyperparameter values. For the NNs with six time steps as input, the model size of those that predict the Lorenz 63 system LLEs is larger than those that predict the Rössler system. This is perhaps reflective of the more chaotic dynamics of the Lorenz 63 system, which has a larger first LE as well as two nonlinear terms in the equations as opposed to only one in the Rössler system (see Equations 14 and 15). The same is not true of the NNs that take one time step as input. This is unsurprising, since the one time step input contains less information on the dynamics, that is, how the state variables are changing in time. In this case, the size of the MLP and the CNN is smaller in the Lorenz 63 system , whereas the size of the LSTM is larger in the Lorenz 63 system. Due to their architectural similarity in the one-time-step case, it is unsurprising that the optimal model size for MLP and CNN behaves similarly.

The maximum size of the RTs is the same for all configurations. This is due to the tight restriction placed on the tree size by the search domain of the maximum leaf nodes hyperparameter (Table 4). However, the maximum depth hyperparameter varies across configurations

TABLE 5 The optimal hyperparameter values selected by 50 iterations of Bayesian optimisation.

			Optimised value			
			Rössler		Lorenz 63	
Algorithm	Target LLEs	Hyperparameter	One time step	Six time steps	One time step	Six time steps
RT	LLE 1	Maximum depth	78	58	100	100
		Minimum examples per leaf	20	13	15	16
	LLE 2	Maximum depth	100	24	100	78
		Minimum examples per leaf	20	20	1	1
	LLE 3	Maximum depth	17	48	100	72
		Minimum examples per leaf	2	1	14	20
MLP	All	Learning rate	7.300e−05	1.222e−04	7.628e−05	1.939e−05
		Number of dense layers	7	6	8	10
		Number of neurons per dense layer	200	182	165	200
		Activity regularisation on each layer	L2	L2	L2	L1
CNN	All	Learning rate	1.081e−04	1.586e−04	3.287e−04	1.016e−04
		Number of filters	100	37	29	59
		Number of dense layers	6	6	3	6
		Number of neurons per dense layer	200	200	122	200
		Activity regularisation on each layer	None	L1	L1	L1
LSTM	All	Learning rate	2.857e−03	1.874e−03	2.722e−03	1.105e−04
		Number of LSTM layers	3	1	2	2
		Number of LSTM units per layer	62	100	100	100
		Activity regularisation on each layer	L1	L1	L1	None

Note: For all RTs, the optimal maximum leaf nodes was 100, and the optimal minimum weight fraction per leaf was 0. Some RT hyperparameters are excluded for brevity.

TABLE 6 The size of the ML models, measured by the number of comparisons (RT) and number of trainable parameters (NNs).

		Rössler		Lorenz 63	
Algorithm		One time step	Six time steps	One time step	Six time steps
RT	Maximum comparisons	99	99	99	99
MLP	Trainable parameters	242,603	170,537	192,888	366,203
CNN	Trainable parameters	222,503	224,262	34,244	237,616
LSTM	Trainable parameters	78,557	41,903	122,303	122,303

Note: The maximum number of comparisons for each RT is 99 (one fewer than the maximum leaf nodes), since the max depth is too large to constrain the number of non leaf nodes in the tree.

and between LLEs. The optimal value of the maximum depth depends on the number of linear separations of the input space that improves the predictions during generation of the tree from the training data. For the same input type, and for a given LLE, the RT for the Lorenz 63 system has a greater maximum depth than the RT for the Rössler system . The greater maximum depth implies that more accurate predictions can be made by separating the input space at smaller scales (i.e., making finer-grained partitions of the input space) in the Lorenz 63 system compared with the Rössler system. In other words, there is clearer detail at smaller relative scales (in the input space) in the Lorenz 63 system than in the Rössler system.

5.3 | Results

Impact of data-set size

As discussed in Section 5.1, whilst we focus on results using a data set of 10^5 examples, we ran all experiments with three data-set sizes to determine the impact on prediction accuracy. We use the same setup for all data-set sizes: hyperparameters as described in Section 5.2, 30 trials, and a partition of 0.6 : 0.2 : 0.2 training–validation–testing. The mean R^2 scores from all data-set sizes are shown in Figure 3. In both systems, there are only small (or negligible) gains in accuracy from using 10^6 examples compared with 10^5 examples. The one exception is the LSTM: in both systems, the R^2 scores of the LSTM increased significantly with data-set size. In the Rössler system , the LSTM becomes the most accurate method when trained on 10^6 examples, whilst in the Lorenz 63 system the LSTM achieves accuracy similar to the MLP and CNN (indeed, Figure 3 shows that the six-time-step input LSTM achieves the best mean R^2 scores). In both systems, the variation of the LSTM over the 30 trials is substantially reduced with 10^6 examples compared with 10^5 examples (notably, for the one-time-step LSTM in Lorenz 63 , the variance of R^2 scores of LLE 2 reduces from 0.0201 to 0.0007). This suggests that choosing 10^5 examples strongly limits the performance of the LSTM. We suspect that the LSTM requires more data than the MLP and CNN due to its more complicated architecture, namely the hidden state and the three parameterised gates that control information flow into and out of the hidden state.

With regards to the smaller data set of 5×10^4, Figure 3 shows that the impact on accuracy is different in the two dynamical systems: in the Rössler system , the accuracy of MLP and CNN (especially with the one-time-step input) is strongly reduced (compared with the 10^5 data set), whereas for Lorenz 63 the equivalent reduction in accuracy is small. This is likely due to the greater sparsity of data points in the loop of the Rössler system. With these insights as context, the remainder of Section 5.3 refers to results from the 10^5 data sets, unless stated otherwise.

Comparisons between systems and across the LLE spectrum

We assess accuracy with the R^2 score of predictions made on the test data sets, each of which has 20,000 examples (see Table 3). There are 30 R^2 scores for each configuration: one from each data-set instance. These R^2 scores

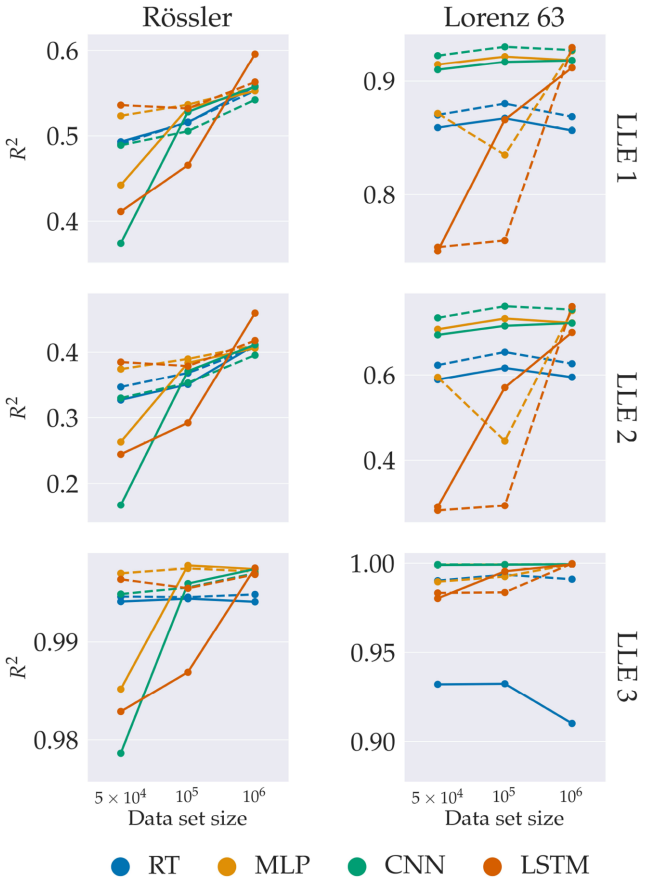

FIGURE 3 The impact of data-set size on mean R^2 scores across 30 trials. The solid lines indicate one-time-step results, the dashed lines indicate six-time-step results. [Colour figure can be viewed at wileyonlinelibrary.com]

are shown in boxplots in Figure 4 and summarized by their mean and standard deviation in Table 7. The immediate observation is that the R^2 scores differ consistently among LLEs (for a given system) and between systems (for a given LLE). The first LLE is predicted at least reasonably well in both dynamical systems (0.54 for Rössler system and 0.93 for Lorenz 63 system). The third LLE is well predicted in both systems, by all ML algorithms and with both types of input. Apart from one case, the mean R^2 scores for LLE 3 are above 0.98. This is unsurprising, given the local homogeneity of LLE 3 on the attractors, as discussed in Section 4. In all cases, the second LLE is the least well predicted (0.39 for the Rössler system and 0.76 for the Lorenz 63 system). This result is to be expected, since it is known that the second LLE is calculated least accurately by the numerical method (Kuptsov and Parlitz, 2012) and has a slower convergence (Bocquet *et al.*, 2017). Also, this result aligns with some recent attempts to emulate chaotic dynamics with ML methods, where the emulators have failed to reproduce near-neutral LEs accurately (Pathak *et al.*, 2017; Brajard

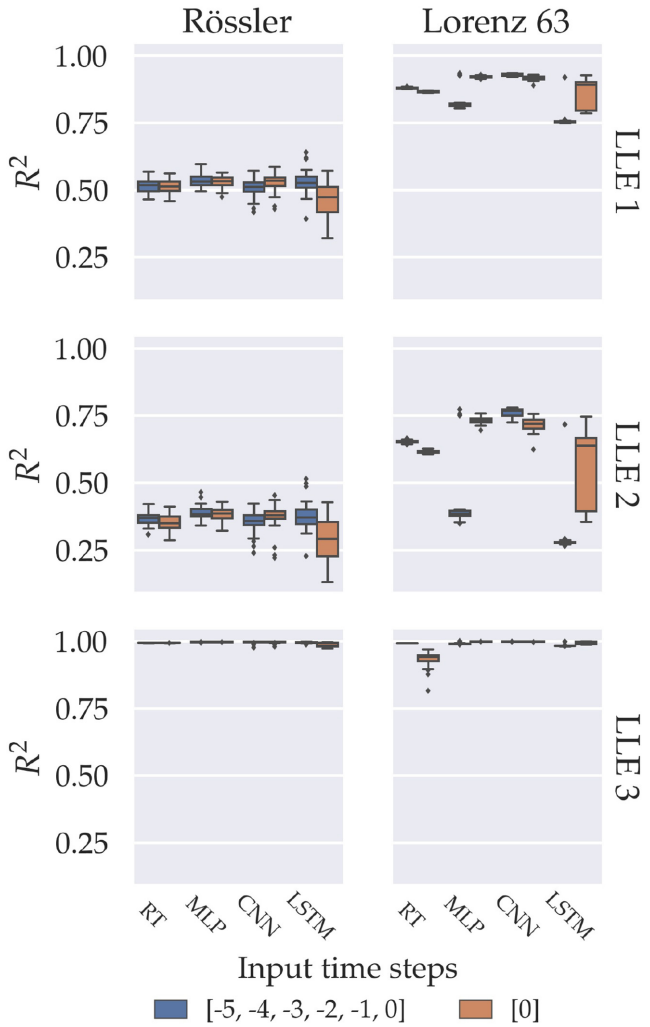

FIGURE 4 The R^2 scores of test data sets from the 30 trials, showing the variation across data-set instances, for each combination of system and ML method. Perfect predictions have an R^2 score of one. These results use 10^5 data sets: each test data set has 20,000 examples. [Colour figure can be viewed at wileyonlinelibrary.com]

et al., 2020). We note that, particularly in multiscale systems such as ocean–atmosphere systems, the neutral and near-neutral exponents play an important role in understanding predictability (see, e.g., De Cruz *et al.*, 2018; Quinn *et al.*, 2020) and are connected to the coupling mechanisms (Vannitsem and Lucarini, 2016; Tondeur *et al.*, 2020).

Next we compare prediction accuracy between dynamical systems. For LLEs 1 and 2, predictions of the Lorenz 63 system tend to be better than those of the Rössler system. The highest mean R^2 scores for LLEs 1 and 2 are 0.9304 and 0.7613 (respectively) for Lorenz 63, yet only 0.5365 and 0.3897 for Rössler. For LLE 3, the mean R^2 scores are similarly high in both systems. These results indicate that the LLEs can be predicted and the variability of the prediction accuracy depends on the LLE and the dynamical system being predicted.

Analysis of predictions on ordered test data

Figures 5 and 6 show time series of target values and predictions for a small set of ordered test data. The predictions are produced by the algorithm that achieves the best mean R^2 scores (on the 10^5 data sets): MLP for the Rössler system and CNN for the Lorenz 63 system. In both systems, LLE 3 is almost perfectly predicted throughout the time series. However, the predictions of LLEs 1 and 2 have error characteristics that are specific to each system.

Figure 5 illustrates that the first and second LLEs of the Rössler system vary intermittently: they are stationary and near the mean value for the majority of the time and then abruptly change and oscillate for a short period before returning to be close to the mean. This corresponds to the system trajectory being in the disc in the xy-plane, and then jumping into the "loop" with positive z-values, before returning to the disc. Predictions are extremely good during the stationary periods, and they are satisfactory during the peaks, which we label "fluctuation events". This is particularly true for LLE 1, where we see that the ML-based predictions always catch the fluctuation event and often its sign. The predictions of LLE 2 follows similar behaviour to LLE 1, however the R^2 score suggests that the pointwise accuracy is slightly worse than for LLE 1.

On the other hand, in the Lorenz 63 system, Figure 6 shows that LLEs 1 and 2 are constantly oscillating. Certain characteristics of the target time series are well reproduced by the predictions: for example, the largest peaks of LLE 1. These large peaks occur when the system trajectory passes close to the origin (cf. Figure 2), a region in which LLE 1 is locally homogeneous on the attractor. Nonetheless, small errors occur frequently. Notably, the higher-frequency features (such as the secondary peaks of LLE 2 between $t = 17$ and $t = 20$) are often relatively poorly reproduced for LLEs 1 and 2.

These time series provide further insight into the lower R^2 scores for LLEs 1 and 2 in the Rössler system compared with the Lorenz 63 system. Recall the definition of R^2 in Equation 13: the distance from the perfect score of 1 is the sum of squared residuals divided by the total sum of squares. The periods of stationarity in the Rössler system LLEs 1 and 2 contribute little to the total sum of squares. Consequently the larger errors during fluctuation events strongly reduce the R^2 score. In contrast, in the Lorenz 63 system, the R^2 score is high despite more frequent prediction errors, because the constant variation of the target values results in a larger total sum of squares.

TABLE 7 The table shows mean R^2 scores over 30 trials, with the corresponding standard deviations in parentheses.

Target	LLE 1		LLE 2		LLE 3	
Input type	**One time step**	**Six time steps**	**One time step**	**Six time steps**	**One time step**	**Six time steps**
R^2 on test data: mean and (standard deviation) of 30 data-set instances						
	Rössler					
RT	0.5155 (0.0248)	0.5161 (0.0268)	0.3506 (0.0299)	0.3681 (0.0278)	0.9944 (0.0002)	0.9946 (0.0002)
MLP	0.5323 (0.0211)	0.5363 (0.0243)	0.3837 (0.0249)	0.3897 (0.0274)	0.9978 (0.0005)	0.9975 (0.0006)
CNN	0.5279 (0.0333)	0.5054 (0.0349)	0.3711 (0.0518)	0.3530 (0.0419)	0.9960 (0.0040)	0.9956 (0.0044)
LSTM	0.4657 (0.0633)	0.5319 (0.0462)	0.2921 (0.0769)	0.3788 (0.0571)	0.9869 (0.0074)	0.9955 (0.0023)
	Lorenz 63					
RT	0.8672 (0.0025)	0.8801 (0.0027)	0.6166 (0.0046)	0.6540 (0.0053)	0.9324 (0.0305)	0.9936 (0.0003)
MLP	0.9217 (0.0038)	0.8350 (0.0438)	0.7325 (0.0123)	0.4449 (0.1446)	0.9993 (0.0003)	0.9925 (0.0033)
CNN	0.9169 (0.0081)	0.9304 (0.0047)	0.7153 (0.0261)	0.7613 (0.0159)	0.9992 (0.0005)	0.9993 (0.0002)
LSTM	0.8659 (0.0530)	0.7594 (0.0302)	0.5702 (0.1419)	0.2933 (0.0803)	0.9955 (0.0044)	0.9838 (0.0028)

Note: The R^2 score measures the accuracy of predictions: the optimum score is 1. We calculate the R^2 on the test data set for each of the 30 trials. The highest mean R^2 score for each combination of LLE and system (for both input types) is shown in bold. "One (six) time step(s)" refers to the number of time steps in the input.

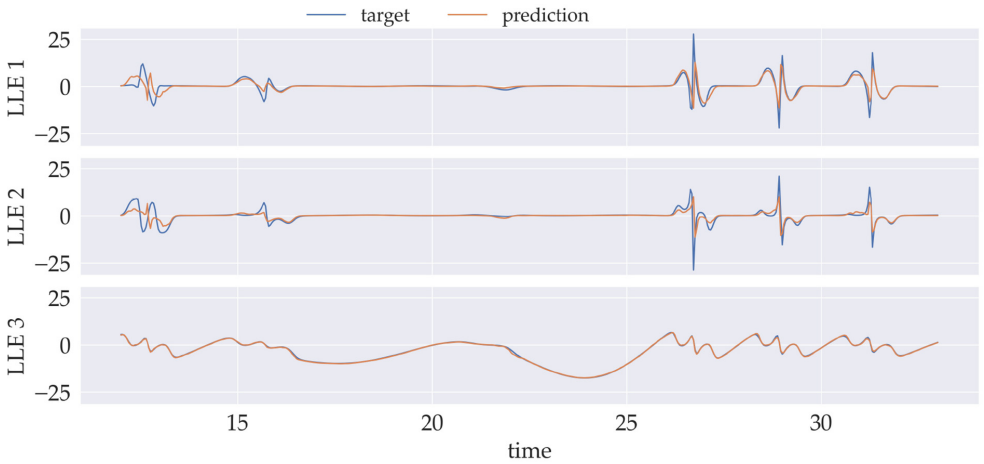

FIGURE 5 Time series of targets and predictions of test data from the Rössler system. Predictions made by an MLP with six input time steps. The R^2 scores for the period shown are 0.4393, 0.3175, and 0.9981 for LLEs 1, 2, and 3, respectively. [Colour figure can be viewed at wileyonlinelibrary.com]

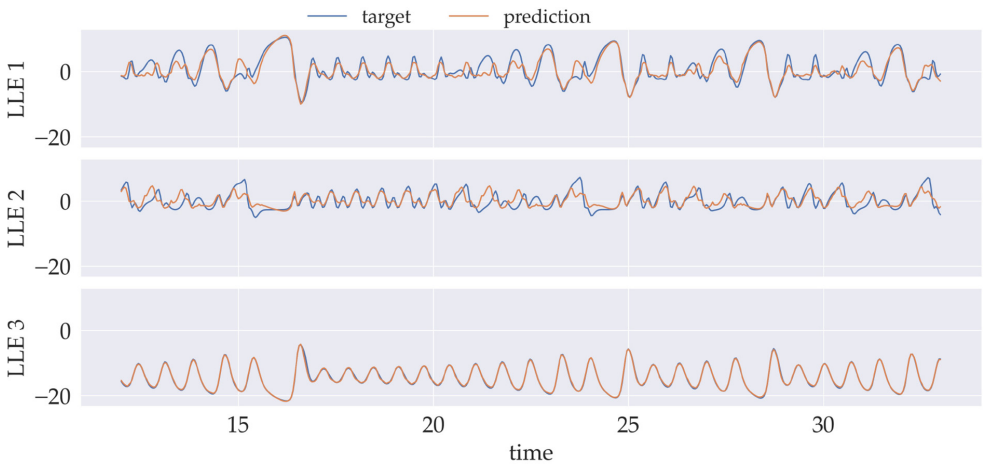

FIGURE 6 As with Figure 5 for the Lorenz 63 system. The predictions are made by a CNN with six input time steps. The R^2 scores for the period shown are 0.7981, 0.4897, and 0.9953 for LLEs 1, 2, and 3, respectively. [Colour figure can be viewed at wileyonlinelibrary.com]

Impact of local heterogeneity in phase space

The variability of the LLEs on the strange attractor of chaotic systems is a known feature, the immediate consequence of which is a highly state-dependent predictability horizon: two slightly different initial conditions can generate trajectories with hugely different degrees of instability. In a recent work, Lucarini and Gritsun (2020) have for the first time shown how this variability is related to the presence and distribution of unstable periodic orbits, each with a different degree of instability, densely filling the attractor. Arbitrary solutions are bounced among these unstable periodic orbits, taking their local instability features when they are in proximity.

Recall from Section 4 that, in both dynamical systems, there are regions of the system's attractor where the values of LLEs 1 and 2 are locally heterogeneous (LLE 3 is everywhere locally homogeneous). The locally heterogeneous regions in the Rössler system are in the loop with positive z-values, and in the Lorenz 63 system they form a strip that lies halfway between the outside edge and the centre of each wing. Figure 7 shows where the larger prediction errors occur on the attractor, for all configurations with a six-time-step input. More precisely, it shows the detraction from the perfect R^2 score of 1 contributed by each point. We see that, for all ML algorithms, larger errors occur in the locally heterogeneous regions. Moreover, the locally heterogeneous regions are robustly difficult: for the most accurate algorithm in the Lorenz 63 system (CNN), larger prediction errors only occur in these regions. This suggests that local heterogeneity plays a key role in determining where on the attractor it is possible for ML to make reliably accurate predictions of LLEs.

Figure 8 shows that a similar pattern occurs for the absolute relative error of predictions. The relative error compares the size of the error with the size of the target value. Notably, there are some large relative errors in the locally homogeneous regions of the Rössler system attractor (i.e., in the disc), since the relative error is especially punitive when the target value is close to zero.

The difficulty of making accurate predictions in these regions is in line with ML theory. ML algorithms work by optimising a model (such as a NN) to approximate the map from the input to the target of the training data. ML algorithms are successful if the optimised model also approximates the map from input to target on unseen data, such as test data. This is possible only if the training data provide enough information about the unseen data. The fundamental problem in locally heterogeneous regions is that the training data cannot provide enough information because the LLE values are noisy. In other words, the target values are highly variable, even as length-scales tend to zero. Thus the target values of unseen data are likely to be quite different from those of nearby seen data. Consequently, as local heterogeneity increases, ML models are less able to generalise from training data to unseen data.

The characteristics of the local heterogeneity explains the differences in prediction accuracy between the two dynamical systems. In the Rössler system, local heterogeneity in the loop of the attractor results in poorer predictions during the aforementioned fluctuation events. As explained above, errors during fluctuation events strongly reduce the R^2 score. On the other hand, in the Lorenz 63 system the values of LLEs 1 and 2 in their respective heterogeneous regions (see Figure 2) are relatively close to the mean: the largest deviations from the mean are in locally homogeneous regions. Therefore, the prediction errors resulting from locally heterogeneous regions are likely to be small compared with the deviation of the target values from the mean. Consequently, the differences in local heterogeneity explain the higher R^2 scores achieved for LLEs 1 and 2 of the Rössler system, compared with the Lorenz 63 system.

Impact of statistical distribution of targets and predictions

The poorer prediction accuracy for the Rössler system is also explained, to a lesser extent, by the statistical distribution of the LLE values. As described in Section 4, the Rössler system LLEs 1 and 2 include "extreme events", that is, values of large magnitude that appear infrequently. Predicting these extreme events is very challenging for any model, and particularly so for ML: one would need to enlarge the training data set commensurately to the (very long) return times of the extreme events. As noted above, we found a marked improvement in the LSTM performance with a tenfold increase in data-set size (see Figure 3).

A complementary picture of the prediction accuracy is given in Figure 9, which shows target values plotted against predicted values. The panels for the first and second LLEs of the Rössler system show that all four ML algorithms fail to predict the larger magnitude targets accurately. Similarly, the QQ plots in Figure 10 show that the predictions fail to replicate the extremities of the true values: the minimum and maximum of the predictions are lower in magnitude than those of the target. For instance, for LLE 1, we see that the NN methods make few predictions greater than 10 in magnitude, despite target values reaching magnitudes of 25. This behaviour can also be seen in the time-series plot (Figure 5). In the Lorenz 63 system, however, the larger-magnitude targets occur more frequently and are thus well represented in the training data set. Consequently, we see in Figure 10 that the larger

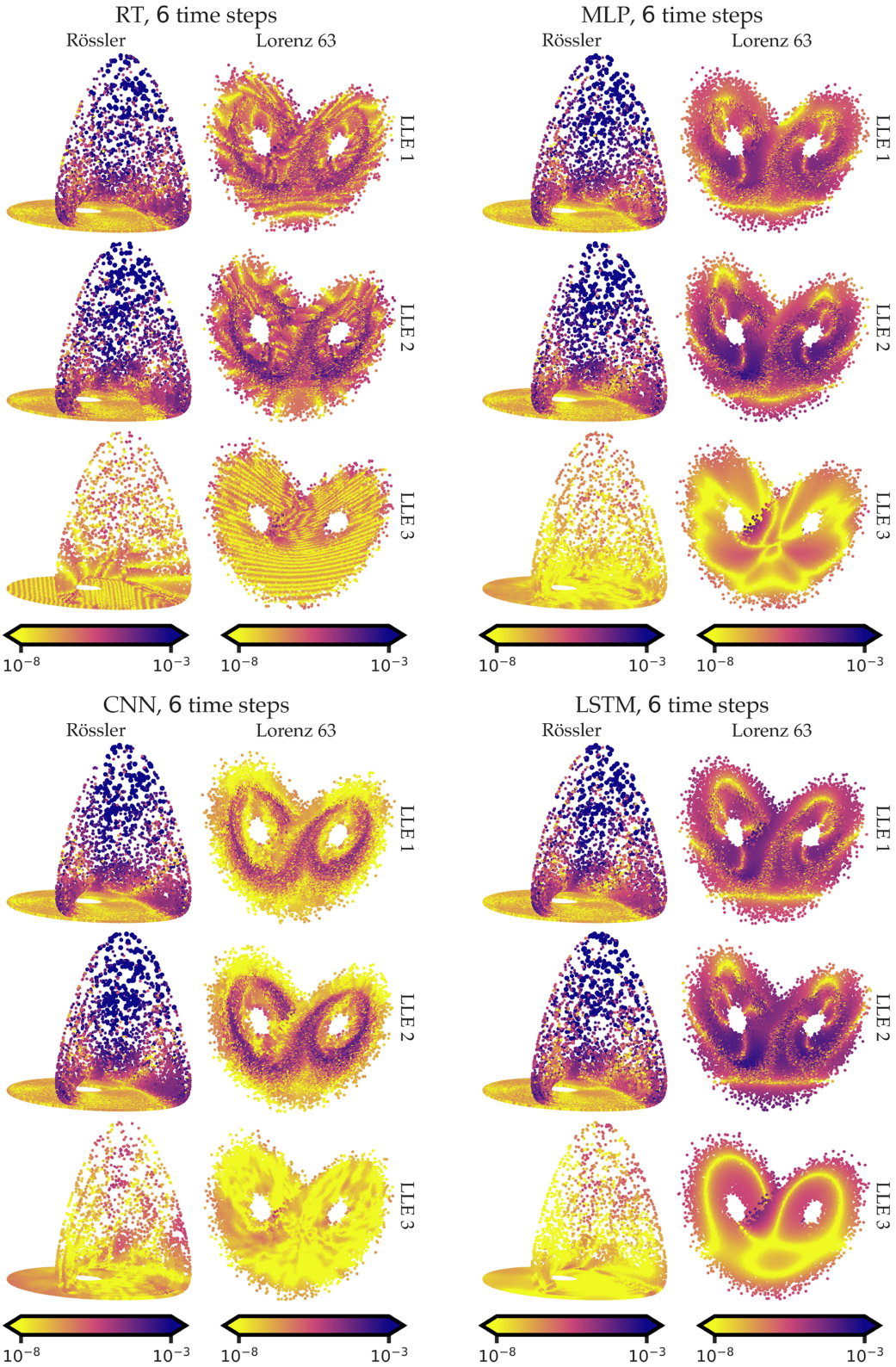

FIGURE 7 Recall the definition of the R^2 score (Equation 13): the distance from the perfect score of 1 is given by the sum of squares of residuals divided by the total sum of squares. Here we give a local (in phase space) description of the contribution towards that distance made by each prediction in a set of 20,000 test examples. Points are located at the current time step $\mathbf{x}_k$ of the input, and both coloured and sized by the square of the residual, divided by the total sum of squares. Darker points contribute a greater reduction to the R^2 score. If we denote the colour values as a_j, then $R^2 = 1 - \sum a_j$. We show all configurations with a six-time-step input. For each configuration, we use the data-set instance for which the R^2 score was closest to the mean (as shown in Table 7). [Colour figure can be viewed at wileyonlinelibrary.com]

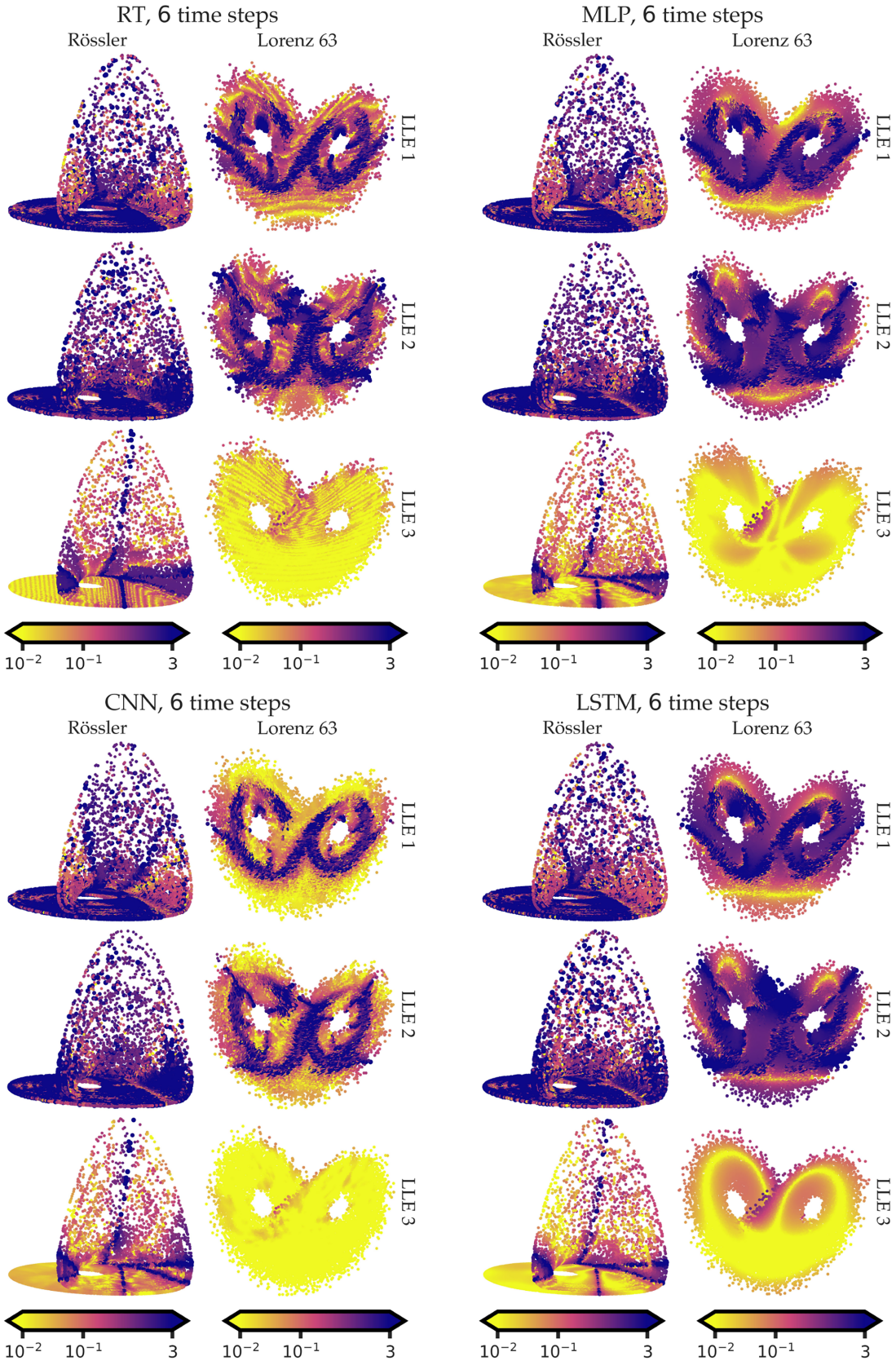

FIGURE 8 As with Figure 7, but point colour and size show the absolute relative error, given by $|y - \hat{y}|/(\epsilon + |\hat{y}|)$, where $\epsilon = 10^{-6}$, y is the prediction, and $\hat{y}$ is the target. The darker the point, the larger the absolute relative error of the prediction. [Colour figure can be viewed at wileyonlinelibrary.com]

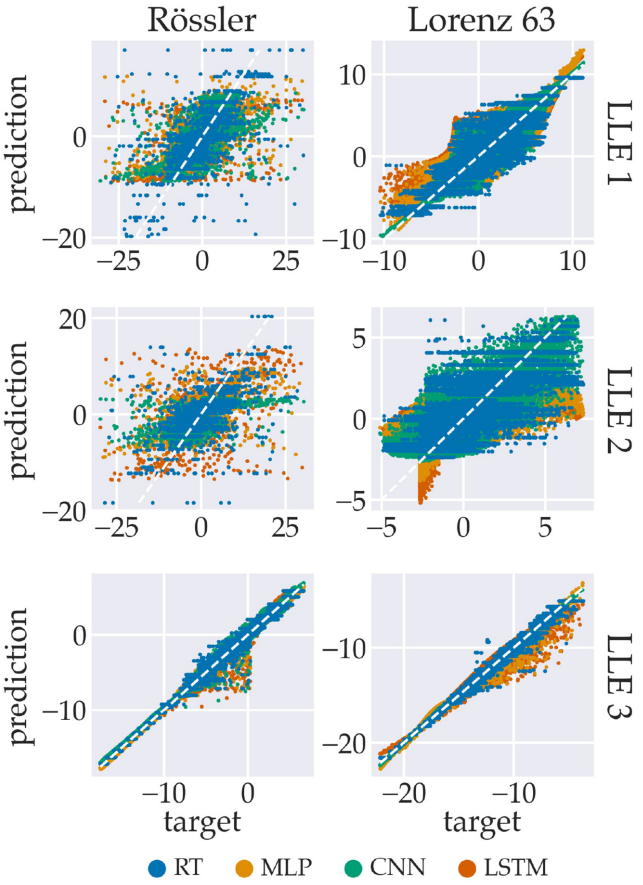

FIGURE 9 Scatter plots of targets versus predictions, for a test data set. Note that the axis scales differ in each panel. For each method–system combination, the data are from the data-set instance with R^2 closest to the mean for all data-set instances. Only results from setups with six input time steps are shown. Note that all scatter plots show different levels of heteroskedasticity, that is, the variance of a predicted value depends on the value of the target. This is a well-known challenge for regression methods. [Colour figure can be viewed at wileyonlinelibrary.com]

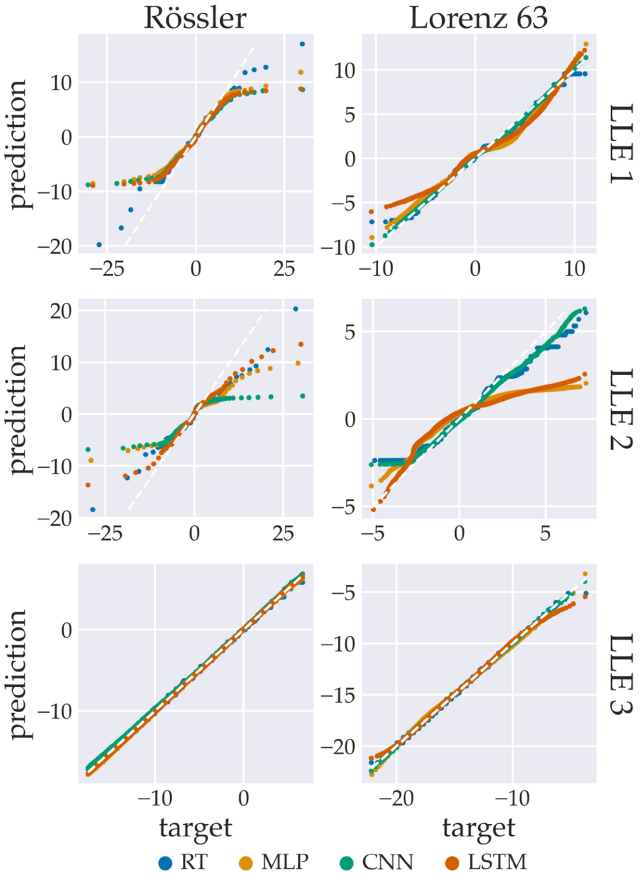

FIGURE 10 The 1000 quantiles of the predictions are plotted against those of the targets, revealing how well the two distributions match. The closer the graph to the $y = x$ line (dashed white), the closer the prediction distribution to the target distribution. Note that there is not necessarily any relationship between the proximity of distributions and (pointwise) accuracy of predictions. In each panel, and for each machine learning method, the quantiles are from the test data of the data-set instance with R^2 closest to the mean. Only results from setups with six input time steps are shown. [Colour figure can be viewed at wileyonlinelibrary.com]

magnitudes are well represented in the predictions distribution, even if the predictions sometimes fail to capture the amplitude of the targets on a pointwise basis, as illustrated in Figure 6.

Impact of ML approaches to exploiting temporal structure

Each of the ML algorithms we test takes one of three approaches to exploiting the temporal structure of the input, as discussed in Section 3.2. A comparison of these approaches can only be made when there is nontrivial temporal structure: this paragraph refers only to results with the six-time-step input. We find that there is no single optimal approach across both systems and all data-set sizes. Figure 4 shows that, with the 10^5 data set, the MLP and LSTM perform comparatively well in the Rössler system but far more poorly in the Lorenz 63 system. However, with the 10^6 data set, clearer patterns emerge: in both systems, the LSTM is the most accurate and the MLP and CNN perform similarly well. With the larger data set, there is not a clear distinction in performance between the three approaches.

Impact of input type

With the 10^5 data set, the input type has a big impact on the MLP and LSTM algorithms, particularly for LLE 2 of Lorenz 63: the mean R^2 score for MLP is 65% better with one time step than with six, and for the LSTM it is

TABLE 8 The time elapsed for one set of LLEs to be calculated (by the standard method) or predicted (by the ML algorithms).

Execution time per set of three LLEs						
		Standard method	**RT**	**MLP**	**CNN**	**LSTM**
Rössler	Mean	2.32e−4	9.96e−5	2.37e−2	2.38e−2	2.34e−2
	σ	2.16e−5	3.51e−6	2.36e−2	2.57e−3	2.51e−2
Lorenz 63	Mean	9.69e−4	9.88e−5	2.36e−2	2.37e−2	2.31e−1
	σ	2.53e−3	2.26e−6	2.85e−2	2.83e−2	1.60e−3

Note: The table shows the mean and standard deviation of elapsed time from 4000 trials.

94% better. However, with the 10^6 data set, clearer patterns emerge: in the Rössler system, the one-time-step input achieves the same or better mean R^2 for all algorithms, whereas in Lorenz 63 the six-time-step input achieves higher mean R^2 for all algorithms and lower variance of R^2 for all NN algorithms. This may be due to the more complicated dynamics of the Lorenz 63 system (see Section 4), or Lorenz 63 having a smaller Lyapunov time (1.1 compared with 5.1 in the Rössler system), meaning that six time steps provides more information in Lorenz 63. As mentioned in Section 5.1, in an idealised case (without computational constraints) one would optimise the input to include more distant time steps. The comparison between the 10^5 and 10^6 data sets shows that some NN algorithms require more data in order to exploit the additional information provided in the six-time-step input fully. With the 10^6 data set, the maximal difference of mean R^2 scores between input types (for any given algorithm) is small: one time step is 10% better in Rössler (LSTM, LLE 2) and six time steps is 9% better in Lorenz 63 (LSTM, LLE 2).

Among two input types we tested, the R^2 scores in Table 7 (for the 10^5 data set) suggest that, across all algorithms, there is only a small advantage to be gained by providing six time steps in the input rather than one. However, for a given algorithm, the input type can have a big impact. Figure 3 shows that this is especially true for the smaller data set of 5×10^4 in the Rössler system, where the NN prediction accuracy was far lower with the one-time-step input.

Computation time

As stated previously, this is a feasibility study focused on accuracy. Nevertheless, we discuss here briefly the computational cost, keeping in mind, however, that optimising the latter was not our priority. These arguments are thus included for context and completeness, and not as a proof of viability. Table 8 shows the mean (and standard deviation) time elapsed per prediction of the three LLEs, computed over 4000 trials on a CPU processor in a personal computer. The RT is two orders of magnitude faster than the NNs. This is expected: the RT has far fewer trainable parameters (see Table 6) and, unlike the NNs, does not require the evaluation of activation functions. The RT is the only ML algorithm that is faster than the standard method of computing LLEs (propagating perturbations and orthogonalising, ignoring the time taken by the spin-up iterations). Despite the comparative simplicity of the RT, it achieves R^2 scores that are close to those of the NNs, although it is less close in the Lorenz 63 system. It is likely that the number of output values (constrained by the maximum leaf nodes hyperparameter) limited the RT performance more strongly in the Lorenz 63 system, due to the greater complexity of the Lorenz 63 attractor.

Finally, we speculate on the cost of making predictions in the subproblem (see Section 3.1). The energy cost (measured in flops) of the RT scales linearly with the number of LLEs, since a separate tree is trained for each scalar target. However, the time cost remains the same, as the RTs can be executed in parallel. For the NNs, the potential cost saving is not clear without further experimentation. Less expensive NNs could be attained via pruning or distilling methods: see, for example, Molchanov *et al.* (2017).

6 | DISCUSSION AND SUMMARY

This study discusses the use of supervised machine learning (ML) to support numerical forecasting of chaotic dynamics. A huge amount of work has appeared recently at the crossroads between ML and the geosciences, whereby the former has provided novel data-driven solutions to complement or, in some ideal scenarios, substitute the physical models see *e.g.* Sonnewald *et al.*, 2021. In this work we took a different approach that we referred to as "nonintrusive".

We did not pursue improving the given physical model with a ML model, but rather using ML models as a supplementary tool that provides information to drive

adaptive decisions while running the prediction. The range of possible desirable information is ample—for example, anticipating a regime change or the onset of intense convective events—as is that of consequent actions. In this work, we focused on chaotic systems where real-time knowledge of the unstable properties of the system's state is of paramount relevance. Local Lyapunov exponents (LLEs) provide this knowledge in the form of the local (in time) exponential rates at which errors about the system's state evolve (Benettin *et al.*, 1980a; Pikovsky and Politi, 2016). Nevertheless, they are notably difficult to compute, require the coding and maintenance of a tangent linear model, and the computational cost grows fast with the system's size. This work is a feasibility study that investigates the accuracy with which supervised ML can estimate the LLEs of a dynamical system trajectory based only on the system state at the current time step and a few recent time steps.

We tested four supervised ML algorithms—a regression tree (RT), a multilayer perceptron (MLP), a convolutional neural network (CNN), and a long short-term memory network (LSTM)—on two dynamical systems (the Rössler system and Lorenz 63 systems). The dynamical systems are chaotic, dissipative, three-variable ODE systems. The algorithms encompass three approaches to exploiting the temporal structure of the input.

Our results indicate that the best algorithm depends on the dynamical system, the size of the data set, and the number of time steps included in the input. Overall, the results show that in certain conditions the LLEs can be predicted well: this depends on the system dynamics, the LLE being predicted, and, importantly, the local heterogeneity of the LLE in the proximity of the given state. In particular, the average accuracy was lowest for the neutral LLE. Further work is required to see if this result also holds in ocean–atmosphere systems (and multiscale systems more generally), where the neutral and near-neutral exponents are key to determining local predictability (De Cruz *et al.*, 2018; Quinn *et al.*, 2020). Our results suggest that the feasibility of using supervised ML to drive adaptive actions in an operational setting will depend on the specific use case: the forecasting model, the desired target information, and the intended adaptive actions.

Additionally, we investigated the impact of the size of the data set used to train the ML algorithms. We found that, with data sets of 10^6 examples, compared with 10^5 examples, the variance of the R^2 score reduced but there were only marginal improvements in the mean R^2 score. With the 10^6 data sets, the LSTM performed best in both systems. However, with the 10^5 data set, the LSTM was limited: the MLP performed best in the Rössler system whereas the CNN performed best in the Lorenz 63 system. The RT achieves an accuracy that is close to the best-performing algorithm in both systems, whilst being computationally much cheaper than the NNs. We tested two input types: one with one time step and one with six time steps (of the system state). We found the best input type depends on data-set size and dynamical system: in the Rössler system, six input time steps is better for the smallest data set (5×10^4), whilst in the Lorenz 63 system many more data (10^6) are required for the LSTM and MLP to achieve comparable performance with the six-time-step input. We further show that large prediction errors occur when the current state is in a region of local heterogeneity on the system attractor. Outside the locally heterogeneous regions, the best-performing algorithms make consistently accurate predictions. The differences in local heterogeneity between the two systems explain the lower R^2 scores achieved for LLEs 1 and 2 in the Rössler system, compared with the Lorenz 63 system. We explain that local heterogeneity is an insurmountable problem for deterministic ML predictions. This challenge could be mitigated if the ML prediction also included a reliable uncertainty quantification. We suspect that an uncertainty quantification could be made either by using Bayesian NNs (Wang and Yeung, 2016) or by including a measurement of the nearby local heterogeneity in the target of each example.

The low-dimensional setting permitted extensive experimentation in this work, providing lessons that will be useful should this "nonintrusive" approach be taken in weather and climate prediction. The next steps will be to apply the approach of this work to spatially extended models with more dimensions. There are several foreseeable challenges on the path from the very low-order models of this work to the envisioned setting of operational weather prediction models. The first challenge is to generate suitable data sets, since the calculation of LLEs does not scale well and requires a tangent linear model (see Section 2.2). However, the requirement of a tangent linear model can be avoided by using bred vectors (*e.g.* Toth and Kalnay, 1997; Uboldi and Trevisan, 2015). Additionally, the attractor of any numerical weather prediction (NWP) model is complex and high-dimensional: very large data sets will be required if sufficient attractor coverage is to be obtained. For a NWP model with $\mathcal{O}(10^8)$ variables, one would need $\mathcal{O}(10^{12})$ input–target pairs to obtain a ratio between the degrees of freedom of the NWP model and the size of the ML data set that is similar to the ratio used in this study. If one were to use ERA5 reanalysis data (Hersbach *et al.*, 2020) as input, a data set of $\mathcal{O}(10^{12})$ single time step inputs would amount to approximately $\mathcal{O}(10^9)$ TB of data, which is unfeasibly large. Furthermore, due to the long time-scales involved in teleconnection events, the number of such events can be small even in long time series. Given the number of input features, this can lead to a "small

data problem": see, for example, Vecchi *et al.* (2022) and references therein.

Therefore, it may be necessary to generate training data using a reduced-dimension version of the operational model. For example, in Quinn *et al.* (2021); Quinn *et al.* (2022), LLEs are computed by reducing the data dimension (via empirical orthogonal functions) and constructing a multistate vector autoregressive model.

Once initial training data have been generated, the cost of making predictions with the ML model can be reduced by reducing the dimension of the training data further, that is, by performing feature extraction (Guyon *et al.*, 2006). For example, ML techniques such as autoencoders can be used for dimension reduction: for example, see Mack *et al.* (2020). Also, it may be possible to curate training data sets strategically to reduce their size. Finally, if the intended use case requires only part of the LLE spectrum, then cost savings can be made (a) when generating training data, which scales as $\mathcal{O}(n^2)$ rather than $\mathcal{O}(n^3)$ (see Section 2.2), and (b) when making predictions (see Section 5.3).

The computational benefit of the ML approach investigated here is twofold. The ML approach estimates LLEs directly from the current system state, thus avoiding the cost of the long spin-up that is required by the conventional method for calculating LLEs. Second, the ML approach has the potential to be cheaper per iteration of LLEs. We found that the lightest algorithm we tested, the RT, was computationally cheaper (by a factor of 10) than the conventional method for calculating LLEs (see Table 8).

Although the NNs were comparatively costly in this setting, we expect that in a higher-dimensional, operational setting, NNs may be competitive. It is unknown how the required NN size will increase with the system dimension: this will require experimentation. The time cost of making predictions with NNs may be reduced (relative to the size of the NNs) by using purpose-built ML hardware. On the other hand, the cost of calculating LLEs numerically (by propagating perturbations and orthogonalising) will scale as $\mathcal{O}(n^3)$ if computing the full spectrum, or $\mathcal{O}(n^2)$ if the number of LLEs computed is much smaller than the dimension of the NWP model n.

AUTHOR CONTRIBUTIONS

Daniel Ayers: conceptualization; data curation; formal analysis; investigation; methodology; software; visualization; writing – original draft; writing – review and editing. **Jack Lau:** formal analysis; investigation; methodology; software; writing – original draft. **Javier Amezcua:** conceptualization; formal analysis; funding acquisition; methodology; supervision; writing – review and editing. **Alberto Carrassi:** conceptualization; formal analysis; funding acquisition; methodology; supervision; writing – review and editing. **Varun Ojha:** conceptualization; formal analysis; funding acquisition; methodology; supervision; writing – review and editing.

ACKNOWLEDGEMENTS

Daniel Ayers is funded by a studentship from the Engineering and Physical Sciences Research Council (EP/N509723/1). Javier Amezcua and Alberto Carrassi acknowledge the support of the UK National Centre for Earth Observation (grant no. NCEO02004). Alberto Carrassi is also supported by the project SASIP funded by Schmidt Futures—a philanthropic initiative that seeks to improve societal outcomes through the development of emerging science and technologies.

ORCID

Daniel Ayers https://orcid.org/0000-0002-5667-8174
Javier Amezcua https://orcid.org/0000-0002-4952-8354
Alberto Carrassi https://orcid.org/0000-0003-0722-5600
Varun Ojha https://orcid.org/0000-0002-9256-1192

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. (2015) TensorFlow: large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Albarakati, A., Budišić, M., Crocker, R., Glass-Klaiber, J., Iams, S., Maclean, J., Marshall, N., Roberts, C. and Van Vleck, E.S. (2021) Model and data reduction for data assimilation: particle filters employing projected forecasts and data with application to a shallow water model. *Computers & Mathematics with Applications*, 116, 194–211. https://doi.org/10.1016/j.camwa.2021.05.026.

Albrecht, M., Bard, G. and Hart, W. (2010) Algorithm 898: efficient multiplication of dense matrices over GF(2). *ACM Transactions on Mathematical Software*, 37, 1. https://doi.org/10.1145/1644001.1644010.

Alman, J. and Williams, V.V. (2021) "A refined laser method and faster matrix multiplication". *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Philadelphia, PA: Society for Industrial and Applied Mathematics, pp. 522–539. https://doi.org/10.1137/1.9781611976465.32.

Arbenz, P. (2016) *The QR Algorithm. In "Lecture notes on Numerical Methods for Solving Large Scale Eigenvalue Problems"*, pages 63-90. http://people.inf.ethz.ch/arbenz/ewp/lnotes.html.

Benettin, G., Galgani, L., Giorgilli, A. and Strelcyn, J.-M. (1980a) Lyapunov characteristic exponents for smooth dynamical systems and for Hamiltonian systems; a method for computing all of them. Part 1: theory. *Meccanica*, 15, 9–20. https://doi.org/10.1007/BF02128236.

Benettin, G., Galgani, L., Giorgilli, A. and Strelcyn, J.-M. (1980b) Lyapunov characteristic exponents for smooth dynamical systems and for Hamiltonian systems; a method for computing all of them. Part 2: numerical application. *Meccanica*, 15, 21–30. https://doi.org/10.1007/BF02128237.

Bishop, C.M. (1995) *Neural networks for pattern recognition*. Oxford: Clarendon Press.

Bocquet, M., Gurumoorthy, K.S., Apte, A., Carrassi, A., Grudzien, C. and Jones, C.K.R.T. (2017) Degenerate Kalman filter error covariances and their convergence onto the unstable subspace. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1), 304–333. https://doi.org/10.1137/16M1068712.

Bolton, T. and Zanna, L. (2019) Applications of deep learning to ocean data inference and subgrid parameterization. *Journal of Advances in Modeling Earth Systems*, 11(1), 376–399. https://doi.org/10.1029/2018MS001472.

Bonavita, M. and Laloyaux, P. (2020) Machine learning for model error inference and correction. *Journal of Advances in Modeling Earth Systems*, 12, 12. https://doi.org/10.1029/2020MS002232.

Boyd, S. and Vandenberghe, L. (2018) *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*. Cambridge: Cambridge University Press. https://doi.org/10.1017/9781108583664.

Brajard, J., Carrassi, A., Bocquet, M. and Bertino, L. (2020) Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: a case study with the Lorenz 96 model. *Journal of Computational Science*, 44, 101171.

Brajard, J., Carrassi, A., Bocquet, M. and Bertino, L. (2021) Combining data assimilation and machine learning to infer unresolved scale parametrization. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194), 20200086. https://doi.org/10.1098/rsta.2020.0086.

Breiman, L., Friedman, J., Stone, C. and Olshen, R. (1984) *Classification and Regression Trees*. New York, NY: Taylor & Francis. https://doi.org/10.1201/9781315139470.

Buizza, R. (2019) Introduction to the special issue on "25 years of ensemble forecasting". *Quarterly Journal of the Royal Meteorological Society*, 145, 1–11.

Carrassi, A., Bocquet, M., Bertino, L. and Evensen, G. (2018) Data assimilation in the geosciences: an overview of methods, issues, and perspectives. *WIREs Climate Change*, 9(5), 1–79. https://doi.org/10.1002/wcc.535.

Carrassi, A., Bocquet, M., Demaeyer, J., Grudzien, C., Raanes, P. and Vannitsem, S. (2022) Data assimilation for chaotic dynamics. In: Park, S.K. and Xu, L. (Eds.) *Data assimilation for atmospheric, oceanic and hydrologic applications (Vol. IV)*. Cham: Springer International Publishing, pp. 1–42. https://doi.org/10.1007/978-3-030-77722-7_1.

Chantry, M., Christensen, H., Dueben, P. and Palmer, T. (2021) Opportunities and challenges for machine learning in weather and climate modelling: hard, medium and soft AI. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194), 20200083. https://doi.org/10.1098/rsta.2020.0083.

Chen, Y., Carrassi, A. and Lucarini, V. (2021) Inferring the instability of a dynamical system from the skill of data assim-ilation exercises. *Nonlinear Processes in Geophysics*, 28(4), 633–649. https://doi.org/10.5194/npg-28-633-2021.

Christandl, M., Gall, F.L., Lysikov, V. and Zuiddam, J. (2020). Barriers for rectangular matrix multiplication. *Electronic Colloquium on Computational Complexity*, https://doi.org/10.48550/ARXIV.2003.03019.

De Cruz, L., Schubert, S., Demaeyer, J., Lucarini, V. and Vannitsem, S. (2018) Exploring the Lyapunov instability properties of high-dimensional atmospheric and climate models. *Nonlinear Processes in Geophysics*, 25(2), 387–412. https://doi.org/10.5194/npg-25-387-2018.

Düben, P., Modigliani, U., Geer, A., Siemen, S., Pappenberger, F., Bauer, P., Brown, A., Palkovic, M., Raoult, B., Wedi, N. and Baousis, V. (2021) Machine learning at ECMWF: a roadmap for the next 10 years. Tech. rep. 878. ECMWF. https://doi.org/10.21957/ge7ckgm.

Fablet, R., Ouala, S. and Herzet, C. (2018) "Bilinear residual neural network for the identification and forecasting of geophysical dynamics". *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, pp. 1477–1481. https://doi.org/10.23919/EUSIPCO.2018.8553492.

Froyland, G., Hüls, T., Morriss, G.P. and Watson, T.M. (2013) Computing covariant Lyapunov vectors, Oseledets vectors, and dichotomy projectors: a comparative numerical study. *Physica D: Nonlinear Phenomena*, 247(1), 18–39. https://doi.org/10.1016/j.physd.2012.12.005.

Ginelli, F., Poggi, P., Turchi, A., Chaté, H., Livi, R. and Politi, A. (2007) Characterizing dynamics with covariant Lyapunov vectors. *Physical Review Letters*, 99(13), 130601. https://doi.org/10.1103/PhysRevLett.99.130601.

Golub, G.H. and Van Loan, C.F. (2013) *Matrix computations*, 4th. edition. Baltimore, MD: Johns Hopkins University Press.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep learning*. Cambridge, MA: MIT Press. http://www.deeplearningbook.org.

Gottwald, G.A. and Reich, S. (2021) Supervised learning from noisy observations: Combining machine-learning tech-niques with data assimilation. *Physica D: Nonlinear Phenomena*, 423, 132911. https://doi.org/10.1016/j.physd.2021.132911.

Graves, A. (2012) Long short-term memory. In: *Supervised sequence labelling with recurrent neural networks. studies in computational intelligence*, Vol. 385. Berlin: Springer, pp. 37–45. https://doi.org/10.1007/978-3-642-24797-2_4.

Graves, A., Mohamed, A.-R. and Hinton, G. (2013) "Speech recognition with deep recurrent neural networks". *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649. https://doi.org/10.1109/ICASSP.2013.6638947.

Guyon, I., Gunn, S.R., Nikravesh, M. and Zadeh, L.A. (2006) *Feature extraction. foundations and applications*. Berlin: Springer-Verlag. https://doi.org/10.1007/978-3-540-35488-8.

Hastie, T., Tibshirani, R. and Friedman, J. (2009) *The elements of statistical learning*. New York, NY: Springer. https://doi.org/10.1007/978-0-387-84858-7.

Head, T., Kumar, M., Nahrstaedt, H., Louppe, G. and Shcherbatyi, I. (2021) *scikit-optimize/scikit-optimize*. Version v0.9.0. https://doi.org/10.5281/zenodo.5565057.

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Hornyi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., Simmons, A., Soci, C., Abdalla, S., Abellan, X., Balsamo, G., Bechtold, P., Biavati, G., Bidlot, J., Bonavita, M., De Chiara, G., Dahlgren, P., Dee, D., Diamantakis, M., Dragani, R., Flemming, J., Forbes, R., Fuentes, M., Geer, A., Haimberger, L., Healy, S., Hogan, R.J., Hólm, E., Janiskov, M., Keeley, S., Laloyaux, P., Lopez, P., Lupu, C., Radnoti, G., de Rosnay, P., Rozum, I., Vamborg, F., Villaume, S. and Thépaut, J.-N. (2020) The ERA5

global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730), 1999–2049. https://doi.org/10.1002/qj.3803.

Hochreiter, S. and Schmidhuber, J. (1997) Long short-term memory. *Neural Computation*, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Holton, J.R. and Hakim, G.J. (2013) *An introduction to dynamic meteorology*. Oxford: Elsevier. https://doi.org/10.1016/C2009-0-63394-8.

Hornik, K. (1991) Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2), 251–257. https://doi.org/10.1016/0893-6080(91)90009-T.

Huang, X. and Pan, V.Y. (1998) Fast rectangular matrix multiplication and applications. *Journal of Complexity*, 14(2), 257–299. https://doi.org/10.1006/jcom.1998.0476.

Kalnay, E. (2002) *Atmospheric modeling, data assimilation and predictability*. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511802270.

Kaplan, J.L. and Yorke, J.A. (1979) Chaotic behavior of multidimensional difference equations. In: Peitgen, H.-O. and Walther, H.-O. (Eds.) *Functional differential equations and approximation of fixed points*. Berlin, Heidelberg: Springer, pp. 204–227.

Kuptsov, P.V. and Parlitz, U. (2012) Theory and computation of covariant Lyapunov vectors. *Journal of Nonlinear Science*, 22(5), 727–762. https://doi.org/10.1007/s00332-012-9126-5.

LeCun, Y., Bengio, Y. and Hinton, G. (2015) Deep learning. *Nature*, 521(7553), 436–444. https://doi.org/10.1038/nature14539.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W. and Jackel, L. (1989) Handwritten digit recognition with a back-propagation network. In: *Proceedings of the 2nd international conference on neural information processing systems*. Cambridge, MA: MIT Press, pp. 396–404.

Legras, B. and Vautard, R. (1996) A guide to Liapunov vectors. In: Palmer, T. (Ed.) *Seminar on predictability*, Vol. 1. Reading, UK: ECMWF, pp. 135–146.

Letellier, C. and Messager, V. (2010) Influences on Otto E. Rössler's earliest paper on chaos. *International Journal of Bifurcation and Chaos*, 20(11), 3585–3616. https://doi.org/10.1142/S0218127410027854.

Lighthill, J., Thompson, J.M.T., Sen, A.K., Last, A.G.M., Tritton, D.T. and Mathias, P. (1986) "The Recently Recognized Failure of Predictability in Newtonian Dynamics [and Discussion]". *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* 407.1832, pp. 35–50.

Lorenz, E.N. (1963) Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2), 130–141. https://doi.org/10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.

Lucarini, V. and Gritsun, A. (2020) A new mathematical framework for atmospheric blocking events. *Climate Dynamics*, 54(1), 575–598.

Mack, J., Arcucci, R., Molina-Solana, M. and Guo, Y.-K. (2020) Attention-based convolutional autoencoders for 3D-variational data assimilation. *Computer Methods in Applied Mechanics and Engineering*, 372, 113291. https://doi.org/10.1016/j.cma.2020.113291.

Molchanov, P., Tyree, S., Karras, T., Aila, T. and Kautz, J. (2017) "Pruning Convolutional Neural Networks for Resource Efficient Inference". *International Conference on Learning Representations*.

National Institute of Standards and Technology (U.S.). (2012) *NIST/SEMATECH e-Handbook of Statistical Methods*. [Online; accessed 21-December-2021]. https://doi.org/10.18434/M32189.

Nguyen, D., Ouala, S., Drumetz, L. and Fablet, R. (2019). EM-like learning chaotic dynamics from noisy and partial observations. *CoRR*, abs/1903.10335. https://doi.org/10.48550/arXiv.1903.10335.

Nguyen, D., Ouala, S., Drumetz, L. and Fablet, R. (2021) *Variational Deep Learning for the Identification and Reconstruction of Chaotic and Stochastic Dynamical Systems from Noisy and Partial Observations*.

O'Gorman, P.A. and Dwyer, J.G. (2018) Using machine learning to parameterize moist convection: potential for modeling of climate, climate change, and extreme events. *Journal of Advances in Modeling Earth Systems*, 10(10), 2548–2563. https://doi.org/10.1029/2018MS001351.

Oseledets, V. (1968) A multiplicative ergodic theorem. Lyapunov characteristic numbers for dynamical systems. *Transactions of the Moscow Mathematical Society*, 19, 197–231.

Ott, E. (2002) *Chaos in dynamical systems*, 2nd edition. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511803260.

Palatella, L., Carrassi, A. and Trevisan, A. (2013) Lyapunov vectors and assimilation in the unstable subspace: theory and applications. *Journal of Physics A: Mathematical and Theoretical*, 46(25), 254020.

Palmer, T.N. (1996) Predictability of the atmosphere and oceans: from days to decades. In: Anderson, D.L.T. and Willebrand, J. (Eds.) *Decadal climate variability*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 83–155.

Patel, D., Canaday, D., Girvan, M., Pomerance, A. and Ott, E. (2021) Using machine learning to predict statistical properties of non-stationary dynamical processes: System climate, regime transitions, and the effect of stochasticity. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(3), 033149. https://doi.org/10.1063/5.0042598.

Pathak, J., Hunt, B., Girvan, M., Lu, Z. and Ott, E. (2018) Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach. *Physical Review Letters*, 120(2), 024102. https://doi.org/10.1103/PhysRevLett.120.024102.

Pathak, J., Lu, Z., Hunt, B.R., Girvan, M. and 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. https://doi.org/10.1063/1.5010300.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011) Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Pikovsky, A. and Politi, A. (2016) *Lyapunov exponents*. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9781139343473.

Quinn, C., Harries, D. and O'Kane, T.J. (2021) Dynamical analysis of a reduced model for the North Atlantic oscillation. *Journal of the Atmospheric Sciences*, 78(5), 1647–1671. https://doi.org/10.1175/JAS-D-20-0282.1.

Quinn, C., O'Kane, T.J. and Harries, D. (2022) Systematic calculation of finite-time mixed singular vectors and characterization of error growth for persistent coherent atmospheric disturbances over Eurasia. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(2), 023126. https://doi.org/10.1063/5.0066150.

Quinn, C., O'Kane, T.J. and Kitsios, V. (2020) Application of a local attractor dimension to reduced space strongly coupled data assimilation for chaotic multiscale systems. *Nonlinear Processes in Geophysics*, 27(1), 51–74. https://doi.org/10.5194/npg-27-51-2020.

Rasp, S. (2020) Coupled online learning as a way to tackle instabilities and biases in neural network parameterizations: general algorithms and Lorenz 96 case study (v1.0). *Geoscientific Model Development*, 13(5), 2185–2196. https://doi.org/10.5194/gmd-13-2185-2020.

Rasp, S., Dueben, P.D., Scher, S., Weyn, J.A., Mouatadid, S. and Thuerey, N. (2020) WeatherBench: a benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, 12, 11. https://doi.org/10.1029/2020MS002203.

Rasp, S., Pritchard, M.S. and Gentine, P. (2018) Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, 115(39), 9684–9689. https://doi.org/10.1073/pnas.1810286115.

Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N. and Prabhat. (2019) Deep learning and process understanding for data-driven Earth system science. *Nature*, 566(7743), 195–204. https://doi.org/10.1038/s41586-019-0912-1.

Rössler, O. (1976) An equation for continuous chaos. *Physics Letters A*, 57(5), 397–398. https://doi.org/10.1016/0375-9601(76)90101-8.

Ruelle, D. (1979) Ergodic theory of differentiable dynamical systems". en. *Publications Mathématiques de l'IHÉS*, 50, 27–58.

Sandri, M. (1996) Numerical calculation of Lyapunov exponents. *Mathematica Journal*, 6(3), 78–84.

Schultz, M.G., Betancourt, C., Gong, B., Kleinert, F., Langguth, M., Leufen, L.H., Mozaffari, A. and Stadtler, S. (2021) Can deep learning beat numerical weather prediction? *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194), 20200097. https://doi.org/10.1098/rsta.2020.0097.

Sinai, Y. (2009) Kolmogorov-Sinai entropy. *Scholarpedia*, 4(3), 2034. https://doi.org/10.4249/scholarpedia.2034.

Snoek, J., Larochelle, H. and Adams, R.P. (2012) "Practical Bayesian optimization of machine learning algorithms". *Proceedings of the 25th International Conference on Neural Information Processing Systems–Volume 2*. NIPS'12. Lake Tahoe, Nevada, pp. 2951–2959.

Sonnewald, M., Lguensat, R., Jones, D.C., Dueben, P.D., Brajard, J. and Balaji, V. (2021) Bridging observations, theory and numerical simulation of the ocean using machine learning. *Environmental Research Letters*, 16(7), 073008. https://doi.org/10.1088/1748-9326/ac0eb0.

Sparrow, C. (1982) *The Lorenz equations*. NY: Springer New York. https://doi.org/10.1007/978-1-4612-5767-7.

Strang, G. (2016) *Introduction to linear algebra*, 5th edition. Cambridge: Cambridge University Press.

Strogatz, S.H. (2018) *Nonlinear dynamics and chaos*. Boca Raton: CRC Press. https://doi.org/10.1201/9780429492563.

Tondeur, M., Carrassi, A., Vannitsem, S. and Bocquet, M. (2020) On temporal scale separation in coupled data assimilation with the ensemble Kalman filter. *Journal of Statistical Physics*, 179(5), 1161–1185. https://doi.org/10.1007/s10955-020-02525-z.

Toth, Z. and Kalnay, E. (1997) Ensemble forecasting at NCEP and the breeding method. *Monthly Weather Review*, 12(125), 3297–3319.

Uboldi, F. and Trevisan, A. (2015) Multiple-scale error growth in a convection-resolving model. *Nonlinear Processes in Geophysics*, 22(1), 1–13. https://doi.org/10.5194/npg-22-1-2015.

Vannitsem, S. (2017) Predictability of large-scale atmospheric motions: Lyapunov exponents and error dynamics. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(3), 032101. https://doi.org/10.1063/1.4979042.

Vannitsem, S. and Duan, W. (2020) On the use of near-neutral Backward Lyapunov Vectors to get reliable ensemble forecasts in coupled ocean–atmosphere systems. *Climate Dynamics*, 55, 1125–1139.

Vannitsem, S. and Lucarini, V. (2016) Statistical and dynamical properties of covariant Lyapunov vectors in a coupled atmosphere-ocean modelmultiscale effects, geometric degeneracy, and error dynamics. *Journal of Physics A: Mathematical and Theoretical*, 49(22), 224001. https://doi.org/10.1088/1751-8113/49/22/224001.

Vecchi, E., Pospíil, L., Albrecht, S., O'Kane, T.J. and Horenko, I. (2022) eSPA+: scalable entropy-optimal machine learning classification for small data problems. *Neural Computation*, 34(5), 1220–1255. https://doi.org/10.1162/neco_a_01490.

Wang, H. and Yeung, D.-Y. (2016) Towards Bayesian deep learning: a framework and some existing methods. *IEEE Transactions on Knowledge and Data Engineering*, 28(12), 3395–3408. https://doi.org/10.1109/TKDE.2016.2606428.

Wilk, M.B. and Gnanadesikan, R. (1968) Probability plotting methods forthe analysis forthe analysis of data. *Biometrika*, 55(1), 1–17. https://doi.org/10.1093/biomet/55.1.1.

Wolf, A., Swift, J.B., Swinney, H.L. and Vastano, J.A. (1985) Determining Lyapunov exponents from a time series. *Physica D: Nonlinear Phenomena*, 16(3), 285–317. https://doi.org/10.1016/0167-2789(85)90011-9.

Wolfe, C.L. and Samelson, R.M. (2007) An efficient method for recovering Lyapunov vectors from singular vectors. *Tellus A*, 59(3), 355–366. https://doi.org/10.1111/j.1600-0870.2007.00234.x.

Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ø., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M. and Dean, J. (2016). Google's neural machine translation system: bridging the gap between human and machine translation. *CoRR*, abs/1609.08144. https://doi.org/10.48550/arXiv.1609.08144.

Zoph, B., Vasudevan, V., Shlens, J. and Le, Q.V. (2018) "Learning transferable architectures for scalable image recognition". *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710. https://doi.org/10.1109/CVPR.2018.00907.

How to cite this article: Ayers, D., Lau, J., Amezcua, J., Carrassi, A. & Ojha, V. (2023) Supervised machine learning to estimate instabilities in chaotic systems: Estimation of local Lyapunov exponents. *Quarterly Journal of the Royal Meteorological Society*, 1–27. Available from: https://doi.org/10.1002/qj.4450