Neural Networks 142 (2021) 252–268

Contents lists available at ScienceDirect

Neural Networks

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

Robust Optimization and Validation of Echo State Networks for learning chaotic dynamics

Alberto Racca [a], Luca Magri [a,b,c,d,*]

[a] Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK
[b] Aeronautics Department, Imperial College London, Exhibition Rd, London, SW7 2AZ, UK
[c] The Alan Turing Institute, 96 Euston Road, London, England, NW1 2DB, UK
[d] Institute for Advanced Study, Technical University of Munich, Lichtenbergstrasse 2a, 85748 Garching, Germany[1]

ARTICLE INFO

Article history:
Received 9 February 2021
Received in revised form 3 May 2021
Accepted 6 May 2021
Available online 14 May 2021

Keywords:
Chaotic dynamical systems
Reservoir Computing
Robustness

ABSTRACT

An approach to the time-accurate prediction of chaotic solutions is by learning temporal patterns from data. Echo State Networks (ESNs), which are a class of Reservoir Computing, can accurately predict the chaotic dynamics well beyond the predictability time. Existing studies, however, also showed that small changes in the hyperparameters may markedly affect the network's performance. The overarching aim of this paper is to improve the robustness in the selection of hyperparameters in Echo State Networks for the time-accurate prediction of chaotic solutions. We define the *robustness* of a validation strategy as its ability to select hyperparameters that perform consistently between validation and test sets. The goal is three-fold. First, we investigate routinely used validation strategies. Second, we propose the *Recycle Validation*, and the *chaotic versions* of existing validation strategies, to specifically tackle the forecasting of chaotic systems. Third, we compare Bayesian optimization with the traditional grid search for optimal hyperparameter selection. Numerical tests are performed on prototypical nonlinear systems that have chaotic and quasiperiodic solutions, such as the Lorenz and Lorenz-96 systems, and the Kuznetsov oscillator. Both model-free and model-informed Echo State Networks are analysed. By comparing the networks' performance in learning chaotic (unpredictable) versus quasiperiodic (predictable) solutions, we highlight fundamental challenges in learning chaotic solutions. The proposed validation strategies, which are based on the dynamical systems properties of chaotic time series, are shown to outperform the state-of-the-art validation strategies. Because the strategies are principled – they are based on chaos theory such as the Lyapunov time – they can be applied to other Recurrent Neural Networks architectures with little modification. This work opens up new possibilities for the robust design and application of Echo State Networks, and Recurrent Neural Networks, to the time-accurate prediction of chaotic systems.

1. Introduction

Chaotic systems naturally appear in many branches of science and engineering, from turbulent flows (e.g., Bec et al., 2006; Boffetta, Cencini, Falcioni, & Vulpiani, 2002; Deissler, 1986), through vibrations (Moon & Shaw, 1983), electronics and telecommunications (Kennedy, Rovatti, & Setti, 2000), quantum mechanics (Stöckmann, 2000), reacting flows (Hassanaly & Raman, 2019; Nastac, Labahn, Magri, & Ihme, 2017), to epidemic modelling (Bolker & Grenfell, 1993), to name only a few. The time-accurate computation of chaotic systems is hindered by the "butterfly effect" (Lorenz, 1963): an error in the system's knowledge – e.g., initial conditions and parameters – grows exponentially until nonlinear saturation. Practically, it is not possible to time-accurately predict chaotic solutions after a time scale, known as the predictability time. The predictability time scales with the inverse of the dominant Lyapunov exponent, which is typically a small characteristic scale of the system under investigation (Boffetta et al., 2002).

An approach to the prediction of chaotic dynamics is data-driven. Given a time series (data), we wish to learn the underlying chaotic dynamics to predict the future evolution. The data-driven approach, also known as model-free, traces back to the delay coordinate embedding by Takens (1981), which is widely used in time series analysis, in particular, in low-dimensional systems (Guckenheimer & Holmes, 2013). An alternative data-driven approach to inferring (or, equivalently, learning) chaotic dynamics from data is machine learning. Machine learning is establishing itself as a paradigm that is complementary to first-principles

* Corresponding author at: Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK.
E-mail address: lm547@cam.ac.uk (L. Magri).

[1] Visiting fellowship.

https://doi.org/10.1016/j.neunet.2021.05.004
0893-6080/

modelling of nonlinear systems in computational science and engineering (Baker et al., 2019). In the realm of neural networks, which is the focus of this paper, the feed-forward neural network is the archetypical architecture, which may excel at classification and regression problems (Goodfellow, Bengio, & Courville, 2016). The feed-forward neural network, however, is not the optimal architecture for chaotic time series forecasting because it is not designed to learn temporal correlations. Specifically, in time series forecasting, inputs and outputs are ordered sequentially, in other words, they are temporally correlated. To overcome the limitations of feed-forward neural networks, Recurrent Neural Networks (RNNs) (Rumelhart, Hinton, & Williams, 1986) have been designed to learn temporal correlations. Examples of successful applications span from speech recognition (Sak, Senior, & Beaufays, 2014), through language translation (Sutskever, Vinyals, & Le, 2014), fluids (Brunton, Noack, & Koumoutsakos, 2020; Doan, Polifke, & Magri, 2021; Nakai & Saiki, 2018; Vlachas, Byeon, Wan, Sapsis, & Koumoutsakos, 2018; Wan & Sapsis, 2018), to thermo-acoustic oscillations (Huhn & Magri, 2020), among many others. RNNs take into account the sequential nature of the inputs by updating a hidden time-varying state through an internal loop. As a result of the long-lasting time dependencies of the hidden state, however, training RNNs with Back Propagation Through Time (Werbos, 1990) is notoriously difficult. This is because the repeated backwards multiplication of intermediate gradients causes the final gradient to either vanish or become unbounded depending on the spectral radius of the gradient matrix (Bengio, Simard, & Frasconi, 1994; Werbos, 1988). This makes the training ill-posed, which may negatively affect the computation of the optimal set of weights. To overcome this problem, two main types of RNN architectures have been proposed: Gated Structures and Reservoir Computing. Gated Structures prevent gradients from vanishing or becoming unbounded by regularizing the passage of information inside the network, as accomplished in architectures such as Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997) and Gated Recurrent Units (GRU) networks (Cho, van Merriënboer, Gulcehre, Bahdanau, Bougares, Schwenk, & Bengio, 2014). Alternatively, in Reservoir Computing (RC) (Jaeger & Haas, 2004; Maass, Natschläger, & Markram, 2002), a high-dimensional dynamical system, the reservoir, acts both as a nonlinear expansion of the inputs and as the memory of the system (Lukoševičius, 2012). At each time step, the output is computed as a linear combination of the reservoir state's components, the weights of which are the only trainable parameters of the machine. Training is, therefore, reduced to a linear regression problem, which bypasses the issue of repeated gradients multiplication in RNNs.

In chaotic attractors, Reservoir Computing has been employed to achieve at least four different goals: to (i) learn ergodic properties, such as Lyapunov exponents (Lu, Hunt, & Ott, 2018; Pathak, Lu, Hunt, Girvan, & Ott, 2017) and statistics (Huhn & Magri, 2020; Lu et al., 2018); (ii) filter out noise to recover the deterministic dynamics (Doan, Polifke, & Magri, 2020b), (iii) reconstruct unmeasured (hidden) variables (Doan, Polifke, & Magri, 2020a; Lu et al., 2017; Racca & Magri, 2021) and (iv) time-accurately predict the dynamics (Doan, Polifke, & Magri, 2019; Pathak, Hunt, Girvan, Lu, & Ott, 2018; Wikner et al., 2020). In this work, we focus on the time-accurate short term prediction of chaotic attractors. A successful Reservoir Computing architecture is the Echo State Network (ESN) (Jaeger & Haas, 2004), which is a universal approximator (Gonon & Ortega, 2021; Grigoryeva & Ortega, 2018) suitable for the prediction of chaotic time series (Pathak, Hunt, Girvan, Lu, & Ott, 2018). There are two broad categories of Echo State Networks: model-free (Lukoševičius, 2012; Pathak, Hunt, Girvan, Lu, & Ott, 2018) and model-informed (Doan et al., 2019; Pathak et al., 2018). On the one hand, in model-free ESNs, which are the original networks, the training is performed on data only (Lukoševičius, 2012). On the other hand, in model-informed ESNs, the governing equations, or a reduced-order form of them, are embedded in the architecture, for example, in the reservoir in hybrid ESNs (Pathak et al., 2018), or in the loss function in physics-informed ESNs (Doan et al., 2019) . In chaotic time series forecasting, model-informed ESNs typically outperform model-free ESNs (Doan et al., 2019; Pathak et al., 2018; Wikner et al., 2020). Both model-free and model-informed Echo State Networks perform as well as LSTMs and GRUs, requiring less computational resources for training (Chattopadhyay, Hassanzadeh, & Subramanian, 2020; Vlachas et al., 2020). However, there are two major challenges when using Echo State Networks. First, a random initialization is required to create the reservoir (Lukoševičius, 2012). Networks with different initializations may perform substantially differently, even after hyperparameter tuning (Haluszczynski & Räth, 2019), thus, network testing through an ensemble of network realizations is required. Secondly, there is high hyperparameter sensitivity (Jiang & Lai, 2019; Lukoševičius, 2012). The most common validation strategy to compute the hyperparameters for learning chaotic dynamics is the Single Shot Validation, which minimizes the error in an interval subsequent to the training interval. Other validation strategies have been investigated, such as the Walk Forward Validation and the K-Fold cross Validation (Lukoševičius & Uselis, 2019), but the study was restricted to non-chaotic systems. The computation of the optimal set of hyperparameters is typically performed by Grid Search (Doan et al., 2021; Jaeger & Haas, 2004; Pathak, Hunt, Girvan, Lu, & Ott, 2018; Pathak et al., 2018), although other optimization strategies such as Evolutionary Algorithms (Ferreira, Ludermir, & De Aquino, 2013; Ishu, van der Zant, Becanovic, & Ploger, 2004), Stochastic Gradient Descent (Thiede & Parlitz, 2019), Particle Swarm Optimization (Wang & Yan, 2015) and Bayesian Optimization (Yperman & Becker, 2016) have been proposed. In particular, Bayesian Optimization (BO) has proved to improve the performance of reservoir-computing architectures in the prediction of chaotic time series, outperforming the commonly used Grid Search strategy (Griffith, Pomerance, & Gauthier, 2019). Bayesian Optimization is a gradient-free search strategy, thereby, it is less sensitive to local minima with respect to gradient descent methods (Thiede & Parlitz, 2019; Yperman & Becker, 2016). Moreover, Bayesian Optimization is based on Gaussian Process (GP) regression (Rasmussen, 2003), therefore, it naturally quantifies the uncertainty on the computation.

This paper studies the robust selection of hyperparameters in Echo State Networks for the time-accurate prediction of chaotic attractors, with a focus on the performance of the hyperparameters between validation and test sets across different network realizations. We define the *robustness* of a validation strategy as its ability to select hyperparameters that perform consistently between validation and test sets. The objective of this work is three-fold. First, we investigate the robustness of the Single Shot Validation, Walk Forward Validation and the K-Fold cross Validation. Second, we propose the Recycle Validation and the chaotic version of existing validation strategies to specifically tackle the forecasting of chaotic systems. Third, we compare Bayesian optimization with grid search for optimal hyperparameter selection. The Lorenz system (Lorenz, 1963), the Lorenz-96 model (Lorenz, 1996) and the Kutznetsov oscillator (Kuznetsov, Kuznetsov, & Stankevich, 2010) are considered as prototypical nonlinear deterministic systems. We highlight fundamental challenges in the robustness of ESNs for chaotic solutions with a comparative investigation on quasiperiodic oscillations. Both model-free and model-informed architectures are analysed.

The paper is organized as follows. Section 2 presents the model-free and model-informed Echo State Network architectures. Section 3 describes the validation strategies. Section 4

investigates the robustness of the Single Shot Validation in forecasting chaotic time series. Section 5 analyses the new validation strategies to improve the robustness in forecasting chaotic time series. Section 6 investigates the robustness of the validation strategies in forecasting quasiperiodic time series. Finally, we summarize the results of this study and discuss future work in the conclusions (Section 7).

2. Echo state networks

As shown in Fig. 1, in the Echo State Network, at any time t_i the input vector, $\mathbf{u}_{\text{in}}(t_i) \in \mathbb{R}^{N_u}$, is mapped into the reservoir state, by the input matrix, $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_r \times N_u}$, where $N_r \gg N_u$. The reservoir state, $\mathbf{r}(t_i) \in \mathbb{R}^{N_r}$, is updated at each time iteration as a function of the current input and its previous value

$$\mathbf{r}(t_{i+1}) = \tanh\left(\mathbf{W}_{\text{in}}\mathbf{u}_{\text{in}}(t_i) + \mathbf{W}\mathbf{r}(t_i)\right), \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{N_r \times N_r}$ is the state matrix. The predicted output, $\mathbf{u}_{\text{p}}(t_{i+1}) \in \mathbb{R}^{N_u}$, is obtained as

$$\mathbf{u}_{\text{p}}(t_{i+1}) = \hat{\mathbf{r}}(t_{i+1})^T\mathbf{W}_{\text{out}}, \qquad \hat{\mathbf{r}}(t_{i+1}) = \mathbf{g}(\mathbf{r}(t_{i+1})); \tag{2}$$

where $\mathbf{g}(\cdot)$ is a nonlinear transformation, $\hat{\mathbf{r}}(t_{i+1}) \in \mathbb{R}^{N_{\hat{r}}}$ is the updated reservoir state, and $\mathbf{W}_{\text{out}} \in \mathbb{R}^{N_{\hat{r}} \times N_u}$ is the output matrix. The input matrix, $\mathbf{W}_{\text{in}}$, and state matrix, $\mathbf{W}$, are (pseudo)randomly generated and fixed, while the weights of the output matrix, $\mathbf{W}_{\text{out}}$, are computed by training the network. In this work, the input matrix, $\mathbf{W}_{\text{in}}$, has only one element different from zero per row, which is sampled from a uniform distribution in $[-\sigma_{\text{in}}, \sigma_{\text{in}}]$, where σ_{in} is the input scaling. The state matrix, $\mathbf{W}$, is an Erdős–Renyi matrix with average connectivity d, in which each neuron (each row of $\mathbf{W}$) has on average only d connections (non-zero elements). The non-zero elements are obtained by sampling from a uniform distribution in $[-1, 1]$; the entire matrix is then rescaled by a multiplication factor to set the spectral radius, ρ. The spectral radius is key to enforcing the *echo state property*. (In a network with the echo state property, the state loses its dependence on its previous values for sufficiently large times and, therefore, it is uniquely defined by the sequence of inputs.) While the echo state property may hold for a wider range of spectral radii (Yildiz, Jaeger, & Kiebel, 2012), the condition $\rho < 1$ is typically chosen (Lukoševičius, 2012).

The ESN can be run either in open-loop or closed-loop configuration. In the open-loop configuration, first, we feed data as the input at each time step to compute and store $\hat{\mathbf{r}}(t_i)$ (1)–(2). In the initial transient of this process, the washout interval, we do not compute the output, $\mathbf{u}_{\text{p}}(t_i)$. The purpose of the washout interval is for the reservoir state to satisfy the echo state property, thereby becoming independent of the arbitrarily chosen initial reservoir state, $\mathbf{r}(t_0) = 0$. Secondly, we train the output matrix, $\mathbf{W}_{\text{out}}$, by minimizing the Mean Square Error (MSE) between the outputs, $\mathbf{u}_{\text{p}}(t_i)$, and the data, $\mathbf{u}_{\text{d}}(t_i)$, over a training set of N_{tr} points

$$\text{MSE} \triangleq \frac{1}{N_{\text{tr}}N_u}\sum_{i=1}^{N_{\text{tr}}} \|\mathbf{u}_{\text{p}}(t_i) - \mathbf{u}_{\text{d}}(t_i)\|^2, \tag{3}$$

where $\|\cdot\|$ is the L_2 norm. Minimizing (3) is a least-squares minimization problem, which can be solved as a linear system through ridge regression

$$(\mathbf{R}\mathbf{R}^T + \beta\mathbf{I})\mathbf{W}_{\text{out}} = \mathbf{R}\mathbf{U}_{\text{d}}^T, \tag{4}$$

where $\mathbf{R} \in \mathbb{R}^{N_{\hat{r}} \times N_{\text{tr}}}$ and $\mathbf{U}_{\text{d}} \in \mathbb{R}^{N_u \times N_{\text{tr}}}$ are the horizontal concatenation of the updated reservoir states, $\hat{\mathbf{r}}(t_i)$, and the data, $\mathbf{u}_{\text{d}}(t_i)$, respectively; $\mathbf{I}$ is the identity matrix and β is the user-defined Tikhonov regularization parameter (Tikhonov, Goncharsky, Stepanov, & Yagola, 2013). We solve the linear system through the `linalg.solve` function in `numpy` (Harris et al., 2020). In the closed-loop configuration, starting from an initial data point as an input and an initial reservoir state obtained after the washout interval, the output, $\mathbf{u}_{\text{p}}(t_i)$, is fed back to the network as an input for the next time step prediction. In doing so, the network is able to autonomously evolve in the future. The closed-loop configuration is used during validation and testing.

2.1. Model-free and model-informed architectures

We consider model-free and model-informed architectures (Fig. 1). The basic model-free ESN is obtained by setting $\mathbf{g}(\mathbf{r}(t_i)) = \mathbf{r}(t_i)$. This architecture, however, generates symmetric solutions in the closed-loop configuration (Huhn & Magri, 2020; Lu et al., 2017), which can cause the predicted trajectory to stray away from the actual attractor towards a symmetric attractor, which is not a solution of the dynamical system (but it is a solution of the network). To break the symmetry, we add biases in the input and output layers

$$\begin{aligned}&\mathbf{r}(t_{i+1}) = \tanh\left(\mathbf{W}_{\text{in}}[\mathbf{u}_{\text{in}}(t_i); b_{\text{in}}] + \mathbf{W}\mathbf{r}(t_i)\right), \quad \hat{\mathbf{r}}(t_{i+1}) = [\mathbf{r}(t_{i+1}); 1],\\ &\mathbf{u}_{\text{p}}(t_{i+1}) = \hat{\mathbf{r}}(t_{i+1})^T\mathbf{W}_{\text{out}};\end{aligned} \tag{5}$$

where $[\cdot;\cdot]$ indicates vertical concatenation, b_{in} is the scalar input bias and $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_r \times (N_u+1)}$. In the model-informed ESN, also known as hybrid as proposed by Pathak et al. (2018), information about the governing equations (model knowledge) is embedded into the model through a function of the input, $\mathcal{K}(\mathbf{u}_{\text{in}}(t_i))$, which, for example, may be a reduced order model that provides information about the output at the next time step as

$$\hat{\mathbf{r}}(t_{i+1}) = [\mathbf{r}(t_{i+1}); 1; \mathcal{K}(\mathbf{u}_{\text{in}}(t_i))]. \tag{6}$$

In this work, we use $\mathcal{K}(\mathbf{u}_{\text{in}}(t_i))$ only to update the reservoir state (Wikner et al., 2020), in order to use the same input matrix, $\mathbf{W}_{\text{in}}$, and state matrix, $\mathbf{W}$, of the model-free architecture. This allows us to directly compare the performances of the model-free and model-informed architectures.

3. Validation

The purpose of the validation is to determine the hyperparameters by minimizing an error. We make a distinction between the hyperparameters (i) that require re-initialization of $\mathbf{W}_{\text{in}}$ and $\mathbf{W}$, and (ii) that do not require re-initialization. The size of the reservoir, N_r, and connectivity, d, require re-initialization, whereas the input scaling, σ_{in}, the spectral radius, ρ, the Tikhonov parameter, β, and the input bias, b_{in}, do not. The fundamental difference between (i) and (ii) is that the random component of the re-initialization of $\mathbf{W}_{\text{in}}$ and $\mathbf{W}$ makes the objective function to be optimized random, which significantly increases the complexity of the optimization. In this study, we minimize the error with respect to the input scaling, σ_{in}, and spectral radius, ρ, which are key hyperparameters for the performance of the network (Jiang & Lai, 2019; Lukoševičius, 2012). For convenience, we rewrite the reservoir state equation (1) as

$$\mathbf{r}(t_{i+1}) = \tanh(\sigma_{\text{in}}\hat{\mathbf{W}}_{\text{in}}[\mathbf{u}_{\text{in}}(t_i); b_{\text{in}}] + \rho\hat{\mathbf{W}}\mathbf{r}(t_i)), \tag{7}$$

where the non-zero elements of $\hat{\mathbf{W}}_{\text{in}}$ are sampled from the uniform distribution in $[-1, 1]$ and $\hat{\mathbf{W}}$ has been scaled to have a unitary spectral radius.

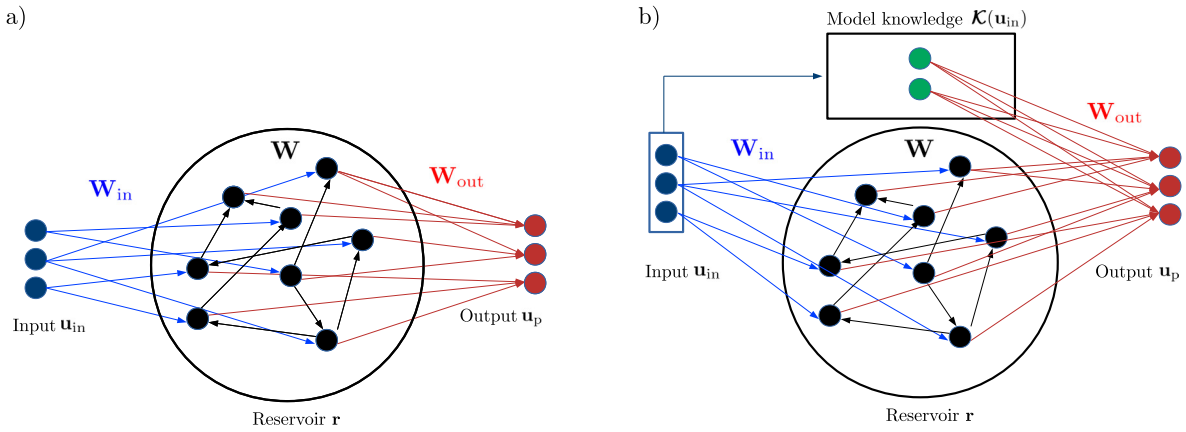

Fig. 1. Schematic representation of (a) model-free and (b) model-informed Echo State Networks (ESNs).

3.1. Performance metrics

We determine the hyperparameters by minimizing the Mean Squared Error (3) in the validation interval of fixed length. The networks are tested on multiple starting points along the attractor by using both the Mean Squared Error and Prediction Horizon (PH), the latter of which is defined as the time interval during which the normalized error is smaller than a user-defined threshold k (Doan et al., 2019; Pathak et al., 2018)

$$\frac{\|\mathbf{u}_{\text{p}}(t_i) - \mathbf{u}_{\text{d}}(t_i)\|}{\sqrt{\frac{1}{N_{\text{PH}}}\sum_{j=1}^{N_{\text{PH}}} \|\mathbf{u}_{\text{d}}(t_j)\|^2}} < k, \tag{8}$$

where N_{PH} are the number of timesteps in the Prediction Horizon, and $k = 0.2$ unless otherwise specified. The Mean Squared Error and Prediction Horizon for the same starting point in the attractor are strictly correlated (see Supplementary Materials S.1). We use the Mean Squared Error to partition the dataset in intervals of fixed length during validation, while we use the Prediction Horizon in the test set because it is the most physical quantity when assessing the time-accurate prediction of chaotic systems (e.g., Doan et al., 2020b; Pathak, Hunt, Girvan, Lu, & Ott, 2018).

3.2. Strategies

The most common validation strategy for ESNs is the Single Shot Validation (SSV), which splits the available data in a training set and a single subsequent validation set (Fig. 2a). The time interval of the validation set, during which the hyperparameters are tuned, is small and represents only a fraction of the attractor. In nonlinear time series prediction, the choice of the validation strategy has to take into account (i) the intervals we are interested in predicting and (ii) the nature of the signal we are trying to reproduce. Here, we are interested in predicting multiple intervals as the trajectory spans the attractor, rather than a specific interval starting from a specific initial condition. Moreover, the trajectory that spans the attractor has no underlying time-varying statistics, e.g there is no time-dependency of the mean of the signal, hence trajectories return indefinitely in nearby regions of the attractor (or, more technically, chaotic dynamics is topologically mixing) (e.g., Guckenheimer & Holmes, 2013). This implies that we can obtain information regarding the intervals that we are interested in predicting from any interval of the trajectory that constitutes our dataset, regardless of the interval position in time within the dataset. This means that (i) all the parts of the dataset are equally important in determining the hyperparameters and (ii) the validation should be performed on the entire dataset and not only on the last portion of it. For this reason, as shown in Section 4, the Single Shot Validation strategy is not suited for chaotic time series prediction.

These observations lead us to use validation strategies based on multiple validation intervals, which may precede the training set, such as the Walk Forward Validation (WFV) and the K-Fold cross Validation (KFV). We also propose an ad-hoc validation strategy—the Recycle Validation (RV). The objective of these strategies is to tune the hyperparameters over an effectively larger portion of the trajectory, by minimizing the average of the objective function (error) over multiple validation intervals. The regular version of these strategies consists of creating subsequent folds by moving forward in time the validation set by its own length. Additionally, we propose the chaotic version, in which we move the fold forward in time by one Lyapunov Time (LT) (Fig. 2). The Lyapunov Time is a key time scale in chaotic dynamical systems, which is defined as the inverse of the leading Lyapunov exponent Λ of the system, which, in turn, is the exponential rate at which infinitesimally close trajectories, $\delta\mathbf{q}(0)$ diverge (e.g., Boffetta et al., 2002)

$$\|\delta\mathbf{q}(t)\| \sim \|\delta\mathbf{q}(0)\| \exp(\Lambda t) \qquad t \to \infty, \quad \|\delta\mathbf{q}(0)\| \to 0. \tag{9}$$

Walk Forward Validation. In the Walk Forward Validation (WFV) (Fig. 2b), we partition the available data in multiple splits, while maintaining sequentiality of the data. From a starting dataset of length n, the first m points ($m < n$) are taken as the first fold, with N_{tr} points for training and v points for validation ($v + N_{tr} = m$). These quantities must respect $(n - m) = (k_1 - 1)v$; $k_1 \in \mathbb{N}$. The remaining $(k_1 - 1)$ folds are generated by moving the training plus validation set forward in time by a number of points v. This way, the original dataset is partitioned in k_1 folds and the hyperparameters are selected to minimize the average MSE over the folds. For every set of hyperparameters and every fold the output matrix, $\mathbf{W}_{\text{out}}$, is recomputed.

K-Fold cross Validation. Although the K-Fold cross Validation (KFV) (Fig. 2c) is a common strategy in regression and classification, it is not commonly used in time series prediction because the validation and training intervals are not sequential to each other. This strategy partitions the available data in k_2 splits. Over the entire dataset of length n, after an initial bv points, with $0 \leq b < 1$, needed to have an integer number of splits, the remaining $n - bv$ points are used as k_2 validation intervals, each of length v. For each validation interval we define a different fold, in which we use all the remaining data points for training. We determine the hyperparameters by minimizing the average of the MSE between the folds. For every set of hyperparameters and every fold the output matrix, $\mathbf{W}_{\text{out}}$, is recomputed.

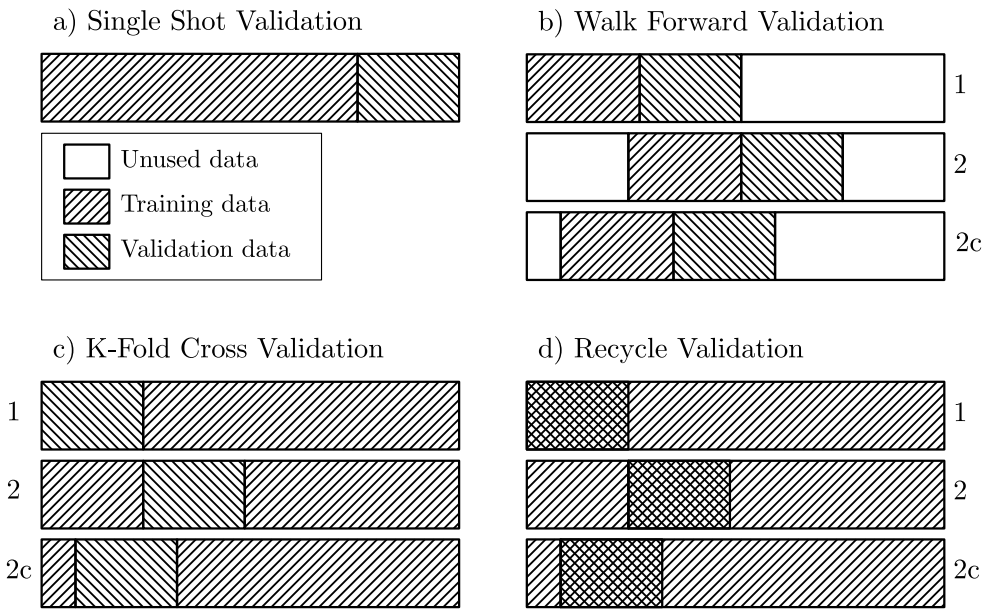

Fig. 2. Partition of the data in the different validation strategies. In (b–d), bar 1 shows the first fold, bar 2 shows the second fold, and bar 2c shows the second fold in the chaotic version (shifted by one Lyapunov time).

Recycle Validation. We propose the Recycle Validation (RV) (Fig. 2d), which exploits the information obtained by both open-loop and closed-loop configurations. Because the network works in two different configurations, it can obtain additional information when validating on data already used in training. To do so, first, we train $\mathbf{W}_{\text{out}}$ only once per set of hyperparameters using the entire dataset of n points. Second, we validate the network on k_2 splits of length v from data that has already been used to train the output weights. Each split is imposed by moving forward in time the previous validation interval by v points. After an initial bv points, with $0 \leq b < 1$, needed to have an integer number of splits, the remaining $n - bv$ points are used as k_2 validation intervals. We determine the hyperparameters by computing the average of the MSE between the splits. This strategy has four main advantages. First, it can be used in small datasets, where the partition of the dataset in separate training and validation sets may cause the other strategies to perform poorly. In small datasets, the validation intervals represent a larger percentage of the dataset since each validation interval needs to be multiple Lyapunov Times to capture the divergence of chaotic trajectories. Therefore, the training set becomes substantially smaller than the dataset and the output matrix used during validation differs substantially from the output matrix of the whole dataset. This results in a poor selection of hyperparameters. Second, for a given dataset, we maximize the number of validation splits, using the same validation intervals of the K-Fold cross Validation. Third, we tune the hyperparameters using the same output matrix, $\mathbf{W}_{\text{out}}$, that we use in the test set. Fourth, it has lower computational cost than the K-Fold cross Validation because it does not require retraining the output matrix for the different folds, which makes it computationally cheaper (Appendix A).

Chaotic version. The chaotic version consists of shifting the validation intervals forward in time, not by their own length, but by one Lyapunov Time when constructing the next fold. In doing so, different splits will overlap, but, since the trajectory related to the split that started one Lyapunov Time (LT) earlier has strayed away from the attractor on average by $e^{\Lambda \times 1\text{LT}} = e$, the two intervals contain different information. The purpose of this version is to further increase the number of intervals on which the network is validated. The regular and chaotic versions for each validation strategy are shown in frames (b–d) in Fig. 2 in bars 2 and 2c, respectively. The chaotic versions of the Walk Forward Validation, the K-fold cross Validation and the Recycle Validation are denoted by the subscript c.

Computing $\mathbf{W}_{\text{out}}$. In each validation strategy, for each pair of input scaling, σ_{in}, and spectral radius, ρ, we recompute the output matrix, $\mathbf{W}_{\text{out}}$. Moreover, for a pair of σ_{in} and ρ in each fold of the K-Fold Validation and Walk Forward Validation a different $\mathbf{W}_{\text{out}}$ is computed. Even with same hyperparameters the folds have different $\mathbf{W}_{\text{out}}$ because the training data is different. For each validation strategy, once $\mathbf{W}_{\text{out}}$ is determined in open-loop, the error that is minimized is that obtained by running the network in closed-loop in the validation interval(s). After training and validation are completed – i.e., we have selected the hyperparameters – the $\mathbf{W}_{\text{out}}$ to be used in the test set is computed on the entire dataset used for training plus validation using the optimal hyperparameters.

3.3. Bayesian optimization and grid search

To find the minimum of the Mean Squared Error (3) of the validation set in the hyperparameter space, we use Bayesian Optimization (BO), which is compared to Grid Search (GS). Bayesian Optimization has been shown to outperform other state-of-the-art optimization methods when the number of evaluations of an expensive objective function is limited (Brochu, Cora, & De Freitas, 2010; Snoek, Larochelle, & Adams, 2012). It is a global search method, which is able to incorporate prior knowledge about the objective function and to use information from the entire search space. It treats the objective function as a black box, therefore, it does not require gradient information. Starting from an initial N_{st} evaluations of the objective function, BO performs a Gaussian Process (GP) regression (Rasmussen, 2003) to reconstruct the function in the search space, using function evaluations as data. Once the GP fitting is available, we select the new point at which to evaluate the objective function so that the new point maximizes the acquisition function. The acquisition function is evaluated on the mean and standard deviation of the GP reconstruction. After the objective function is evaluated at a new point, the enlarged data set, comprising of the new point, is used to perform another GP regression, select a new point and so

on and forth. In this work, we use the gp-hedge Bayesian Optimization algorithm implemented in `scikit-optimize` library in Python (Hoffman, Brochu, & de Freitas, 2011; Virtanen & et al., 2020). The details of the formulation are explained in Appendix B and in the Supplementary Material S.5.

4. Robustness of the single shot validation

As a prototypical chaotic system, we investigate the Lorenz system (Lorenz, 1963), which is a reduced-order model of Rayleigh–Bénard convection

$$\begin{aligned}\dot{x} &= \sigma_L(y - x)\\ \dot{y} &= x(\rho_L - z) - y\\ \dot{z} &= xy - \beta_L z,\end{aligned} \tag{10}$$

where $[\sigma_L, \beta_L, \rho_L] = [10, 8/3, 28]$ is selected to generate chaotic solutions. The system is integrated with a forward Euler scheme with step $dt = 0.009$ LT. The Lyapunov Time is LT $= \Lambda^{-1} \approx 1.1$ (Viswanath, 1998).

We analyse the performance of the Single Shot Validation (SSV), which is employed for training (1 LT to 9 LTs), validation (9 LTs to 12 LTs), and testing (12 LTs to 15 LTs), as shown in Fig. 3. The input, $\mathbf{u}_{\text{in}}(t_i)$, is normalized by its maximum variation. (This is done because we are using a single scalar quantity σ_{in} to scale all the components of the input.) The network has a fixed number of neurons, $N_r = 100$, connectivity, $d = 3$, Tikhonov parameter, $\beta_t = 10^{-11}$ and input bias, $b_{\text{in}} = 1$ (Lukoševičius, 2012). The bias, b_{in}, is set for it to have the same order of magnitude of the normalized input. The input scaling, σ_{in}, and spectral radius, ρ, are tuned during validation in the range $[0.5, 5] \times [0.1, 1]$ to minimize the $\log_{10}$(MSE). The range of the spectral radius, ρ, is selected for the network to respect the echo state property, whereas the range of the input scaling, σ_{in}, is selected to normalize the inputs. The optimization is performed with (i) Grid Search (GS) consisting of 7×7 points, and (ii) Bayesian Optimization (BO) consisting of 5×5 starting points and 24 points acquired by the gp-hedge algorithm. The two optimization schemes are applied to an ensemble of $N_{\text{ens}} = 50$ networks, which differ by the random initialization of the input matrix, $\mathbf{W}_{\text{in}}$, and the state matrix, $\mathbf{W}$. N_{ens} is selected after a test on statistical convergence (more details in Supplementary Material S.2).

Fig. 4 shows the performance of the optimal hyperparameters computed by Grid Search and Bayesian Optimization for the ensemble members. First, we analyse the performance in validation (panel (a)). As shown by the medians reported in the caption, Bayesian optimization markedly outperforms Grid Search. Second, we analyse the performance in the test set (panel (b)). The performance of each network is assessed by computing the MSE in the test set for the hyperparameters found in the validation set. For this, the output matrix, $\mathbf{W}_{\text{out}}$, of the test set is obtained by retraining over both the training and validation sets. There is a significant difference between the performance of the ensemble in the validation set (a) and test set (b). This means that the Single Shot Validation is not *robust*, i.e., it is not able to select hyperparameters that perform consistently between validation and test set. This marginal robustness causes the overall performance of the networks and the benefit of using Bayesian Optimization to be markedly reduced. This is a signature of chaos, whose unpredictability results in a weak correlation between validation and test sets. We further verify the low correlation between the sets by computing the mean of the Gaussian process reconstruction from a 30×30 grid of $\log_{10}$(MSE) for a representative network of the ensemble (Fig. 5). The performance of the hyperparameters can deteriorate by four, or more, orders of magnitude from the validation set to the test set (panel (c)).

To assess quantitatively the correlation of the optimal hyperparameters' performance between the validation and test sets, we use the Spearman coefficient (Spearman, 1904)

$$\tilde{r}_S(\mathbf{x}, \mathbf{y}) = \frac{\sum_{j=1}^{2N_{\text{ens}}}(z(x)_j - N_{\text{ens}})(z(y)_j - N_{\text{ens}})}{\sqrt{\sum_{j=1}^{2N_{\text{ens}}}(z(x)_j - N_{\text{ens}})^2}\sqrt{\sum_{j=1}^{2N_{\text{ens}}}(z(y)_j - N_{\text{ens}})^2}},$$
$$\mathbf{x} = \begin{bmatrix} \mathbf{m}_{\text{Val}}^{(\text{BO})} \\ \mathbf{m}_{\text{Val}}^{(\text{GS})} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} \mathbf{m}_{\text{Test}}^{(\text{BO})} \\ \mathbf{m}_{\text{Test}}^{(\text{GS})} \end{bmatrix}, \tag{11}$$

where $\mathbf{z}(\mathbf{x})$ is the ranking function; $\mathbf{m} \in \mathbb{R}^{N_{\text{ens}}}$ contains the MSE for the optimal hyperparameters in validation (subscript Val), or test (subscript Test) obtained by Bayesian Optimization (superscript BO), or Grid Search (superscript GS). $\tilde{r}_S$ quantifies the correlation between the MSE of the optimal hyperparameters, obtained during validation by both Bayesian Optimization and Grid Search, and the MSE for the same hyperparameters in the test set over the ensemble. The values $\tilde{r}_S = \{-1, 0, 1\}$ indicate anticorrelation, no correlation and correlation, respectively.

Fig. 6 shows the correlation analysis. The scatter plot for $\mathbf{x}$ and $\mathbf{y}$ (panel (a)) shows that the MSE of the optimal hyperparameters in the validation and test sets are weakly correlated with $\tilde{r}_S = 0.32$. Panels (b, c) show the values of the optimal hyperparameters, which vary substantially from one network realization to another.

4.1. Remarks

First, because the MSE and optimal hyperparameters vary significantly in different network realizations, we advise performing optimization separately for each network to increase the performance (as further verified in Appendix C). Second, hyperparameters that are optimal in the validation set may have a poor performance in the test set, which may greatly reduce the benefit of using Bayesian Optimization to select the hyperparameters. This highlights a fundamental challenge in learning chaotic solutions, in which validation and test sets may be topologically different portions of the attractor. We, thus, advise that the Single Shot Validation not be used in the validation of Echo State Networks in chaotic attractors. To improve robustness, we analyse different validation strategies (Section 3.2) in the next section.

5. Validation strategies in chaotic systems

5.1. Lorenz system

We compare different validation strategies on the ensemble of $N_{ens} = 50$ networks in a "short" dataset (12 LTs) and a "long" dataset (24 LTs) of the Lorenz system. The long dataset is obtained by the integration of the time series in Fig. 3. In addition to the short dataset, we analyse the long dataset for two reasons. First, we wish to test validation strategies that require larger datasets to fully perform, such as the Walk Forward Validation. Second, we wish to investigate how the robustness is affected by the size of the dataset. We use the Single Shot Validation (SSV), Walk Forward Validation (WFV), K-Fold Validation (KFV), Recycle Validation (RV), and corresponding chaotic versions (subscript c). The long dataset allows us to define an additional chaotic Walk Forward Validation (WFV_c) denoted by the superscript $*$ as detailed in the Supplementary Material (S.3).

The test set has $N_t = 100$ starting points on the attractor to sample different regions of the solutions (more details in Supplementary Material S.2). For each starting point in the test set, the initial reservoir state vector is obtained by performing 1 LT of washout. The Prediction Horizon is globally quantified as an arithmetic mean, $\overline{\text{PH}}_{\text{test}}$, with threshold $k = 0.2$; whereas the Mean Squared Error is globally quantified as a geometric mean, $\overline{\text{MSE}}_{\text{Test}}$, in intervals of 3 LTs.

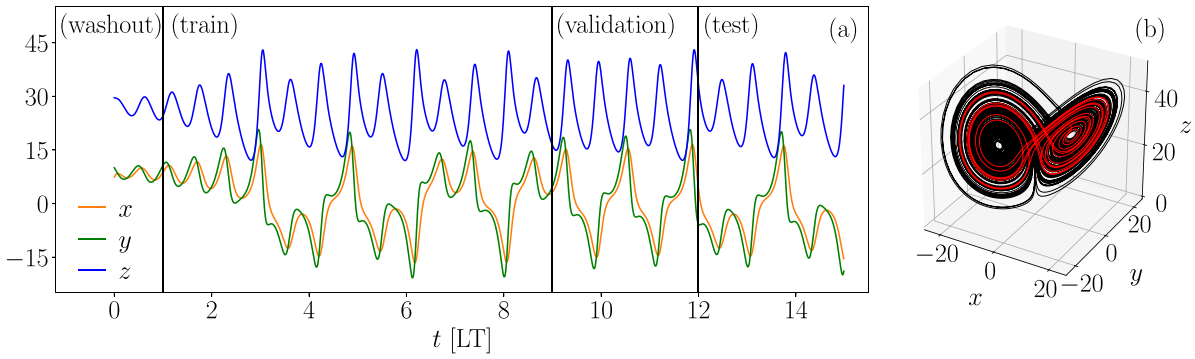

Fig. 3. Solution of the Lorenz system. (a) Time series, and (b) phase plot for a longer time window, where the red trajectory indicates the 0 to 12 LTs interval. Time is expressed in Lyapunov time (LT) units. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

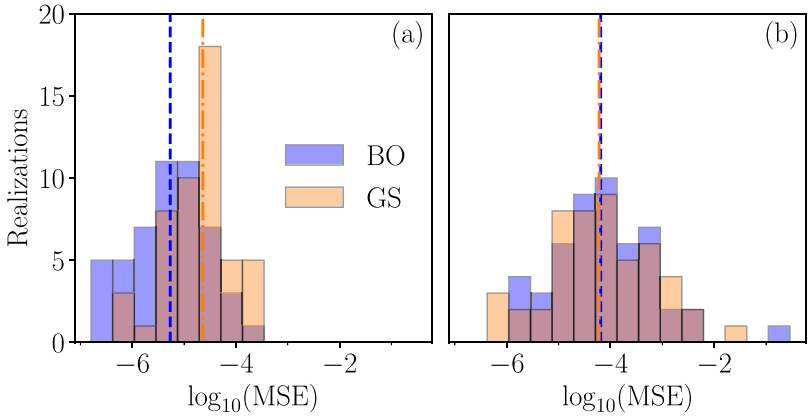

Fig. 4. Performance of the optimal hyperparameters computed by Grid Search (GS) and Bayesian Optimization (BO) in (a) validation and (b) test sets. Vertical lines indicate the median of Grid Search (dash-dotted) and Bayesian Optimization (dashed). The medians are $[5.4, 23.0] \times 10^{-6}$ in the validation set and $[64.8, 60.5] \times 10^{-6}$ in the test set for BO and GS, respectively.

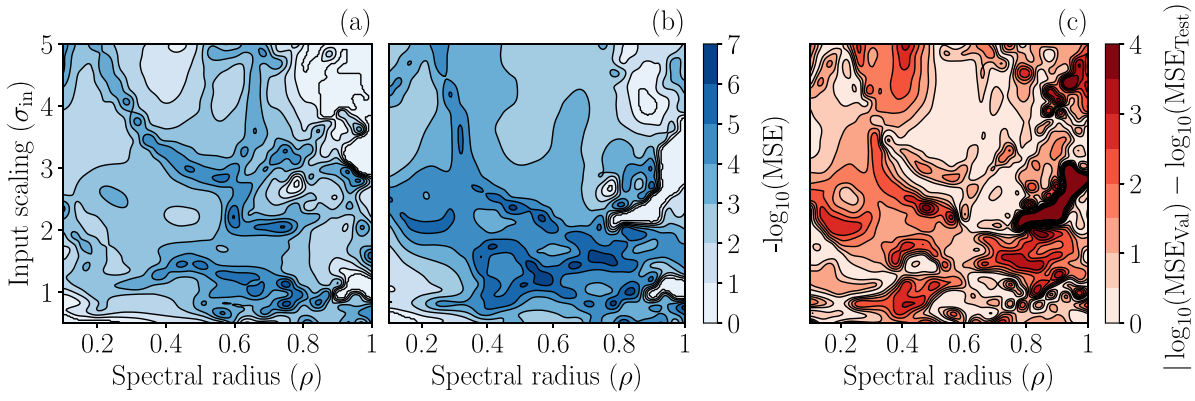

Fig. 5. Mean of the Gaussian Process reconstruction of the MSE in the (a) validation and (b) test sets for a representative network in the ensemble. Frame (c) shows the difference between the two sets. The MSE is saturated to be ≤ 1 in (a, b), whereas the error is saturated to be $\leq 10^4$ in (c). The reconstruction is performed on a grid of 30×30 evaluations of $\log_{10}$(MSE). For the same hyperparameters, the MSE can differ by orders of magnitude between the validation and test sets.

Model-free ESN

Fig. 7 shows the mean of the Gaussian Process reconstruction of $\log_{10}$(MSE) in the hyperparameter space for a representative network of the ensemble. Panels (a, b, c) show the performance of three validation strategies in the validation set, whereas panel (d) shows the performance of the network in the test set. Because the error in (b, c) is similar to the error in (d), and the error in (a) differs from (d), we conclude that in the test set the hyperparameters computed through KFV_c and RV_c perform well, but the hyperparameters computed through SSV perform poorly.

A correlation analysis is shown in Table 1 with the Spearman correlation coefficients, $\tilde{r}_s$ (11) (short and long datasets); and Fig. 8 with scatter plots of the optimal hyperparameters' performance (long dataset, for brevity). The Single Shot Validation has the lowest correlation among all the validation strategies in both datasets. The chaotic versions of the validation strategies correlate better than the corresponding regular versions. In particular, the chaotic K-Fold Validation and the chaotic Recycle Validation have the highest correlations. In general, increasing the size of the dataset increases the correlation, but the Single Shot Validation in the long dataset has a lower correlation than

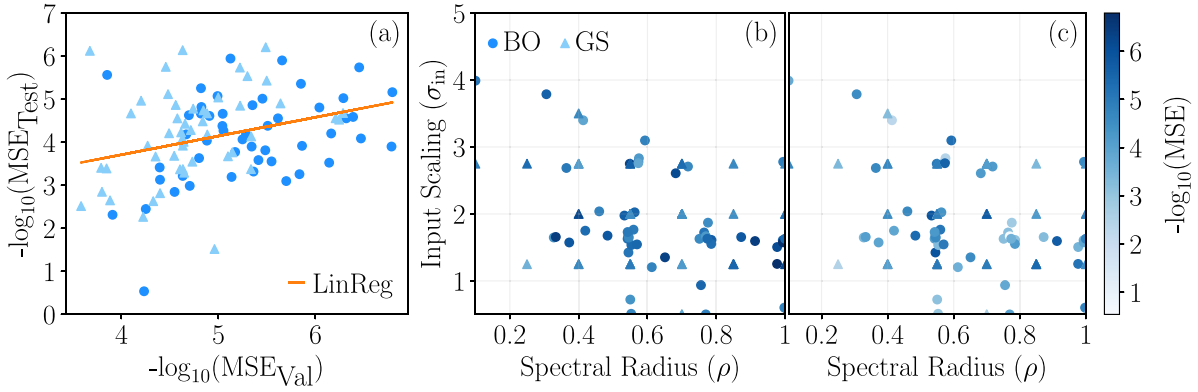

Fig. 6. (a) Linear regression (LinReg) and scatter plot of the MSE of the optimal hyperparameters obtained from Bayesian Optimization (BO) and Grid Search (GS) for each network. Optimal hyperparameters for each network and corresponding MSE in (b) validation and (c) test sets. For different networks the optimal hyperparameters, and their performance, vary significantly.

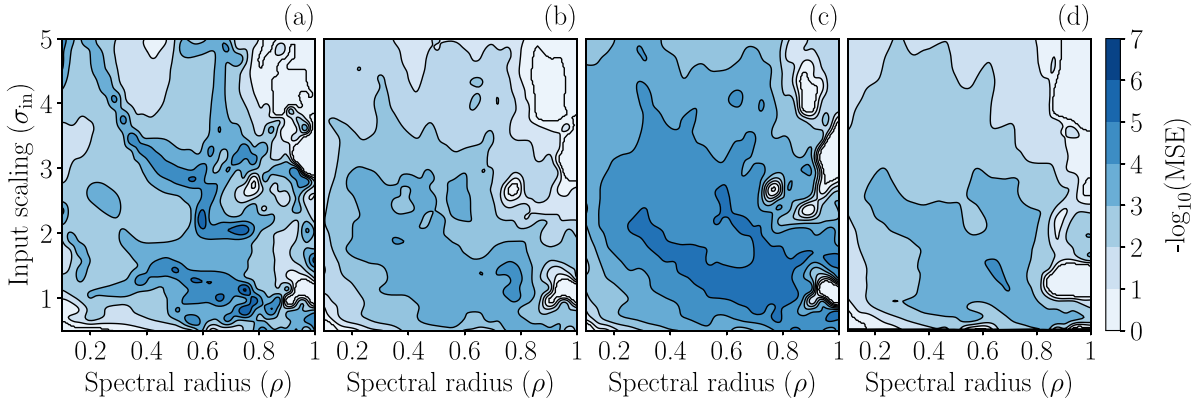

Fig. 7. Mean of the Gaussian Process reconstruction of the MSE for the short dataset of the Lorenz system for a representative network of the ensemble. Validation set for (a) Single Shot Validation (SSV), (b) chaotic K-Fold Validation (KFV_c), and (c) chaotic Recycle Validation (RV_c); and test set (d). The MSE is saturated to be ≤ 1. The Gaussian Process is based on a grid of 30×30 data points.

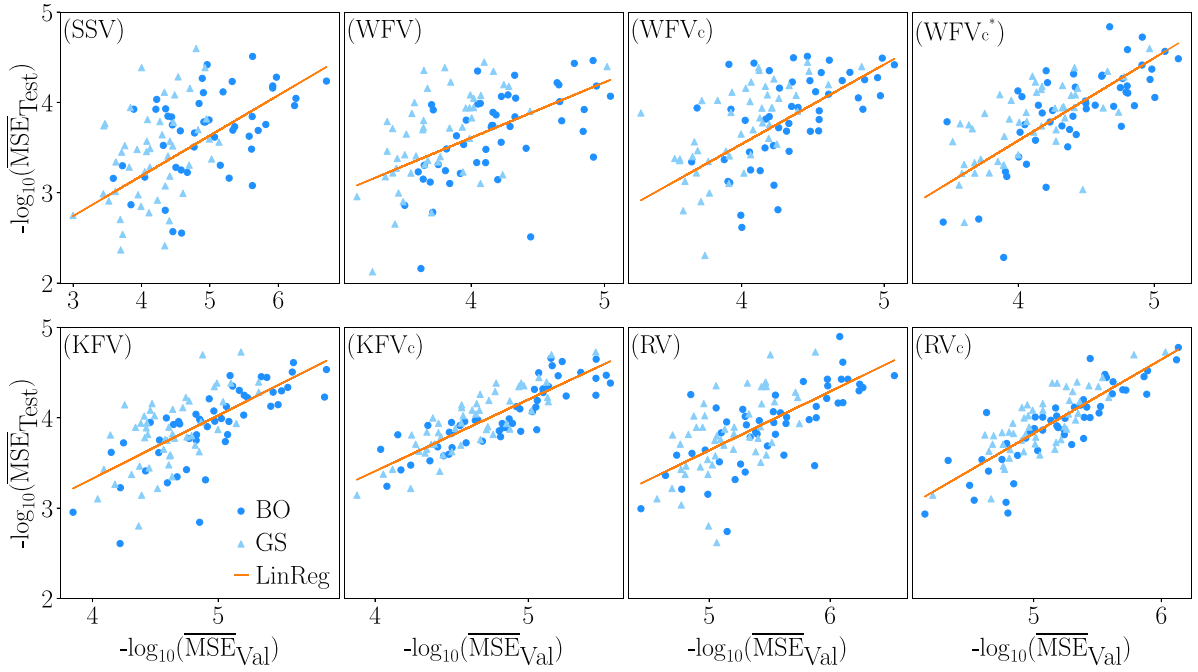

Fig. 8. Linear regression (LinReg) and scatter plot of the MSE of the optimal hyperparameters obtained from Bayesian Optimization (BO) and Grid Search (GS) for each network. Single Shot Validation (SSV), Walk Forward Validation (WFV), K-Fold Validation (KFV), Recycle Validation (RV), and their chaotic versions (subscript c). Long dataset of the Lorenz system.

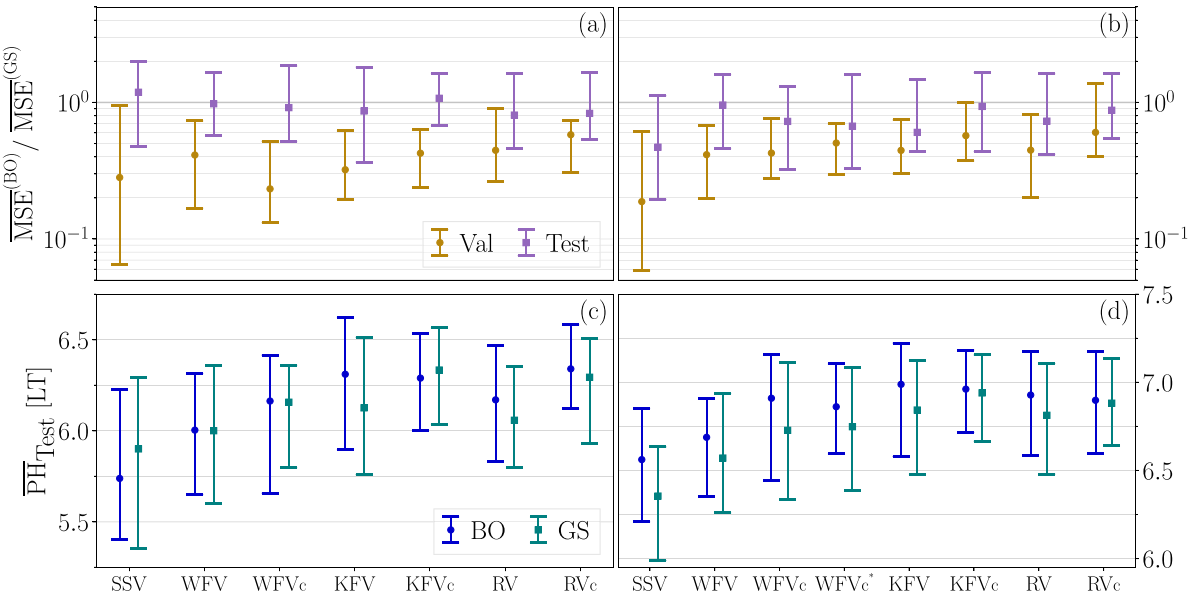

Fig. 9. Comparison between hyperparameter optimization by Bayesian Optimization (BO) and Grid Search (GS). The performance metrics are the Mean Square error (MSE) and Predictability Horizon (PH). 25th (lower bar), 50th (marker) and 75th (upper bar) percentiles. (a, c) short dataset, (b, d) long dataset. Lorenz system.

Table 1
Spearman coefficients between validation and test sets for model-free ESN in the Lorenz system. Bold text indicates the highest correlation in the dataset.

$\tilde{r}_S$	SSV	WFV	WFV_c	WFV_c^*	KFV	KFV_c	RV	RV_c
Short dataset (12 LTs)	0.31	0.31	0.50	–	0.60	**0.65**	0.59	0.62
Long dataset (24 LTs)	0.49	0.51	0.61	0.70	0.70	**0.85**	0.67	0.81

the K-Fold Validations and the Recycle Validations in the short dataset. This further demonstrates the poor robustness of the Single Shot Validation. Last, but not least, the Recycle Validation is computationally cheaper than the K-Fold Validation because the output matrix is the same for the different folds (more analysis on the computational time can be found in Appendix A).

A comparison between Bayesian Optimization (BO) and Grid Search (GS) is shown in Fig. 9. Panels (a, b) show the ratio of the MSE between the optimal hyperparameters obtained by Bayesian Optimization and Grid Search in the validation (Val) and test (Test) sets. In both datasets, Bayesian Optimization outperforms Grid Search in the validation set in $\sim 75\%$ of the networks (except for one outlier). However, BO and GS perform similarly in the test set, especially in the short dataset (a). In the long dataset (b), Bayesian Optimization on average outperforms Grid Search, although there is a decrease in performance with respect to the validation set. Panels (c, d) show the Prediction Horizon (PH) in the test set. The chaotic K-Fold Validation and the chaotic Recycle Validation increase the Prediction Horizon by 0.5 LTs on average with respect to the Single Shot Validation. The Prediction Horizon of the long datasets (d) is $\gtrsim 0.5$ LTs larger than that of the short dataset (c). This results in the performance of the KFV_c and RV_c in the short dataset being closer to the performance of the SSV in the long dataset. Because Bayesian Optimization does not produce a substantial increase in the Prediction Horizon with respect to Grid Search, we conclude that the performance of the networks is more sensitive to the validation strategy rather than the optimization scheme.

Fig. 10 shows the performance and robustness of selected validation strategies for different sizes of the reservoir. The chaotic K-Fold Validation and the chaotic Recycle Validation outperform and are more robust than the Single Shot Validation in all cases studied (apart from one outlier). In small reservoirs, the chaotic K-Fold Validation and the chaotic Recycle Validation are robust ($\tilde{r}_S \geq 0.9$), but as the number of neurons increases, we observe a decrease in robustness for all validation strategies with smaller improvements in the Prediction Horizon. This may be caused by a slight overfitting: the networks are large with respect to the relatively simple and small datasets of the Lorenz system. The overfitting becomes more significant in the Recycle Validation, as compared to the K-Fold Validation, because the validation uses data that has been already seen by the network during training.

Model-informed ESN

We leverage knowledge about the governing equations through $\mathcal{K}(\mathbf{u}_{\text{in}}(t_i))$ in the model in Eq. (6). In this testcase, we use a reduced-order model obtained through Proper Orthogonal Decomposition (POD) (Lumley, 1967; Weiss, 2019) to define a POD-informed ESN. POD provides a fixed rank subspace, $\mathcal{E}$, of the state space, in which the projection of the original state vector optimally preserves its energy. The POD modes/energies are the eigenvectors/eigenvalues of the data covariance matrix, $\mathbf{C} = \frac{1}{m-1}\mathbf{U}^T\mathbf{U}$. The $M \times N_u$ matrix $\mathbf{U}$ is the vertical concatenation of the M snapshots of the N_u-dimensional timeseries used for washout, training and validation of the network, from which its mean, $\mathbf{d} \in \mathbb{R}^{N_u}$, is subtracted columns-wise. We create an N_{POD}-dimensional reduced-order model by taking the modes, $\boldsymbol{\phi}_i$, associated with the N_{POD} largest eigenvalues of $\mathbf{C}$. Because $\mathbf{C}$ is a symmetric matrix, its eigenvectors form an orthonormal basis, which is stored in the orthogonal matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1; \ldots; \boldsymbol{\phi}_{n_{\text{POD}}}]$. The state vector, $\mathbf{q}$, is expressed as a function of its components $\boldsymbol{\xi}$ in the subspace, $\mathcal{E}$, spanned by $\boldsymbol{\Phi}$, and its components $\boldsymbol{\eta}$ in the orthogonal complement of $\mathcal{E}$ spanned by the basis $\boldsymbol{\Psi}$: $\mathbf{q} = \boldsymbol{\Phi}\boldsymbol{\xi} + \boldsymbol{\Psi}\boldsymbol{\eta} + \mathbf{d}$. The evolution equations are then obtained by using a *flat Galerkin approximation* (Matthies & Meyer, 2003), which neglects the contribution of the orthogonal complement: $\boldsymbol{\Psi}\boldsymbol{\eta} \simeq 0$. The dynamical system, $\dot{\mathbf{q}} = \mathbf{f}(\mathbf{q})$, is projected onto $\mathcal{E}$ through $\boldsymbol{\xi} = \boldsymbol{\Phi}^T(\mathbf{q} - \mathbf{d})$ as

$$\dot{\boldsymbol{\xi}} = \boldsymbol{\Phi}^T\mathbf{f}(\boldsymbol{\Phi}\boldsymbol{\xi} + \mathbf{d}). \tag{12}$$

In the POD-informed ESN ($\mathbf{q} \equiv \mathbf{u}_{\text{in}}$), we use $N_{\text{POD}} = 2$ to generate the reduced-order model, which accounts for 96% of the energy of the original signal. We use the evolution of the trajectory on the POD subspace, $\mathcal{E}$, to inform the ESN through $\mathcal{K}(\mathbf{u}_{\text{in}}(t_i)) = \boldsymbol{\xi}(t_{i+1})$.

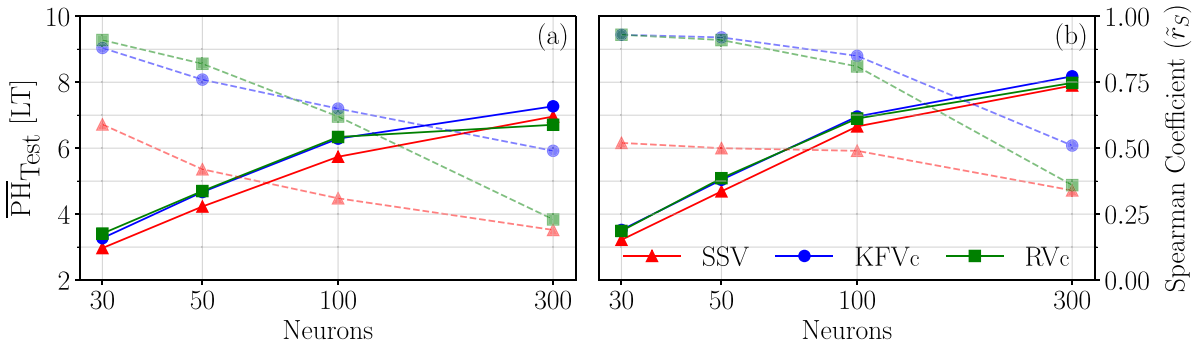

Fig. 10. Prediction Horizon medians (continuous lines) and Spearman coefficients (dashed lines) as functions of the size of the reservoir. The networks are optimized through Bayesian Optimization in the Single Shot Validation (SSV), chaotic K-Fold Validation (KFV_c) and chaotic Recycle Validation (RV_c). Short (a) and long (b) datasets. The x-axis scale highlights the logarithmic relationship between the performance and the size of the reservoir. Lorenz system.

Table 2
Spearman coefficients between validation and test sets for the model-informed ESN in the Lorenz system. Bold text indicates the highest correlation in the dataset.

$\tilde{r}_S$	SSV	WFV	WFV_c	WFV_c^*	KFV	KFV_c	RV	RV_c
Short dataset (12 LTs)	0.15	0.39	0.34	–	0.32	**0.73**	0.41	0.56
Long dataset (24 LTs)	0.19	0.42	0.36	0.51	0.59	**0.80**	0.55	**0.80**

Table 3
Spearman coefficients between validation and test sets in the Lorenz-96 model. Bold text indicates the highest correlation in the dataset.

$\tilde{r}_S$	SSV	WFV	WFV_c	KFV	KFV_c	RV	RV_c
Lorenz-96	0.20	0.62	0.76	0.84	**0.92**	0.69	0.84

We solve the ODE system in Eq. (12) using at each time step forward Euler with initial condition $\boldsymbol{\xi}(t_i) = \boldsymbol{\Phi}^T(\mathbf{u}_{\text{in}}(t_i) - \mathbf{d})$, which is the projection of the input to the network onto $\mathcal{E}$.

As compared to the model-free ESN, there is a decrease in correlation between validation and test sets for almost all the validation strategies (Table 2). The Single Shot Validation is still outperformed by the other strategies, while the chaotic K-Fold Validation and chaotic Recycle Validation have the highest correlation. The POD-informed architecture increases the performance by 1 LT on average (see additional results in the Supplementary Material S.4), but it does not increase the robustness of the networks with respect to the model-free ESN.

5.2. High-dimensional chaos: Lorenz-96

The Lorenz-96 model (Lorenz, 1996) describes the time evolution of an atmospheric scalar that is spatially discretized at J points. It consists of a series of coupled ODEs for the components of the state vector $\mathbf{x} = [x_1, x_2, \ldots, x_J]$

$$\dot{x}_j = (x_{j+1} - x_{j-2})x_{j-1} - x_j + F. \tag{13}$$

The objective of this testcase is to verify that the results observed for low-dimensional chaos in the Lorenz system hold in a higher-dimensional system. We consider the case with periodic boundary conditions, $J = 10$ and $F = 8$, which we integrate in time using a fourth-order Runge–Kutta scheme with time step $dt = 0.024$ LTs (1 LT = $\Lambda^{-1} \approx 0.83$). The dataset used for training and validation consists of 10^4 data points, which span a 240 LTs time series. The network parameters, the size of the ensemble, and the optimization strategies are the same of the previous sections. We study larger networks, $N = 1000$, and modify the input scaling, $b_{\text{in}} = 0.1$, for it to have the same order of magnitude of the input, which is obtained by normalizing the signal by its maximum variation (component-wise). We select the hyperparameters by maximizing the arithmetic mean of the Prediction Horizon (PH) in the validation intervals, instead of minimizing the geometric average of the Mean Squared Error (MSE). We do this for two reasons. First, from preliminary runs and previous studies (Vlachas et al., 2020), we expect the PH to be smaller than 1 LT in most intervals. This means that the optimization of the MSE in intervals that last multiple LTs may not be strongly correlated with maximizing the PH in the same intervals. Second, in contrast to the three-dimensional Lorenz system, the large size of the dataset with respect to the PH allows us to define multiple validation intervals to optimize the PH directly. We define the Spearman coefficients (11) with the PH. We test the validation strategies (detailed in Supplementary Material S.3) by computing the arithmetic mean $\overline{\text{PH}}_{\text{test}}$ of the Prediction Horizon on $N_t = 100$ starting points along the attractor. For each point, the initial reservoir state vector is obtained by performing 100 steps of washout.

The Spearman coefficients are shown in Table 3. The Single Shot Validation is the least robust strategy, whereas the chaotic Recycle Validation and chaotic K-Fold Validation are the most robust. Apart from the Single Shot Validation, the Spearman coefficients are markedly larger than in the three-dimensional Lorenz system (Table 1). This is due to the large size of the dataset with respect to the Lyapunov Time of the system, which allows us to use a large number of splits in the validation strategies. This means that in large chaotic datasets robust results are obtained by selecting the hyperparameters through the average of multiple intervals in the dataset.

Fig. 11 shows the performance of the validation strategies. Bayesian Optimization (BO) outperforms Grid Search (GS) in all the strategies in the validation (Val) set (panel (a)). On the one hand, the poor robustness of the Single Shot Validation causes the two optimization methods to perform similarly in the test set. On the other hand, BO outperforms GS in more than 75% networks in the test set in the K-Fold Validations and Recycle Validations. Panel (b) shows the Prediction Horizon in the test set. The Single Shot Validation has the smallest PH, whereas the chaotic K-Fold Validation has the largest. The performance for reservoirs of different sizes is shown in Panel (c). The Prediction Horizon increases with the size of the reservoir, while the robustness and relative performance of the selected validation strategies is qualitatively similar to the 1000-neuron case.

6. Quasiperiodic versus chaotic solutions

We analyse the nonlinear oscillator proposed by Kuznetsov et al. (2010), which physically represents a self-oscillatory discharge in an electric circuit. The oscillator is a three-dimensional

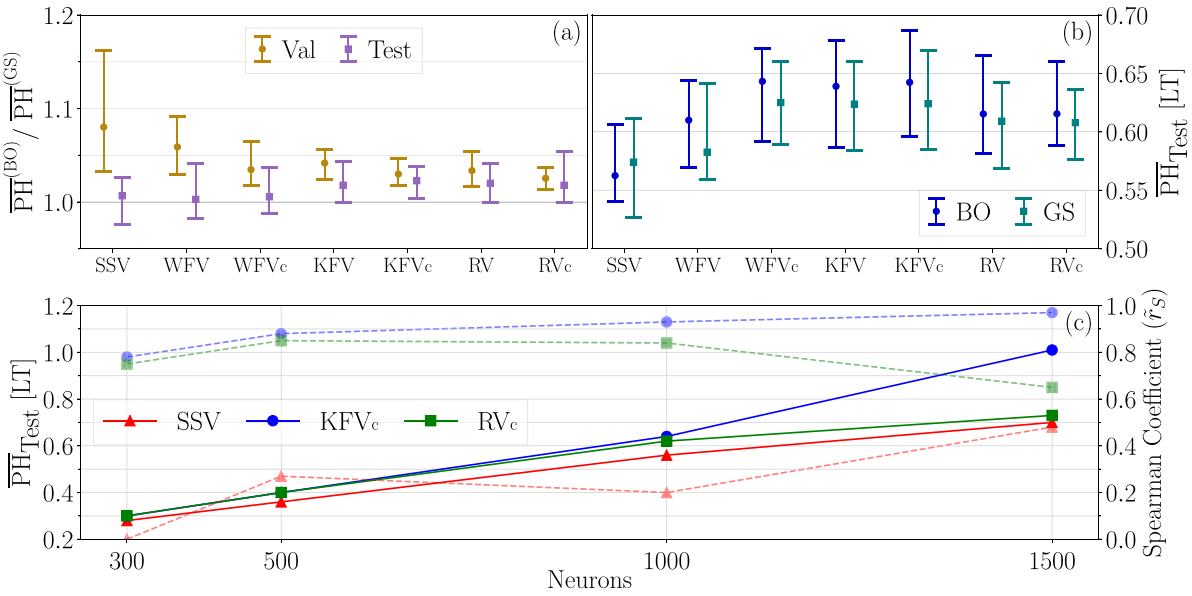

Fig. 11. 25th (lower bar), 50th (marker) and 75th (upper bar) percentiles of the Predictability Horizon (PH) for hyperparameter optimization by Bayesian Optimization (BO) and Grid Search (GS), (a, b). PH medians from BO (continuous lines) and Spearman coefficients (dashed lines) as a function of the size of the reservoir, (c). Lorenz-96 model.

system, which can display periodic, quasiperiodic and chaotic behaviours as a function of the parameters $[\lambda, \omega_0, \mu]$

$$\dot{x} = y,$$
$$\dot{y} = y(\lambda + z + x^2 - \frac{1}{2}x^4) - \omega_0^2 x,$$
$$\dot{z} = \mu - x^2. \tag{14}$$

The primary purpose of this testcase is to compare the robustness of Echo State Networks in forecasting quasiperiodic solutions versus chaotic solutions. This enables us to determine whether the challenges encountered in Section 5 are specific to learning chaotic time series. We obtain quasiperiodic and chaotic solutions by setting $\lambda = 0$, $\omega_0 = 2.7$ and $\mu_{Qp} = 0.9$ and $\mu_{Ch} = 0.5$, respectively. In both cases, we generate the solution using the adaptive integration scheme implemented in `scipy.odeint` and store the solution every $dt = 0.0008$ LT, where LT ≈ 25 for the chaotic case (Kuznetsov et al., 2010). In both cases, the dataset consists of 7.5 LTs to be used for washout, training and validation.

The network parameters, the size of the ensemble, and the optimization strategies are the same as those of Section 4. We modify the input scaling, $b_{\text{in}} = 0.1$, for it to have the same order of magnitude of the input, which is obtained by normalizing the signal by its maximum variation component-wise. We study the enlarged interval $\rho = [0.01, 1]$ because we observed empirically that the optimal hyperparameters often lie in the range $\rho \leq 0.1$. Given the multiple orders of magnitude of the spectral radius, the hyperparameter space is analysed in a logarithmic scale. We use the same architecture and validation strategies for the quasiperiodic and chaotic case (as detailed in the Supplementary Material S.3). The different strategies are tested by computing the arithmetic mean $\overline{\text{PH}}_{\text{test}}$ of the Prediction Horizon on N_t starting points for the chaotic case, and by computing the geometric mean $\overline{\text{MSE}}_{\text{test}}$ of the Mean Squared Error in 2 LTs intervals starting from the same points. In the chaotic case, we select $N_t = 75$, whereas in the quasiperiodic we select $N_t = 50$ through the procedure described in Supplementary Material S.2. For each starting point in the test set, the initial reservoir state vector is obtained by performing 0.5 LTs of washout. The performance in the quasiperiodic dataset is assessed only through the Mean Squared Error because the Prediction Horizon is infinite, i.e., a quasiperiodic solution has zero dominant Lyapunov exponents (Kantz & Schreiber, 2004).

Table 4
Spearman coefficients between validation and test sets for the model-free ESN in the Kuznetsov oscillator. Bold text indicates the highest correlation in the dataset.

$\tilde{r}_S$	SSV	WFV	WFV$_c$	KFV	KFV$_c$	RV	RV$_c$
Quasiperiodic dataset	0.80	0.75	0.71	0.93	0.92	**0.97**	**0.97**
Chaotic dataset	0.49	0.48	0.58	0.70	0.76	0.66	**0.81**

Model-free ESN

The Spearman coefficients (Table 4) show that the correlation between validation and test sets is higher in the quasiperiodic dataset than the chaotic dataset. Notably, the peak $\tilde{r}_s = 0.97$ obtained in the Recycle Validations indicates almost complete correlation. As before, the Single Shot Validation is outperformed by the K-Fold Validation and Recycle Validation, but its correlation in the quasiperiodic dataset is higher than that of chaotic cases. The high correlation in the quasiperiodic dataset is identified as the dense clustering around the linear regression of Fig. 12. Two remarks can be made. On the one hand, the high correlation in the quasiperiodic dataset implies that the challenges in producing robust results in Echo State Networks in chaotic attractors are due to the complexity of the chaotic signal, rather than the properties of the networks. On the other hand, the marked difference in performance between different networks is still present in the quasiperiodic dataset, which means that ESNs are sensitive to the realizations (further analysis is reported in Appendix C). Practically, we advise that different networks be optimized independently in the quasiperiodic case as well.

Panels (a, b) of Fig. 13 show the ratio of the MSE between the optimal hyperparameters obtained by Bayesian Optimization (BO) and the optimal hyperparameters from Grid Search (GS) in the validation and test sets. On the one hand, in the quasiperiodic case (a) the performance in the validation set is similar to the test

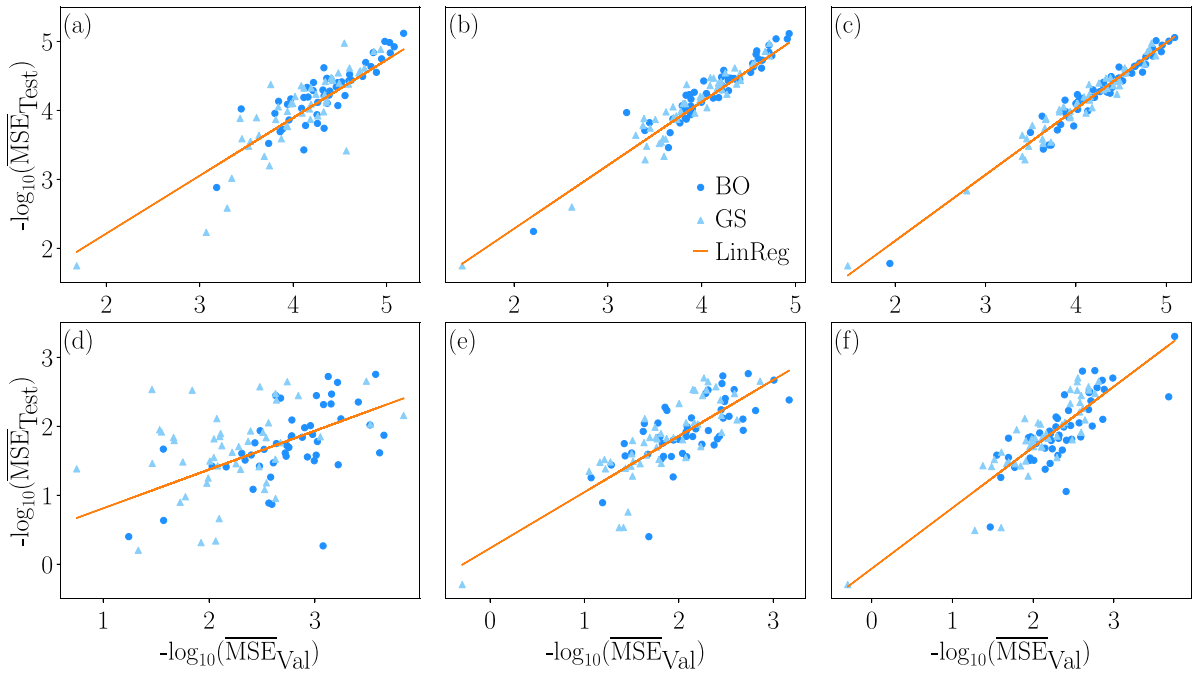

Fig. 12. Linear regression (LinReg) and scatter plot of the MSE of the optimal hyperparameters obtained from Bayesian Optimization (BO) and Grid Search (GS) for each network. (a, d) Single Shot Validation, (b, e) chaotic K-Fold Validation and (c, f) chaotic Recycle Validation. (a–c) quasiperiodic and (d–f) chaotic datasets. Kuznetsov oscillator.

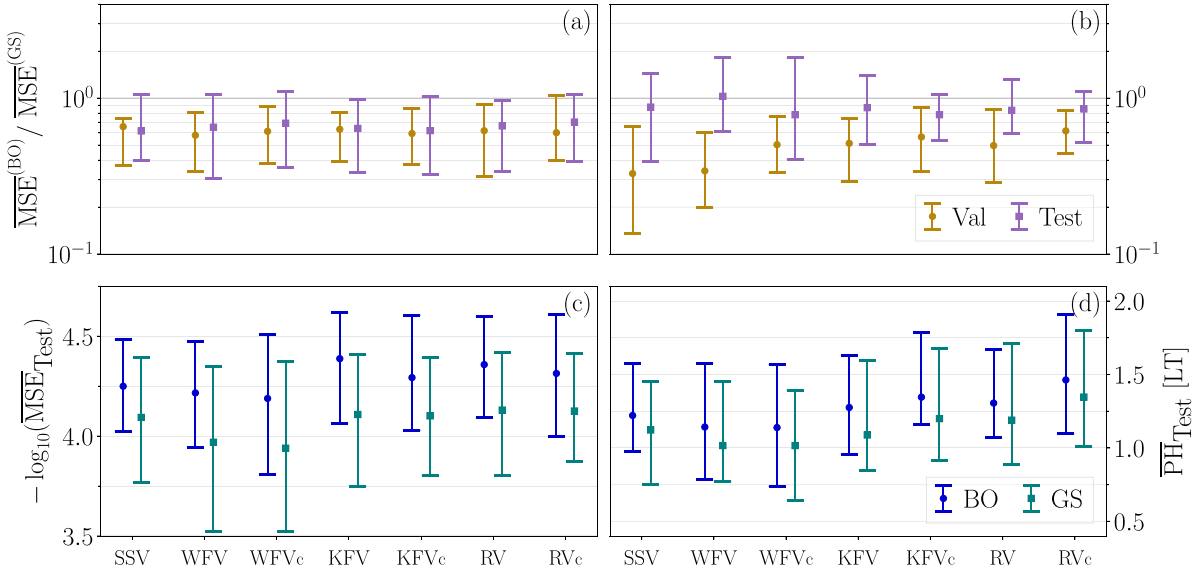

Fig. 13. Comparison between hyperparameter optimization by Bayesian Optimization (BO) and Grid Search (GS) for the two performance metrics (MSE, PH). 25th (lower bar), 50th (marker) and 75th (upper bar) percentiles. (a, c) quasiperiodic dataset, (b, d) chaotic dataset. Kuznetsov oscillator.

set. One the other hand, in the chaotic case (b) BO outperforms GS in the validation set, although the two schemes perform similarly in the test set. In panels (c, d), we show the performance of the networks in the test set using the MSE for the quasiperiodic dataset (c) and the Prediction Horizon in the chaotic dataset (d). In the quasiperiodic dataset, Bayesian Optimization outperforms Grid Search, and the K-Fold Validations and Recycle Validations outperform the other validation strategies. In the chaotic dataset, as seen in the Lorenz system, Bayesian Optimization only slightly outperforms Grid Search, while the K-fold Validations and Recycle Validations still outperform the other validation strategies. An analysis of the performance and robustness of the validation strategies for different sizes of the reservoir (up to 500 neurons) is reported in the Supplementary Material S.4. The results are qualitatively similar to those observed for 100 neurons. In all cases (except for one outlier), the chaotic Recycle Validation shows the smallest MSE, the largest Prediction Horizon and the highest robustness.

Model-informed ESN

We design a Forward Euler (FE) informed ESN (6) by integrating in time with forward Euler the y Eq. (14) only

$$\mathcal{K}(\mathbf{u}_{\text{in}}) = y + dt(y(\lambda + z + x^2 - \frac{1}{2}x^4) - \omega_0^2 x). \tag{15}$$

Table 5
Spearman coefficients between validation and test sets for the model-informed ESN in the Kuznetsov oscillator. Bold text indicates the highest correlation in the dataset.

$\tilde{r}_S$	SSV	WFV	WFV_c	KFV	KFV_c	RV	RV_c
Quasiperiodic dataset	0.78	0.65	0.67	0.71	0.80	**0.98**	**0.98**
Chaotic dataset	0.57	0.63	0.63	0.75	0.79	0.71	**0.85**

Table 5 shows the Spearman coefficients for the FE-informed model. In the quasiperiodic dataset, the correlation decreases for all the validation strategies with respect to the model-free case (see Table 4) except for the Recycle Validations, which have the highest correlation. However, in the chaotic dataset, the correlation increases for all the validation strategies. Here, the chaotic K-Fold Validation and chaotic Recycle Validation are the strategies with the highest correlation. In the same fashion as the Lorenz system, the FE-informed architecture *per se* does enhance the performance, but it does not enhance the robustness of Echo State Networks (additional results can be found in the Supplementary Material S.4).

7. Conclusions

The Echo State Network (ESN) is a reservoir computing architecture that is able to learn accurately the nonlinear dynamics of systems from data. The overarching objective of this paper is to investigate and improve the robustness of ESNs, with a focus on the forecasting of chaotic systems. First, we analyse the Single Shot Validation, which is the commonly used strategy to select the hyperparameters. We show that the Single Shot Validation is the least performing strategy to fine-tune the hyperparameters. Second, we validate the ESNs on multiple points of the chaotic attractor, for which the validation set is not necessarily subsequent in time to the training set. We propose the Recycle Validation and the chaotic version of existing validation strategies based on multiple folds, such as the Walk Forward Validation and the K-Fold Cross Validation. The K-Fold Validation and Recycle Validation offer the greatest robustness and performance, with their chaotic versions outperforming the corresponding regular versions. Importantly, the Recycle Validation is computationally cheaper than the K-Fold Cross Validation. Third, we compare Bayesian Optimization with Grid Search to compute the optimal hyperparameters. We find that in the validation set Bayesian Optimization is an optimization scheme that consistently finds a set of hyperparameters that perform significantly better than the Grid Search. On the one hand, in learning quasiperiodic solutions, hyperparameters that work optimally in the validation set continue to work optimally in the test set. This is because quasiperiodic solutions are predictable (i.e., they do not have positive Lyapunov exponents). This finding is, thus, expected to generalize to other predictable solutions, such as frequency-locked solutions and limit cycles. On the other hand, in learning chaotic solutions, hyperparameters that work optimally in the validation set do not necessarily work optimally in the test set. We argue that this occurs because of the chaotic nature of the attractor, in which the nonlinear dynamics, although deterministic, manifest themselves as unpredictable variations. Fourth, we analyse the model-free ESN, which is fully data-driven, and the model-informed ESN, which leverages knowledge of the governing equations. We find that the model-informed architecture markedly improves the network's prediction capabilities, but it does not improve the robustness. Finally, we find that the optimal hyperparameters are significantly sensitive to the random initialization of the ESN. Practically, when working with an ensemble of ESNs, we recommend computing the optimal hyperparameters for each network. In the test performed in the paper, this can increase up to six Lyapunov Times the network's Prediction Horizon as compared to using the same set of hyperparameters for all realizations.

This work opens up new possibilities for using Echo State Networks and, in general, recurrent neural networks, for robust learning of chaotic dynamics.

Code and supplementary material

The code for the validation strategies and optimization schemes used in this work can be found in the openly-available GitLab repository https://gitlab.com/ar994/robust-validation-esn. Supplementary material related to this article can be found online at https://doi.org/10.1016/j.neunet.2021.05.004.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

A. Racca is supported by the EPSRC-DTP and the Cambridge Commonwealth, European & International Trust under a Cambridge European Scholarship. L. Magri is supported by the Royal Academy of Engineering Research Fellowship scheme and the visiting fellowship at the Technical University of Munich – Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement no. 291763. The authors would like to thank Dr. N. A. K. Doan and F. Huhn for insightful discussions. The authors are grateful to two anonymous reviewers and the handling editor (Prof. H. Jaeger) for their thorough comments.

Appendix A. Computational time

In Fig. A.14, we show the CPU time required by the validation strategies to perform a Grid Search in hyperparameters space for a single network. The computational advantage of the Recycle Validation with respect to the K-Fold Validation increases with the size of the dataset and the size of the reservoir. We expect the improvement in computational time to be more significant in RNN architectures whose training is more expensive, such as LSTMs and GRUs.

The Bayesian Optimization described in Section 4 costs approximately 6 s more per network in all the cases shown. This is because the additional cost of the Bayesian Optimization is independent of the cost of the evaluation function.

Implementation

To reduce the computational cost of the Walk Forward Validation and of the K-Fold Validation, for each set of hyperparameters, we store $\mathbf{R}$, $\mathbf{R}\mathbf{R}^T$ and $\mathbf{R}\mathbf{U}_{\mathrm{d}}^T$ in Eq. (4) for the entire dataset. For the i-th fold, we then compute $\mathbf{R}_i\mathbf{R}_i^T$ (and in the same way $\mathbf{R}_i\mathbf{U}_{\mathrm{d}i}^T$) that we need for ridge regression through

$$\mathbf{R}_i\mathbf{R}_i^T = \mathbf{R}\mathbf{R}^T - \hat{\mathbf{R}}_i\hat{\mathbf{R}}_i^T, \tag{A.1}$$

where $\hat{\mathbf{R}}_i$ is the complement to $\mathbf{R}_i$ for the entire dataset, namely $\mathbf{R} \in \mathbb{R}^{N_{\hat{r}} \times N_{\mathrm{tr}}}$, $\mathbf{R}_i \in \mathbb{R}^{N_{\hat{r}} \times N_i}$ and $\hat{\mathbf{R}}_i \in \mathbb{R}^{N_{\hat{r}} \times \hat{N}_i}$, where $N_i + \hat{N}_i = N_{\mathrm{tr}}$. Since for large datasets computing $\mathbf{R}_i\mathbf{R}_i^T$ (and $\mathbf{R}_i\mathbf{U}_{\mathrm{d}i}^T$) is the main computational cost of solving Eq. (4) and $N_i \gg N_i$, using

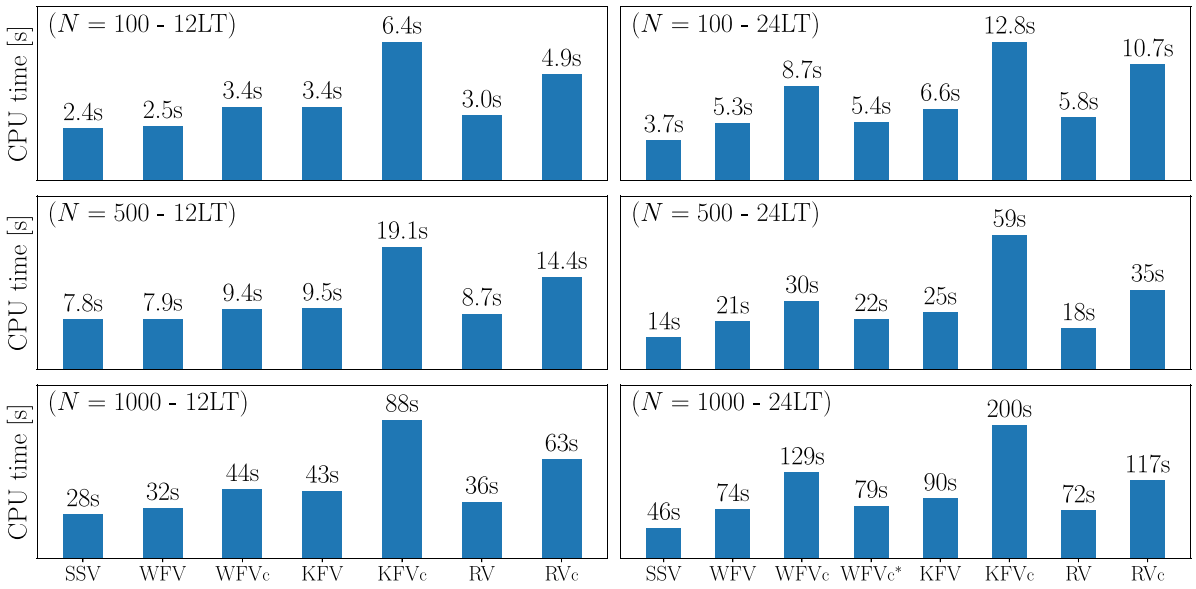

Fig. A.14. CPU time required for a single network of size N to perform a 7 $\times$ 7 Grid Search in hyperparameters space in the 12 LT and 24 LT datasets of the Lorenz system. The validation strategies are the Single Shot Validation (SSV), the Walk Forward Validation (WFV), K-Fold Validation (KFV), Recycle Validation (RV), and respective chaotic versions (subscript c). The runs are on a single Intel i7-8750H processor.

Eq. (A.1) reduces significantly the computational time required by the strategies. For example, in the Lorenz-96 testcase the total time required by the chaotic K-Fold is reduced by a factor three with respect to computing $\mathbf{R}_i\mathbf{R}_i^T$ directly.

Appendix B. Bayesian optimization for hyperparameters

After we evaluate the objective function at N_{st} starting points, the objective function is reconstructed in the hyperparameter search space using the function evaluations as data points for noise-free Gaussian Process Regression. The computational cost of the regression is proportional to N_d^3, where N_d is the number of data points, because of the inversion of the covariance matrix. The inversion is performed by Cholesky factorization regularized by the addition of $\alpha = 10^{-10}$ on the diagonal elements.

Once the Gaussian Process is performed, the next point at which to evaluate the objective function is selected in the hyperparameter space to maximize the acquisition function. The acquisition function evaluates a potential point usefulness in finding the global minimum, so that points with a high value of the acquisition function are selected during the search. A new point can be chosen for one of two reasons: (i) to try to find a new minimum by using current knowledge of the search space and (ii) to increase the knowledge of the space by exploring new regions. This trade-off is called balance between exploitation and exploration. Practically, the most used acquisition functions in the literature are the Probability of Improvement (PI), the Expected Improvement (EI) and the Lower Confidence Bound (LCB) (Brochu et al., 2010). On a given testcase, it is difficult to determine a priori which acquisition function will perform better. For this reason we use the gp-hedge algorithm (Hoffman et al., 2011), which improves the performance with respect to the single acquisition functions. In the algorithm, when deciding the next point of the search, the three acquisition functions are evaluated over the search space. Each acquisition function provides its own optimal point as a candidate. The next point at which the function is going to be evaluated is selected among the three candidates with probability given by the softmax function. The softmax function is evaluated on the cumulative reward from previous candidate points proposed by the acquisition functions, so that the strategy leans towards exploitation as the search progress. Once the point is selected, the Gaussian Process Regression is performed again using the updated set of data points, until the prescribed maximum number of function evaluations is reached. More details are reported in the Supplementary Material S.5.

Appendix C. Hyperparameter variations for different realizations

As shown in Fig. 6 for the Lorenz system, different network realizations have different optimal hyperparameters, which vary significantly from one network realization to another. This suggests that different networks need to be trained independently. If we select a fixed set of hyperparameters, some networks will perform poorly (Haluszczynski & Räth, 2019). In this appendix, we quantify the difference in performance between optimizing the network independently and using a fixed set of hyperparameters for the entire ensemble. Fig. C.15 shows the mean of the Gaussian Process reconstruction of the $\log_{10}$(MSE) in the test set for the short dataset in the Lorenz system. In panels (a, b), we show the MSE in the test set for two representative networks from the ensemble, while in panel (c), we show the error between the two networks. The two networks differ substantially. The same hyperparameters may result in MSEs that differ by more than four orders of magnitude.

To quantitatively evaluate the performance of the networks, we assess two possible choices of fixed hyperparameters: (i) we search the optimal fixed hyperparameters by minimizing the geometric mean over the 50 networks of the MSE in the validation set; (ii) we use the hyperparameters obtained by performing the search on a representative network from the ensemble and use those hyperparameters for all the networks. In both (i) and (ii), we perform the search using Bayesian Optimization in the chaotic K-Fold Cross Validation (KFV_c) and chaotic Recycle Validation (RV_c). Fig. C.16 shows the violin plots and 25th, 50th and 75th percentiles for the Prediction Horizon in the test set for the Lorenz system. Using fixed hyperparameters yields a decrease in performance in the percentiles of around 0.5 LTs when using (i), and of more than 1 LTs when using (ii). In addition, the tail of the distribution prolongates to values of the Prediction Horizon below 1 LT, which means that the fixed hyperparameters perform poorly in a fraction of the networks. Finally, we note

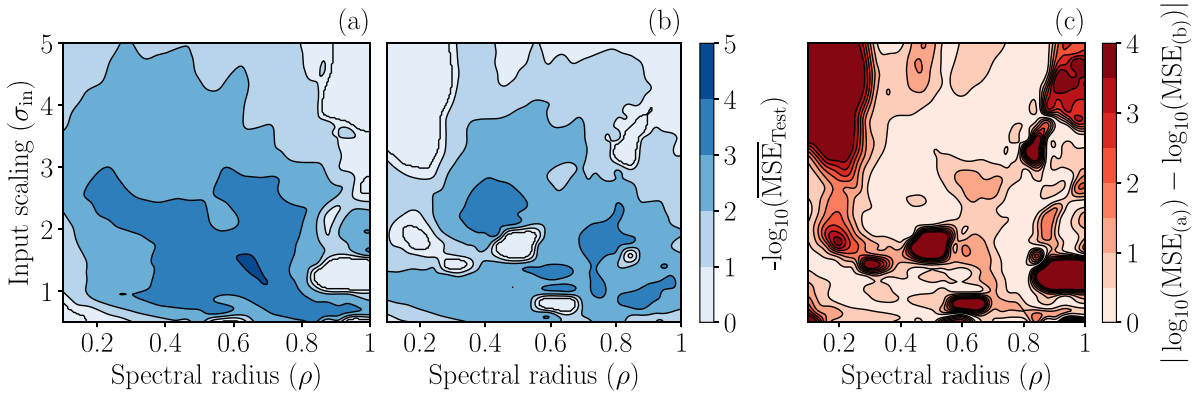

Fig. C.15. Mean of the Gaussian Process reconstruction of the MSE in the test set for (a, b) two representative networks in the short dataset of the Lorenz system, and (c) difference between the two networks. For visualization purposes we saturate the MSE to be $\leq 10^0$ and the error to be $\leq 10^4$. The Gaussian Process is based on a grid of 30 $\times$ 30 data points. For the same hyperparameters, the MSE can differ by orders of magnitude between the two networks.

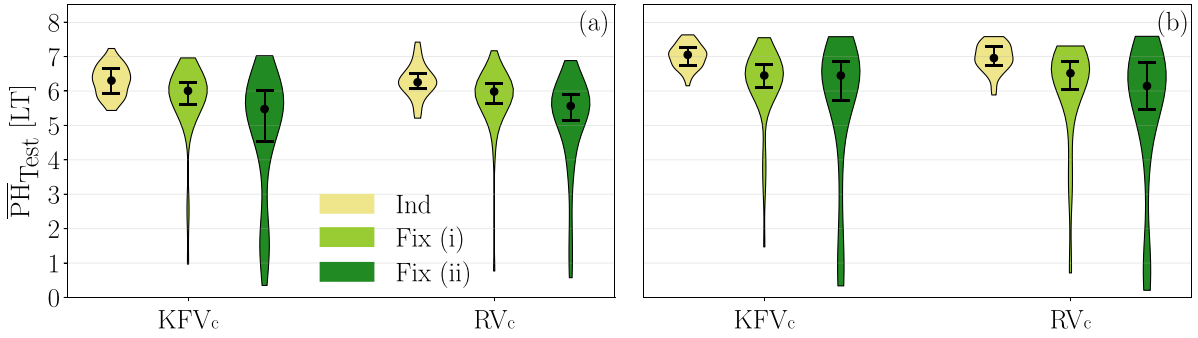

Fig. C.16. Violin plots and 25th (lower bar), 50th (marker) and 75th (upper bar) percentiles of the Prediction Horizon in the test set for the 50 networks ensemble in the (a) short (b) and long datasets in the Lorenz system. Independent optimization (Ind) of each network, optimal set of fixed hyperparameters (Fix (i)), and optimal hyperparameters of a single network (Fix (ii)). We use Bayesian Optimization in the chaotic K-Fold Validation (KFV_c) and chaotic Recycle Validation (RV_c).

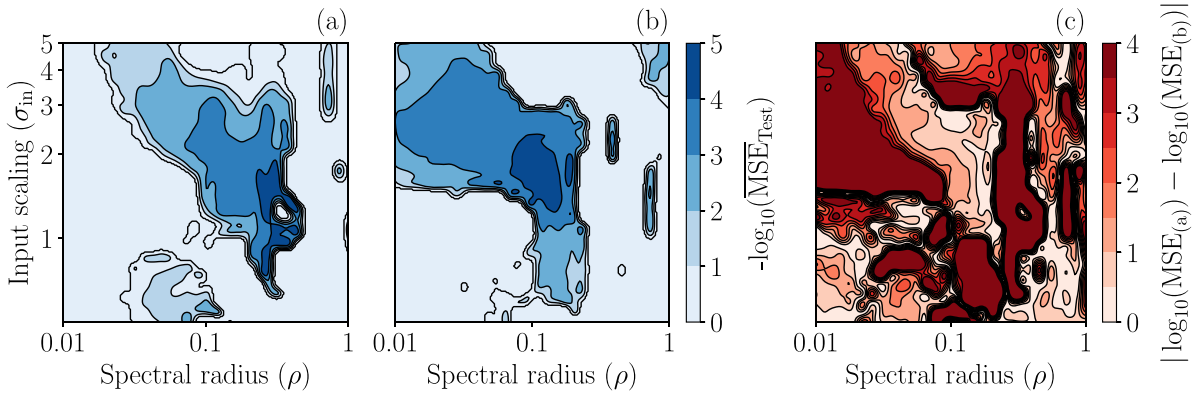

Fig. C.17. Mean of the Gaussian Process reconstruction of the MSE in the test set for (a, b) two representative networks in the quasiperiodic dataset, and (c) difference between the two networks. For visualization purposes we saturate the MSE to be ≤ 1 and the error to be $\leq 10^4$. The Gaussian Process is based on a grid of 30 $\times$ 30 data points.

that the decrease in the Prediction Horizon percentiles for (ii) is larger than the improvement that we obtain when using the new validation strategies, the increased size of the dataset or the model-informed architecture. This means that optimizing the network independently, and therefore not using hyperparameters obtained from validating one network for another network, is key in Echo State Networks. Similar conclusions can be drawn for the quasiperiodic dataset in the Kuznetsov oscillator (Figs. C.17, C.18). This means that the difference between realizations is caused by the ESNs, and not by chaos.

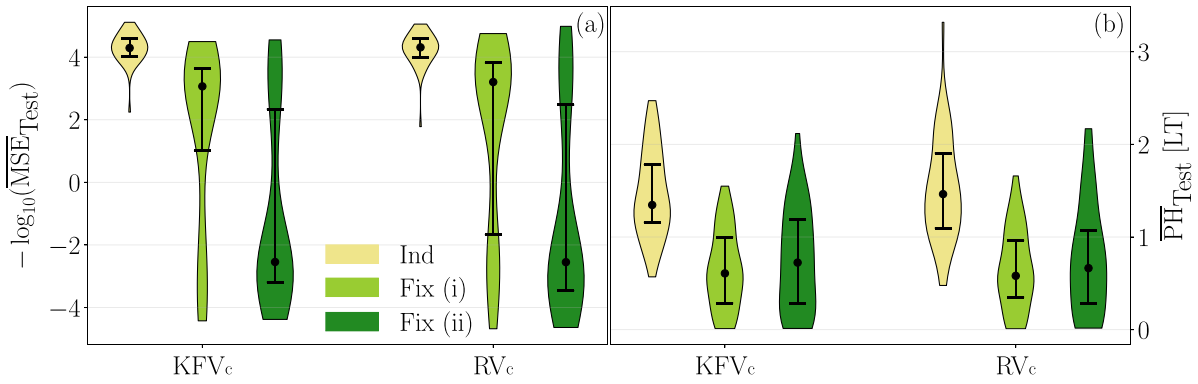

Fig. C.18. Violin plots and 25th (lower bar), 50th (marker) and 75th (upper bar) percentiles for the 50 networks ensemble of the MSE in the quasiperiodic dataset, (a), and the Prediction Horizon for the chaotic dataset, (b), in the test set in the Kuznetsov Oscillator. Independent optimization (Ind) of each network, optimal set of fixed hyperparameters (Fix (i)), and optimal hyperparameters of a single network (Fix (ii)). We use Bayesian Optimization in the chaotic K-Fold Validation (KFV_c) and chaotic Recycle Validation (RV_c).

References

Baker, N., Alexander, F., Bremer, T., Hagberg, A., Kevrekidis, Y., Najm, H., et al. (2019). *Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence*: *Tech. rep.*, Washington, DC (United States): USDOE Office of Science (SC).

Bec, J., Biferale, L., Boffetta, G., Cencini, M., Musacchio, S., & Toschi, F. (2006). Lyapunov Exponents of heavy particles in turbulence. *Physics of Fluids*, *18*(9), 1–5. http://dx.doi.org/10.1063/1.2349587, arXiv:0606024.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166.

Boffetta, G., Cencini, M., Falcioni, M., & Vulpiani, A. (2002). Predictability: A way to characterize complexity. *Physics Reports*, *356*(6), 367–474. http://dx.doi.org/10.1016/S0370-1573(01)00025-4, arXiv:0101029.

Bolker, B. M., & Grenfell, B. T. (1993). Chaos and biological complexity in measles dynamics. *Proceedings of the Royal Society of London, Series B*, *251*(1330), 75–81.

Brochu, E., Cora, V. M., & De Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv preprint arXiv:1012.2599.

Brunton, S. L., Noack, B. R., & Koumoutsakos, P. (2020). Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, *52*, 477–508.

Chattopadhyay, A., Hassanzadeh, P., & Subramanian, D. (2020). Data-driven predictions of a multiscale lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network. *Nonlinear Processes in Geophysics*, *27*(3), 373–389.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., et al. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724–1734). Doha, Qatar: Association for Computational Linguistics, http://dx.doi.org/10.3115/v1/D14-1179, https://www.aclweb.org/anthology/D14-1179, arXiv preprint arXiv:1406.1078.

Deissler, R. G. (1986). Is Navier-Stokes turbulence chaotic? *Physics of Fluids*, *29*(5, May 1986), 1453–1457. http://dx.doi.org/10.1063/1.865663, URL https://aip.scitation.org/doi/10.1063/1.865663.

Doan, N. A. K., Polifke, W., & Magri, L. (2019). Physics-informed echo state networks for chaotic systems forecasting. In *International Conference on Computational Science* (pp. 192–198). Springer.

Doan, N. A. K., Polifke, W., & Magri, L. (2020a). Learning hidden states in a chaotic system: A physics-informed echo state network approach. In *International Conference on Computational Science* (pp. 117–123). Springer.

Doan, N. A. K., Polifke, W., & Magri, L. (2020b). Physics-informed echo state networks. *Journal of Computer Science*, *47*, Article 101237. http://dx.doi.org/10.1016/j.jocs.2020.101237, URL http://www.sciencedirect.com/science/article/pii/S1877750320305408.

Doan, N. A. K., Polifke, W., & Magri, L. (2021). Short-and long-term prediction of a chaotic flow: A physics-constrained reservoir computing approach. arXiv preprint arXiv:2102.07514.

Ferreira, A. A., Ludermir, T. B., & De Aquino, R. R. (2013). An approach to reservoir computing design and training. *Expert Systems with Applications*, *40*(10), 4172–4182.

Gonon, L., & Ortega, J. -P. (2021). Fading memory echo state networks are universal. *Neural Networks*, http://dx.doi.org/10.1016/j.neunet.2021.01.025, URL https://www.sciencedirect.com/science/article/pii/S0893608021000332.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Griffith, A., Pomerance, A., & Gauthier, D. J. (2019). Forecasting chaotic systems with very low connectivity reservoir computers. *Chaos. An Interdisciplinary Journal of Nonlinear Science*, *29*(12), Article 123108.

Grigoryeva, L., & Ortega, J. -P. (2018). Echo state networks are universal. *Neural Networks*, *108*, 495–508. http://dx.doi.org/10.1016/j.neunet.2018.08.025, URL https://www.sciencedirect.com/science/article/pii/S089360801830251X.

Guckenheimer, J., & Holmes, P. (2013). *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields, Vol. 42*. Springer Science & Business Media.

Haluszczynski, A., & Räth, C. (2019). Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing. *Chaos. An Interdisciplinary Journal of Nonlinear Science*, *29*(10), Article 103143.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., et al. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. http://dx.doi.org/10.1038/s41586-020-2649-2.

Hassanaly, M., & Raman, V. (2019). Ensemble-LES analysis of perturbation response of turbulent partially-premixed flames. *Proceedings of the Combustion Institute*, *37*(2), 2249–2257. http://dx.doi.org/10.1016/j.proci.2018.06.209.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hoffman, M., Brochu, E., & de Freitas, N. (2011). Portfolio allocation for bayesian optimization. In *UAI'11, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence* (pp. 327–336).

Huhn, F., & Magri, L. (2020). Learning ergodic averages in chaotic systems. In *International Conference on Computational Science* (pp. 124–132). Springer.

Ishu, K., van der Zant, T., Becanovic, V., & Ploger, P. (2004). Identification of motion with echo state network. In *Oceans '04 MTS/IEEE Techno-Ocean '04 (IEEE Cat, No.04CH37600): Vol. 3* (pp. 1205–1210).

Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, *304*(5667), 78–80.

Jiang, J., & Lai, Y. -C. (2019). Model-free prediction of spatiotemporal dynamical systems with recurrent neural networks: Role of network spectral radius. *Physical Review Research*, *1*(3), Article 033056.

Kantz, H., & Schreiber, T. (2004). *Nonlinear time series analysis, Vol. 7*. Cambridge university press.

Kennedy, M., Rovatti, R., & Setti, G. (2000). *Chaotic electronics in telecommunications*. CRC press.

Kuznetsov, A., Kuznetsov, S., & Stankevich, N. (2010). A simple autonomous quasiperiodic self-oscillator. *Communications in Nonlinear Science and Numerical Simulation*, *15*(6), 1676–1681.

Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, *20*(2), 130–141.

Lorenz, E. N. (1996). Predictability: A problem partly solved. In *Proc. seminar on predictability*: *Vol. 1*.

Lu, Z., Hunt, B. R., & Ott, E. (2018). Attractor reconstruction by machine learning. *Chaos. An Interdisciplinary Journal of Nonlinear Science*, *28*(6), Article 061104.

Lu, Z., Pathak, J., Hunt, B., Girvan, M., Brockett, R., & Ott, E. (2017). Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos. An Interdisciplinary Journal of Nonlinear Science*, *27*(4), Article 041102.

Lukoševičius, M. (2012). A practical guide to applying echo state networks. In *Neural networks: Tricks of the trade* (pp. 659–686). Springer.

Lukoševičius, M., & Uselis, A. (2019). Efficient cross-validation of echo state networks. In *International conference on artificial neural networks* (pp. 121–133). Springer.

Lumley, J. L. (1967). The structure of inhomogeneous turbulent flows. In *Atmospheric turbulence and radio wave propagation*. Nauka.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, *14*(11), 2531–2560.

Matthies, H. G., & Meyer, M. (2003). Nonlinear Galerkin methods for the model reduction of nonlinear dynamical systems. *Computers and Structures*, *81*(12), 1277–1286.

Moon, F. C., & Shaw, S. W. (1983). Chaotic vibrations of a beam with non-linear boundary conditions. *International Journal of Non-Linear Mechanics*, *18*(6), 465–477.

Nakai, K., & Saiki, Y. (2018). Machine-learning inference of fluid variables from data using reservoir computing. *Physical Review E*, *98*(2), Article 023111.

Nastac, G., Labahn, J. W., Magri, L., & Ihme, M. (2017). Lyapunov Exponent as a metric for assessing the dynamic content and predictability of large-eddy simulations. *Physical Review Fluids*, *2*(9), Article 094606. http://dx.doi.org/10.1103/PhysRevFluids.2.094606.

Pathak, J., Hunt, B., Girvan, M., Lu, Z., & Ott, E. (2018). Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical Review Letters*, *120*(2), Article 024102.

Pathak, J., Lu, Z., Hunt, B. R., Girvan, M., & Ott, E. (2017). Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. *Chaos. An Interdisciplinary Journal of Nonlinear Science*, *27*(12), Article 121102.

Pathak, J., Wikner, A., Fussell, R., Chandra, S., Hunt, B. R., Girvan, M., et al. (2018). Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos. An Interdisciplinary Journal of Nonlinear Science*, *28*(4), Article 041101.

Racca, A., & Magri, L. (2021). Automatic-differentiated physics-informed echo state network (API-ESN). In *International Conference on Computational Science (accepted)*. arXiv preprint arXiv:2101.00002.

Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer school on machine learning* (pp. 63–71). Springer.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.

Sak, H., Senior, A. W., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems* (pp. 2951–2959).

Spearman, C. (1904). The proof and measurement of association between two things. *American Journal of Psychology*, *15*(1), 72–101, URL http://www.jstor.org/stable/1412159.

Stöckmann, H. -J. (2000). *Quantum chaos: An introduction*. American Association of Physics Teachers.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Q. Weinberger (Eds.), *27*, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., arXiv preprint arXiv:1409.3215.

Takens, F. (1981). Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980* (pp. 366–381). Springer.

Thiede, L. A., & Parlitz, U. (2019). Gradient based hyperparameter optimization in echo state networks. *Neural Networks*, *115*, 23–29.

Tikhonov, A. N., Goncharsky, A., Stepanov, V., & Yagola, A. G. (2013). *Numerical methods for the solution of ill-posed problems, Vol. 328*. Springer Science & Business Media.

Virtanen, P., & et al. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*, 261–272.

Viswanath, D. (1998). *Lyapunov exponents from random Fibonacci sequences to the lorenzequations*: *Tech. rep.*, Cornell University.

Vlachas, P. R., Byeon, W., Wan, Z. Y., Sapsis, T. P., & Koumoutsakos, P. (2018). Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of The Royal Society of London. Series A. Mathematical, Physical and Engineering Sciences*, *474*(2213), Article 20170844.

Vlachas, P. R., Pathak, J., Hunt, B. R., Sapsis, T. P., Girvan, M., Ott, E., et al. (2020). Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks*.

Wan, Z. Y., & Sapsis, T. P. (2018). Machine learning the kinematics of spherical particles in fluid flows. *Journal of Fluid Mechanics*, *857*.

Wang, H., & Yan, X. (2015). Optimizing the echo state network with a binary particle swarm optimization algorithm. *Knowledge-Based Systems*, *86*, 182–193.

Weiss, J. (2019). A tutorial on the proper orthogonal decomposition. In *AIAA aviation 2019 forum* (pp. 3333).

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, *1*(4), 339–356.

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

Wikner, A., Pathak, J., Hunt, B., Girvan, M., Arcomano, T., Szunyogh, I., et al. (2020). Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems. *Chaos. An Interdisciplinary Journal of Nonlinear Science*, *30*(5), Article 053111.

Yildiz, I. B., Jaeger, H., & Kiebel, S. J. (2012). Re-visiting the echo state property. *Neural Networks*, *35*, 1–9. http://dx.doi.org/10.1016/j.neunet.2012.07.005, URL https://www.sciencedirect.com/science/article/pii/S0893608012001852.

Yperman, J., & Becker, T. (2016). Bayesian optimization of hyper-parameters in reservoir computing. arXiv preprint arXiv:1611.05193.