Chaos, Solitons and Fractals 153 (2021) 111570

Contents lists available at ScienceDirect

Chaos, Solitons and Fractals

Nonlinear Science, and Nonequilibrium and Complex Phenomena

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

Frontiers

Forecasting of noisy chaotic systems with deep neural networks

Matteo Sangiorgio*, Fabio Dercole, Giorgio Guariso

Department of Electronics, Information and Bioengineering, Politecnico di Milano, Via Ponzio 34/5, Milan, I-20133, Italy

ARTICLE INFO

Article history:
Received 21 July 2021
Revised 20 October 2021
Accepted 24 October 2021

Keywords:
Recurrent neural networks
LSTM cell
Teacher forcing
Multi-step prediction
Deterministic chaos
Non-stationary processes

ABSTRACT

Recurrent neural networks have recently proved the state-of-the-art approach in forecasting complex oscillatory time series on a multi-step horizon. Researchers in the field investigated different machine learning techniques and training approaches on dynamical systems with different degrees of complexity. Still, these analyses are usually limited to noise-free chaotic time series. This paper extends the analysis from a deterministic to a noisy environment, by considering both observation and structural noise. Observation noise is evaluated by adding different levels of artificially-generated random values on deterministic processes obtained from the simulation of four archetypal chaotic systems. A case of structural noise is implemented through a time-varying version of the logistic map, which exhibits a slow structural change of the system's dynamic that makes the system non-stationary. Finally, a time series of ozone concentration in Northern Italy is considered to test the theoretical findings on a real-world case study in which both forms of noise play a significant role. Recurrent neural networks formed by LSTM cells are compared with two benchmark feed-forward architectures. LSTM trained without the standard teacher forcing approach, i.e., with training that replicates the setting used in inference mode, proved to have the best performance in compensating the stochasticity generated by the observation noise and reproducing the structural non-stationarity of the process.

1. Introduction

The concept of chaos, more precisely deterministic chaos, is usually linked to the concept of predictability, such as in the iconic sentence by Edward Lorenz: "Chaos: When the present determines the future, but the approximate present does not approximately determine the future". Indeed, arbitrarily small system's perturbations or measurement errors get amplified along a chaotic trajectory, even when produced by a deterministic yet nonlinear, dynamical system. The resulting time series are hardly predictable beyond the characteristic time of the local expansion and show complex oscillatory patterns.

Some early attempts to discover how far the evolution of chaotic systems can be predicted have been performed starting from the late 80s [1–9]. The topic became more and more debated in recent years, thanks to modern machine learning techniques and ever-increasing computational power [10].

Recurrent neural networks (RNNs, endowed with internal memory) proved to be particularly effective in forecasting nonlinear time series compared to traditional feed-forward (FF, memoryless) networks [11–14], though their training is computationally heavier [15]. In particular, two RNNs classes, namely reservoir computers [16–23] and long short-term memory (LSTM) networks [24–27], have proved to perform extremely well in predicting chaotic dynamics. Reservoir computers are composed of a single untrained hidden layer—the reservoir—of many randomly connected recurrent neurons, while training is only performed for the hidden-to-output connections. In LSTM nets, instead, all recurrent neurons are trained using the backpropagation algorithm (backpropagation through time of the internal memory). Additionally, the LSTM cell structure is specifically designed to prevent the converging issues of backpropagation in classical RNNs. An extensive comparison between the two classes focused on chaotic time series forecasting is provided in Hassanzadeh et al. [28] and Vlachas et al. [29].

Despite the high performances achieved on many different chaotic systems on the prediction one step ahead [26,30,31], the forecasting over a multi-step horizon is unequivocally recognized as an open issue and a challenging testing ground for machine learning techniques [32,33].

Depending on the length of the predictive horizon, we usually discriminate between two distinct typologies of problems. The first considers the forecasting of the system's dynamics on relatively short-term horizons. The second aims at replicating the so-called

* Corresponding author.
E-mail address: matteo.sangiorgio@polimi.it (M. Sangiorgio).

https://doi.org/10.1016/j.chaos.2021.111570

"climate" associated with the long-term behavior of the system. The focus is on the exact timing of the prediction in the first case, on the statistical properties of the original system in the second.

The papers on both topics usually study noise-free stationary time series, except a few recent exceptions [34–37]. For instance, Patel et al. [36] show that machine learning techniques are effective in reproducing the climate of chaotic attractors even in a noisy and non-stationary environment. In this paper, we extend in the same direction the results obtained by Sangiorgio and Dercole [27] in the short-mid-term forecasting of deterministic and stationary chaotic dynamics. They demonstrated that purely data-driven machine learning approaches can reproduce the dynamics of chaotic systems with high accuracy up to about six times the so-called Lyapunov time (LT, the characteristic time of the local expansion given by the inverse of the largest Lyapunov exponent, LLE, of the system's attractor). Since this limit has been found by different authors [17,20,29,38] in a wide range of continuous- and discrete-time systems of different degree of complexity and chaoticity, it can be considered as the predictability threshold nowadays achievable.

To extend these findings to a noisy environment, we analyze how the predictors' forecasting skills are affected by different sources of noise: observation and structural noise. The effect of observation noise is evaluated by adding different levels of artificially-generated random values, representing the observation uncertainty always affecting real-world applications on the deterministic processes obtained from the simulation of four archetypal chaotic systems. The structural noise is implemented through a time-varying version of the logistic map obtained by introducing a slow dynamic for the parameter defining the growth rate in the traditional formulation of such a map. This system has concurrent slow and fast dynamics and exhibits, in turn, a stable, periodic, or chaotic behavior. The two experiments stand somehow between the deterministic systems and real-world applications. They thus can be seen as an attempt to bridge the gap between theoretical and practical aspects of chaotic time series forecasting. Finally, we apply the proposed methodologies to a real-world time series of ozone concentration that exhibits chaotic behavior. Indeed, real time series constitute an intrinsic mixing of deterministic and stochastic components and when we assume the possibility of separating the two components we are, in a way, already defining a certain structure (e.g., an additive noise) of the signal and the process behind it. When this additive assumption is acceptable, the analysis can be split into two distinct steps: A filtering phase that aims at isolating deterministic and stochastic processes and a predictive phase that considers the deterministic signal [34,39].

Whenever it is not possible to separate the noisy component, the option is to directly consider the time series as it is (i.e., without preprocessing) in a single-step procedure, exploiting the neural networks ability of being nonlinear filters. Due to their basic structure, FF nets have a lower filtering ability than recurrent and convolutional architectures (the first thanks to the internal states; the second to convolutional filters). This ability can thus be boosted by combining convolutional and recurrent neurons [40] or by training the neural predictor, so that the parametrization can best exploit all the features of the input (including noise) as proposed in the paper by Chen et al. [35]. In practice, the effectiveness of all the techniques mentioned above depends on the considered task and their application could even negatively affect the forecasting accuracy. For instance, Romanuke [39] highlights that a preliminary smoothing can worsen the performance in the case of a noisy periodic dynamic, much easier to handle than a chaotic one.

The remainder of the paper is structured as follows. Section 2 presents the learning problem set up and describes the features of the considered neural predictors. Section 3 describes the experimental settings specifically thought to evaluate observation and structural noise, including a real-world case study. Finally, in Section 4, we recap the effect of different noise typologies and draw some concluding remarks. Appendix A provides the technical details about the training process and the hyper-parameter tuning. Appendix B reports the equations of the archetypal chaotic systems considered in this study.

2. Neural forecasting methods

In this section, we present the structure of the neural predictors and their identification procedure. To facilitate the comparison with the noise-free scenario, we use the very same setting introduced by Sangiorgio and Dercole [27]. Although some overlapping is unavoidable, we here adopt a more tutorial style for the reader's convenience.

2.1. Learning problem set up

Identifying a neural predictor for a time series $y(t)$ requires setting a supervised learning problem. The predictor uses the most recent m samples (lags), $y(t-m+1), \ldots, y(t-1), y(t)$, to generate the h predictions (leads), $\hat{y}(t+1), \hat{y}(t+2), \ldots, \hat{y}(t+h)$, corresponding to the targets $y(t+1), y(t+2), \ldots, y(t+h)$. In practice, the time series is arranged into N input-output pairs as shown in Fig. 1. Given the matrix of inputs (N rows and m columns), each predictor estimates the corresponding matrix of targets (N rows and h columns).

The learning problem is feasible if m is sufficient to map the predictor's input into the next target value $y(t+1)$. The smallest m is the dataset embedding dimension and is generically given by the integer larger than twice the fractal dimension of the process's attractor [41]. It should be pointed out that both FF and RNN predictors can solve the predictive problem presented above. The difference between them lies in the approach they use to handle the task: the former adopts a static scheme, the latter a dynamical one.

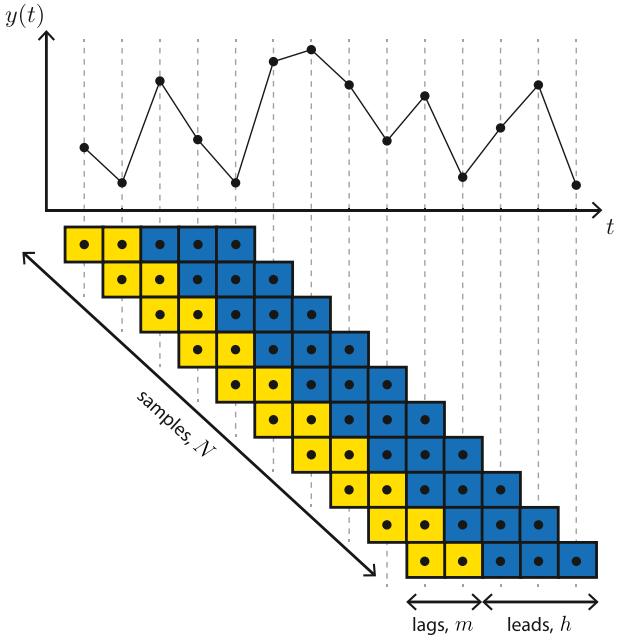

Fig. 1. The time series prediction is framed as a supervised learning task. For illustrative purposes, we represent N pairs of inputs ($m = 2$ lags, in yellow) and outputs ($h = 3$ leads, in blue). Note that the number of input-output pairs, N, is equal to the number of points in the time series minus $(m + h - 1)$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

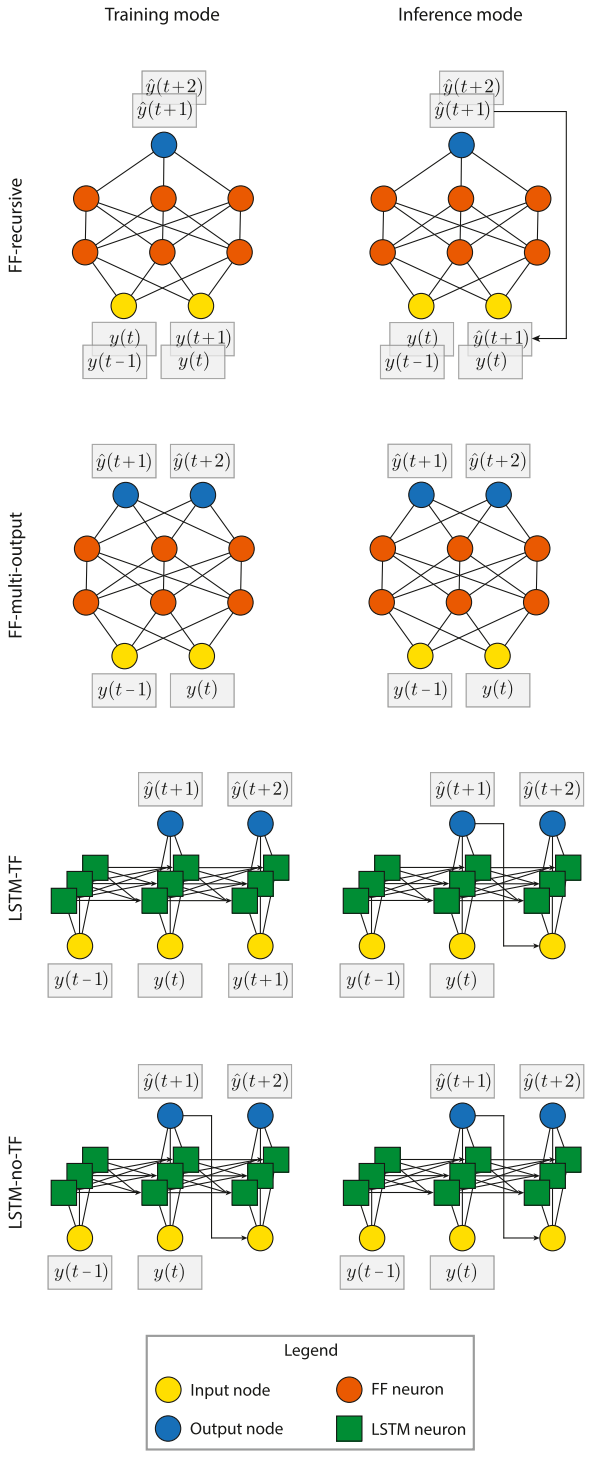

Fig. 2. The four considered neural predictors: FF recursive and multi-output, and LSTM trained with and without teacher forcing (TF, no-TF). Their functioning in training and inference modes is illustrated for $m = 2$ and $h = 2$ (FF architectures with 2 hidden layers of 3 neurons, LSTMs with a single layer of 3 cells). Note that the weights of recurrent neurons at different time steps are the same (parameter sharing). Consequently, the number of parameters does not depend on m and h for LSTM predictors.

2.2. Predictors' structure

We compare the neural architectures reported in Fig. 2 in the multi-step prediction of chaotic time series: two FF and an LSTM net (trained with two different approaches) that will be presented in the following. The structure of these neural predictors is intentionally maintained unchanged across time series of different complexity. This reduces the computational effort related to the hyperparameter tuning (see Appendix A), and allows to quantify how the predictive accuracy degrades with the complexity of the task.

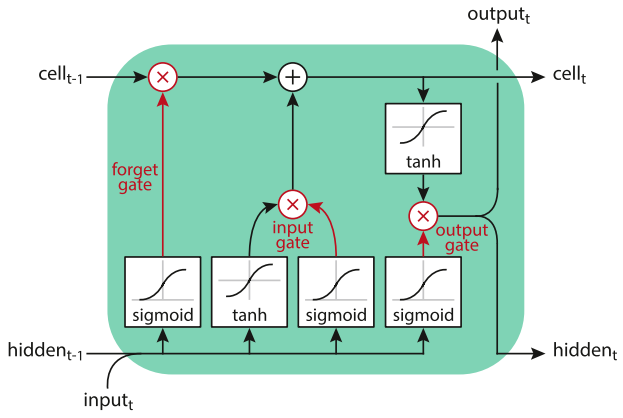

Fig. 3. Schematic diagram of an LSTM cell. The input, forget, and output gates are represented in red. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

FF-recursive predictor

The multi-step ahead forecasting of a chaotic dynamic is usually performed by identifying a single-step ahead predictor [42–46]. It is then used recursively, feeding each predicted value back as input for the subsequent step (FF-recursive predictor in Fig. 2). Despite the dynamic nature of the time series, the identification of such a predictor is a purely static task that requires reproducing the mapping from m inputs to a single output.

Once trained, the FF-recursive predictor can generate a sequence of outputs of arbitrary length. This is its main advantage, but, at the same time, its main weakness: the predictor is not explicitly optimized to deal with a multi-step forecasting, and thus its performance is not guaranteed on a multi-step horizon. This is a critical issue, especially when forecasting chaotic dynamics, due to their high sensitivity to perturbations.

FF-multi-output predictor

An alternative to the previous approach consists of training the neural net to predict the whole multiple-step horizon at once. A first attempt can be made by adopting the same structure used for the FF-recursive predictor, with the only difference of increasing the number of output neurons from 1 to h (FF-multi-output in Fig. 2). Each of these neurons computes the forecast of the variable at a specific time step neglecting the dynamic that links these values. Due to its static nature, this architecture acts as if the outputs were different systems variables at the same time step instead of the same variable sampled at different steps [47].

LSTM predictor

To overcome the critical aspects of FF-recursive and FF-multi-output nets, one should adopt an architecture that is optimized on a multiple-step horizon and explicitly considers the temporal dynamics. This can be done using recurrent neurons in the hidden layers of the net because their internal state gives them the ability to efficiently tackle tasks with a sequential/temporal structure.

In particular, we consider a specific type of recurrent neuron, the LSTM cell [15,48]. As depicted in Fig. 3, each LSTM cell has a two-dimensional state composed of the so-called hidden state, which is also the LSTM's output and corresponds to the state of a standard recurrent neuron and of the cell state, responsible for keeping track of the input's long-term effects.

Three gates modulate the information flows: input, output, and forget gates. Each gate's value is the output of a sigmoid in the range from 0 (closed gate) to 1 (open gate). The forget gate enables the LSTM neuron to reset the cell state at appropriate times (when the combination of input and hidden state into the sigmoid is low). Similarly, the input gate decides the extent to which the candidate cell state update (generated by a hyperbolic tangent) affects the cell state, protecting it from irrelevant contributions. The output gate modulates the candidate hidden state (generated by the other hyperbolic tangent) and avoids the negative effect of currently irrelevant memory contents. During training, this gated structure prevents vanishing (or exploding) gradients that generally affect convergence in traditional RNNs.

In our implementation, the internal states are initialized to zero at the beginning of each sequence of Fig. 1. This allows the parallelization of the training procedure and a fair comparison with the FF predictors.

The RNNs can be trained adopting two techniques: the so-called teacher forcing (TF), which represents the current standard, and the non-standard version without teacher forcing (no-TF). With the TF algorithm [49], the target data are fed as input at each time step in the future (LSTM-TF in Fig. 2). Thus, the net learns to forecast only one step, as each further prediction is based on the target data up to the previous step and the internal memory. In inference mode, however, each further prediction must be based on previous predictions so that the LSTM-TF predictor behaves differently with respect to the training mode (this discrepancy is known under the name of "exposure bias" in the neural networks literature [50,51]). Consequently, TF prevents the net from correcting its own errors, leading to a situation similar to that of the FF-recursive predictor. TF indeed turned out to be critical in the prediction of the noise-free chaotic dynamics [27].

Training the LSTM architecture without TF (LSTM-no-TF in Fig. 2) means that the network's behavior in training and inference modes coincide. In principle, it solves the drawbacks of the FF-recursive and multi-output predictors and of LSTM-TF. Indeed, LSTM-no-TF nets ranked first in all the noise-free experiments presented in Sangiorgio and Dercole [27].

Note that all the high-level libraries for deep learning use TF by default to train RNNs [52]. Implementing a training without TF requires coding with lower-level libraries, such as TensorFlow or PyTorch. This derives from the huge success of LSTMs in natural language processing, where TF proved to be necessary to limit the accumulation of errors especially in the initial phase of training [53].

2.3. Training, validation, and test

As a common practice in machine learning, we split the available data into training, validation, and test datasets, each organized as sequences of input-output pairs (Fig. 1). The training dataset is used to calibrate the network's parameters (i.e., connection weights and bias). The input-output pairs of the validation dataset are used to tune the network's hyper-parameters, which specify the neural architecture's configuration and the learning algorithm (details are summarized in Appendix A). Finally, the test dataset is held out and used, in inference mode, to evaluate the forecasting accuracy of the identified predictor objectively.

In the supervised learning framework, the training is performed by minimizing a loss function that measures the distance between the target values and the corresponding predictions. The metric traditionally adopted as the loss function is the mean squared error (MSE, hereafter). Consider N target samples $\mathbf{y} = \left[y_1, y_2, \ldots, y_N\right]$, and the corresponding i-step ahead predictions $\hat{\mathbf{y}}^{(i)} = \left[\hat{y}_1^{(i)}, \hat{y}_2^{(i)}, \ldots, \hat{y}_N^{(i)}\right]$. For each target sample y_k there is a time step t_k in the dataset such that $y_k = y(t_k + i)$, so that $\hat{y}_k^{(i)}$ is the i^{th} element of the output vector $\left[y(t_k+1),\ y(t_k+2),\ldots,\ y(t_k+h)\right]$ computed on the input $\left[y(t_k-m+1),\ldots,\ y(t_k-1),\ y(t_k)\right]$.

The MSE specific for the i^{th} step ahead is

$$\mathrm{MSE}\left(\mathbf{y}, \hat{\mathbf{y}}^{(i)}\right) = \frac{1}{N}\sum_{k=1}^{N}\left(y_k - \hat{y}_k^{(i)}\right)^2, \tag{1}$$

which is the loss function, for $i = 1$, used to train the FF-recursive predictor. To train the other multi-step predictors (FF-multi-output, LSTM-TF, and LSTM-no-TF), we use the average MSE over the entire predictive horizon as loss, i.e.,

$$\langle \mathrm{MSE} \rangle = \frac{1}{h}\sum_{i=1}^{h}\mathrm{MSE}\left(\mathbf{y}, \hat{\mathbf{y}}^{(i)}\right). \tag{2}$$

The training is performed for each of the considered combinations of the hyper-parameters (see Appendix A), selecting the one that provides the best performance—lowest loss—on the validation dataset.

Finally, we note that the MSE is not a suitable metric to test the obtained predictor, because the prediction quality corresponding to its values depends on the range of variability of the data. It is preferable to use a normalized metric, such as the R^2 score:

$$R^2\left(\mathbf{y}, \hat{\mathbf{y}}^{(i)}\right) = 1 - \frac{\mathrm{MSE}\left(\mathbf{y}, \hat{\mathbf{y}}^{(i)}\right)}{\mathrm{var}(\mathbf{y})}, \tag{3}$$

$$\left\langle R^2 \right\rangle = \frac{1}{h}\sum_{i=1}^{h} R^2\left(\mathbf{y}, \hat{\mathbf{y}}^{(i)}\right). \tag{4}$$

The R^2 score varies in the interval $(-\infty, 1]$, 1 corresponding to perfect forecasting. Two additional reference values are 0 and -1. At 0, the predictor behaves as the trivial one, always forecasting the data average $\bar{y}$ (note that $\mathrm{var}(\mathbf{y}) = \mathrm{MSE}\left(\mathbf{y}, \bar{y}\right)$). At -1, the target and predicted values are two uncorrelated samples from the data generator (so that the MSE is twice the variance), a situation that typically occurs for large i if the predictor correctly captures the long-term regime of the data [54].

3. Numerical experiments

3.1. Sensitivity to observation noise

To quantify the effect of the observation noise, we add a white Gaussian disturbance to the deterministic signals used in the noise-free case [27]. The noise-free signals are produced by simulating four archetypal chaotic systems in discrete time (logistic, Hénon, and two generalized Hénon maps), whose equations and main features are summarized in Appendix B. We examine three different noise levels, with standard deviation set to 0.5%, 1%, and 5% of the standard deviation of the noise-free signal.

We compare each noise-free case with the three corresponding noisy cases, using the same predictors' structure (in particular, same number of lags m), over the same multi-step predictive horizon (same number of lead h). The values of m and h for each chaotic system are reported in Table 1, together with the system's LT. The length of the predictive horizon is set to the first step i for which the R^2 score (Eq. 3) of the FF-recursive predictor becomes negative in the noise-free case.

The noise-free datasets are composed of 50,000 data points, split into training (70%), validation (15%), and test (15%). For each level of noise, each predictor is retrained on the noisy dataset. The results in testing are shown in the left column of Fig. 4 for the intermediate noise level (1%). Each panel is specific for one of the

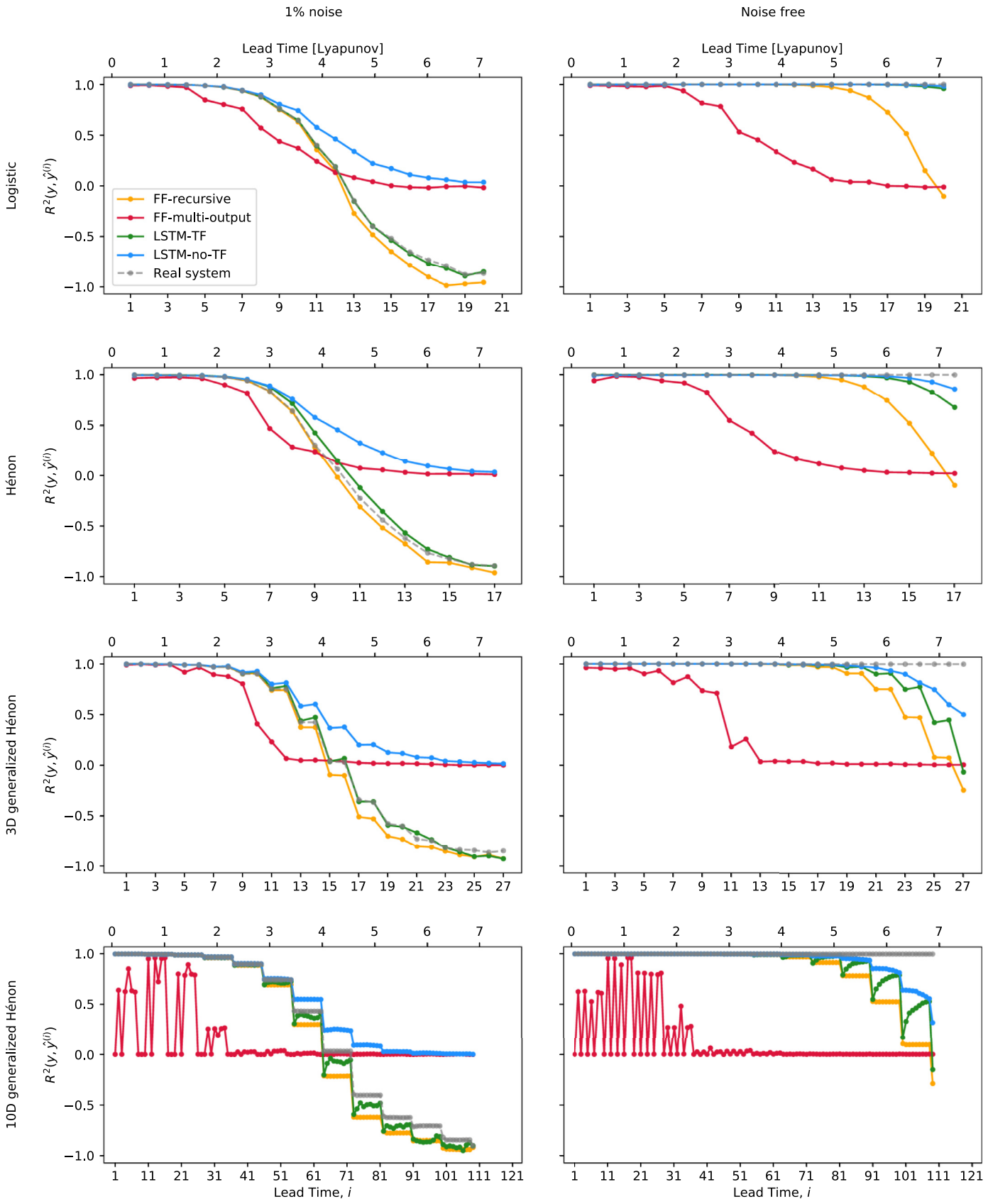

Fig. 4. $R^2(\mathbf{y}, \hat{\mathbf{y}}^{(i)})$ score of the four predictors (solid colored lines) and the real system used as a predictor (grey dashed line). Each row considers a different archetypal system. The first column reports the performances on the test dataset with a 1% noise level, the second on the noise-free test dataset.

Table 1
Features of the four archetypal chaotic systems.

System	Lags (m)	Leads (h)	LT
Logistic	1	20	2.83
Hénon	2	17	2.33
3D generalized Hénon	3	27	3.62
10D generalized Hénon	10	109	15.87

chaotic systems and shows the performances of the four considered predictors (solid colored lines) and the naive prediction obtained using the chaotic system itself as a predictor (real system, dashed grey line). The performances are presented in terms of the R^2 score for each step ahead of the predictive horizon, and the time scale in LT units is reported on top. To facilitate the comparison, we report the noise-free results in the right column.

As expected, the performances in the noisy cases are considerably worse than those obtained in the noise-free environment. This is due to the chaotic behavior of the considered dynamics, which results in predictions that unavoidably amplify the noise on the input data. The main result is that the performance ranking of the four predictors is robust to the observation noise. In particular, LSTM-no-TF ranks first, further confirming the importance of training recurrent architectures without TF that already emerged in the noise-free environment.

The FF-recursive predictor forecasts almost perfectly the evolution of the systems' dynamics over a horizon that is about halved with respect to the noise-free case. After that, its accuracy rapidly decays and the R^2 score becomes negative at about 4 LTs (with 1% noise). Note the peculiar step-wise behavior of the R^2 score in the 3D and 10D generalized Hénon maps. It is because the value $y(t+1)$ depends only on the two oldest inputs, i.e., $y(t-m+2)$ and $y(t-m+1)$ ($y(t-8)$ and $y(t-9)$ in the 10D map). The first $m-1$ predictions are therefore computed using the actual data as predictor's inputs. These $m-1$ predictions (that already contain a certain error) are then used to forecast of the time steps from m to $2\cdot(m-1)$. The same process occurs in the following blocks producing a significant drop of the accuracy every $m-1$ steps.

The FF-multi-output remains the worst predictor, even though it is the least sensitive to the observation noise. The reason lies in its static nature, which prevents the sequential expansion of the input noise. This is, however, the same reason for its poor performance, especially when forecasting too far ahead. In this situation, the iterated maps to be reproduced become too complex, and the network only learns to forecast the average value of the data (null R^2 score) [10]. Moreover, training the network on a long multi-step horizon, negatively affects also the predictions at the beginning of the horizon, as the same number of hidden neurons has to learn too many iterated maps independently. This is visible for the 10D generalized Hénon map (both noise-free and noisy cases), where only a few non-necessarily consecutive steps are forecasted with acceptable accuracy.

The LSTM-TF predictor is the most sensitive to observation noise. Its performances are significantly better than those of FF-recursive in the deterministic case, while they behave similarly in the presence of observation noise. The positive effect of the internal state propagation in the noise-free case seems to be lost in the noisy environment. The predictor essentially learns to predict one step only, and it is, therefore, subject to the same error exponential expansion characterizing the FF-recursive prediction.

Training the LSTM architecture without TF not only provides robustness to the system complexity but also to the presence of noise. Note, however, that the performance gain with respect to the other architectures is significantly reduced, even by a relatively small (1%) observation noise. The horizon of a precise forecast is more than halved with respect to the deterministic environment. Beyond that limit, the LSTM-no-TF, however, grants the least severe performance degradation.

The effect of training with or without TF is particularly marked in the 10D generalized Hénon map. With both the training approaches, the R^2 score has the step-wise behavior (with blocks of length $m-1$) discussed above. Within each block, the LSTM-TF shows a weird increasing trend. It suffers in the initial part of each block when the predictor uses the values predicted with a lower accuracy (this does not happen during training with TF since it always provides the actual input). The hidden state's internal dynamics then allow the predictor to recover (especially in the noise-free case) until a new drop of the R^2 score occurs at the beginning of the following block. Training the LSTM without TF prevents this problem, confirming that no-TF is necessary to teach the recurrent predictor to propagate the information with its internal dynamics correctly.

An interesting benchmark for the neural architectures is the performance obtained with the real systems generating the data used as predictors (dashed grey line in Fig. 4). In the noise-free case, it always provides a perfect prediction (R^2 score equal to one in the whole forecasting horizon). When considering noisy data, we are simulating the same dynamical system from two slightly different initial conditions. The two trajectories diverge due to the system's chaoticity. In other words, even when one can identify the real system perfectly, it is still not possible to prevent the multi-step error divergence.

The comparison between the neural networks and the real model of the system used as a predictor allows to evaluate the two components of the prediction error separately. The first is caused by the uncertainty in the identification process (identification error). The second is due to the propagation of the observation noise from the input data to the output (observation error). The neural predictors are affected by both sources of error, while the identification error is by definition null when considering the real system used as a predictor. The fact that the FF-recursive and LSTM-TF architectures show a predictive power similar to the real system used as a predictor confirms that these two neural predictors essentially solve a system identification task (they are optimized 1-step-ahead), rather than searching for the best multiple-step forecasting. Conversely, the LSTM-no-TF is specific for the considered horizon and, due to the way it is optimized, it provides better performance than the other competitors. We can conclude that the

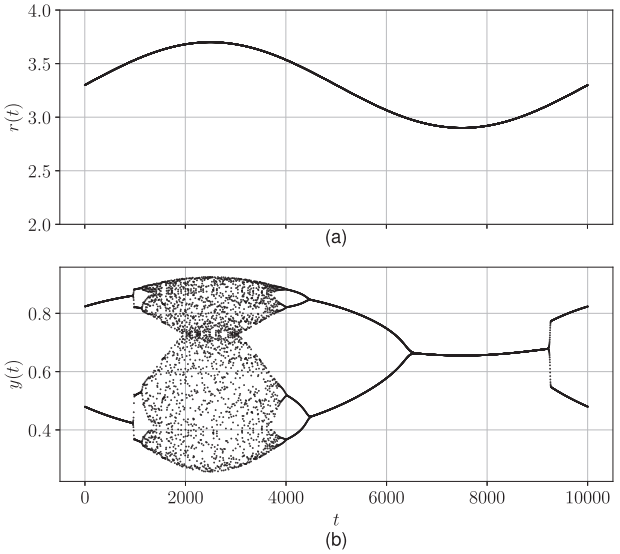

Fig. 5. Dynamics of the parameter $r(t)$ (a) and of the traditional logistic map variable $y(t)$ (b) for 10,000 steps.

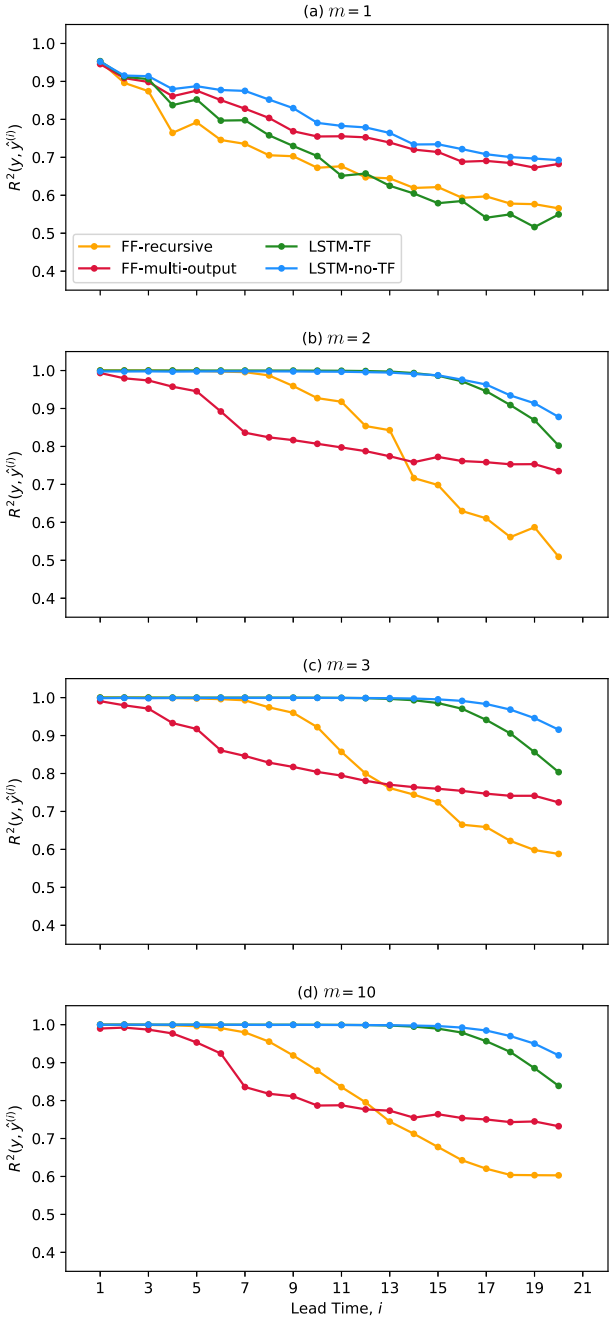

Fig. 6. $R^2\left(\mathbf{y}, \hat{\mathbf{y}}^{(i)}\right)$ score of the four predictors (colored lines) on the slow-fast logistic map for $m = 1$ (a), $m = 2$ (b), $m = 3$ (c), and $m = 10$ (d). Performances computed on the test dataset.

identification error is almost negligible compared to the observation error in forecasting tasks. This means that the knowledge of the actual system does not help to improve the predictive accuracy when dealing with a chaotic dataset affected by observation noise.

Table 2 reports the sensitivity analysis performed by testing the three levels of noise in terms of two metrics: the average R^2 score $\langle R^2 \rangle$ (Eq. 4) and the number of LTs for which the single-step R^2 score (Eq. 3) remains above 0.5, a threshold traditionally deemed acceptable in complex forecasting applications [55]. The table shows that the performance of the FF-recursive and LSTM-TF is similar to that of the real system used as a predictor also for 0.5% and 5% noise. For the two lower noise levels, the LSTM-no-TF predictor outperforms the FF-recursive and LSTM-TF competitors, confirming the better accuracy of predictors specifically trained for the multi-step forecasting over those optimized on the single-step only. The situation is substantially different with the 5% noise level: all the predictors provide almost identical accuracy. This is because such observation noise is so high that its propagation is predominant compared to the differences between the predictors.

An interesting finding of this study is the fairly uniform accuracy shown by each predictor (both with and without observation noise, see Fig. 4 and Table 2) across the different chaotic systems widely varying in complexity (their attractor fractal dimension spans from 0.5 to 9.13). The FF-recursive and LSTM-TF behave as the real system used as a predictor in all cases, even though the structural complexity of the latter is equivalent to the complexity of the task by definition. This is relevant because it proves that adopting the system's LT as time unit (horizontal axis) and the R^2 score as a metric (vertical axis) is an appropriate way to scale both the performances and the time horizon.

3.2. Effect of structural noise

The structural noise is implemented by introducing a slow (periodic) dynamic for the parameter $r(t)$ in the traditional logistic map. The resulting process is a slow-fast system. This kind of processes characterizes many natural phenomena and attracted the attention of many researchers in the field of dynamical systems theory (see, for instance, Rossetto et al. [56]).

The following equations define the slow-fast version of the logistic map:

$$\begin{cases} y(t+1) = r(t) \cdot y(t) \cdot \Big(1 - y(t)\Big) \\ r(t) = \frac{r_{max}+r_{min}}{2} + \frac{r_{max}-r_{min}}{2} \cdot \sin\left(\frac{t \cdot \pi}{5000}\right), \end{cases} \tag{5}$$

where $r_{\min} = 2.9$ and $r_{\max} = 3.7$ represent the lower and upper bound of the growth rate. These values are specifically selected so that the system exhibits, in turn, a stable, periodic or chaotic behavior, as shown in Fig. 5.

Evaluating the neural predictors' accuracy on this kind of system is interesting because the forecasting task requires retaining information about both the slow-varying context (long-term memory), and the fast dynamic of the logistic map. Correspondingly, the parabola which defines the mapping between $y(t)$ and $y(t+1)$ slowly changes in time. This task has different degrees of complexity: it is fairly simple when the system has a stable or periodic behavior and much more complex when the dynamic becomes chaotic.

The results obtained for different values of m (i.e., the number of lags fed as input to the predictor) are reported in Fig. 6.

When m is equal to one, the information is not sufficient to derive a mapping between the input and the output space (indeed, the embedding dimension is 2). Fig. 6a confirms that none of the predictors can reach an R^2 score close to one even on the one-step-ahead prediction. For values of m greater than one (Fig. 6b-d), the accuracy of all the predictors sensibly increases.

The results demonstrate that $m = 2$ is an appropriate embedding dimension for this system since the single-step forecast is almost perfect. In general, the recurrent structure of the LSTM nets provides better predictive accuracy than a feed-forward one (FF-recursive and FF-multi-output). This is because the LSTM architectures have a memory, i.e., their internal state is sequentially updated after processing each input. Such networks are thus better in reproducing the slow-varying dynamic of the parameter $r(t)$. Training without teacher forcing further improves the performance proving that LSTM-no-TF is the most powerful architecture among those considered in this study.

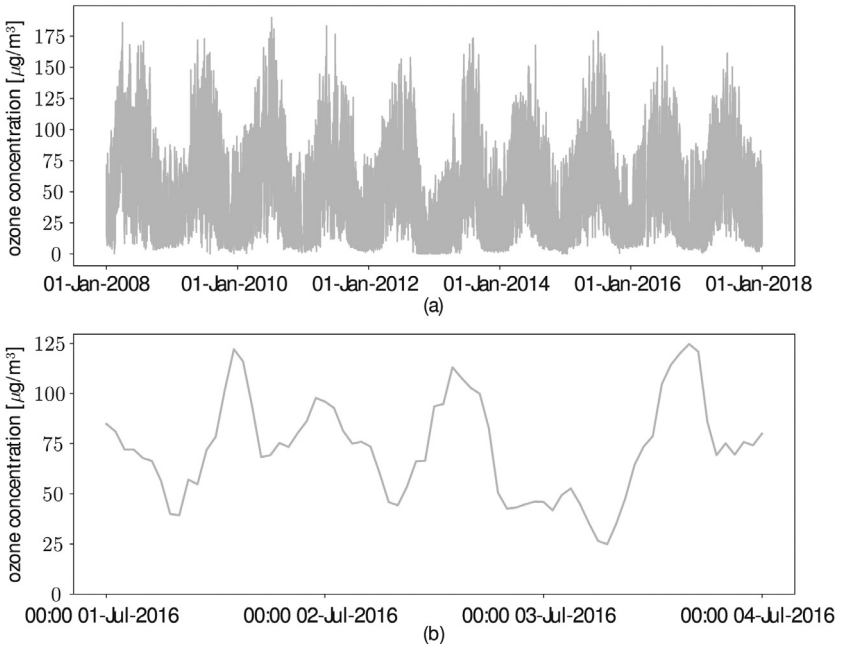

Fig. 7. Hourly ozone concentration time series for the 10-year period from 2008 to 2017 (a) and for a few specific days (b).

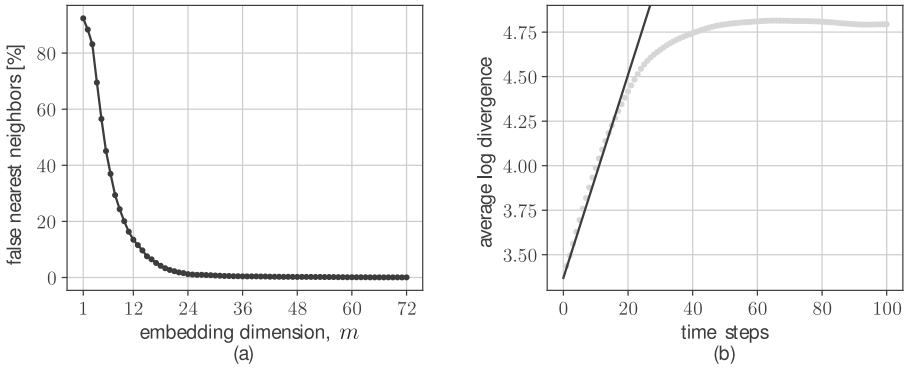

Fig. 8. Output of the false nearest neighbors algorithm (a) and estimation of the LLE = 0.057 (b) for the ozone concentration time series.

Table 2
Sensitivity analysis of the forecasting performance of the predictors and of the real system used as a predictor over the four chaotic systems. Three different levels of noise (0.5%, 1%, and 5%) are examined.

System	Predictor	$\langle R^2 \rangle$				LT $R^2 > 0.5$			
		noise free	0.5% noise	1% noise	5% noise	noise free	0.5% noise	1% noise	5% noise
Logistic	FF-recursive	0.85	0.36	0.18	- 0.16	6.37	4.25	3.54	2.12
	FF-multi-output	0.46	0.43	0.41	0.33	3.18	2.83	2.83	2.12
	LSTM-TF	> 0.99	0.41	0.23	- 0.19	> 7.07	4.25	3.54	2.12
	LSTM-no-TF	> 0.99	0.66	0.57	0.35	> 7.07	4.60	3.89	2.12
	Real system	1.00	0.41	0.24	- 0.21	∞	4.24	3.53	2.12
Hénon	FF-recursive	0.83	0.33	0.15	- 0.22	6.43	3.86	3.43	2.14
	FF-multi-output	0.43	0.42	0.41	0.33	3.00	3.00	2.57	2.14
	LSTM-TF	0.96	0.37	0.22	- 0.18	> 7.28	3.86	3.43	2.14
	LSTM-no-TF	0.98	0.63	0.56	0.36	> 7.28	4.29	3.86	2.57
	Real system	1.00	0.37	0.18	- 0.23	∞	3.86	3.43	2.16
3D generalized Hénon	FF-recursive	0.82	0.30	0.12	- 0.29	6.07	3.86	3.31	2.21
	FF-multi-output	0.35	0.35	0.35	0.29	2.76	2.76	2.48	2.21
	LSTM-TF	0.89	0.34	0.17	- 0.23	6.62	4.42	3.31	2.21
	LSTM-no-TF	0.94	0.61	0.53	0.33	> 7.45	4.42	3.86	2.21
	Real system	1.00	0.35	0.17	- 0.25	∞	4.42	3.31	2.21
10D generalized Hénon	FF-recursive	0.84	0.35	0.19	- 0.21	6.23	3.97	3.40	2.27
	FF-multi-output	0.13	0.12	0.13	0.12	0.00	0.01	0.00	0.01
	LSTM-TF	0.90	0.42	0.23	- 0.19	6.23	3.97	3.40	2.27
	LSTM-no-TF	0.94	0.63	0.54	0.32	6.80	4.54	3.97	2.27
	Real system	1.00	0.47	0.28	- 0.20	∞	4.54	3.40	2.27

3.3. Ozone concentration time series

A remarkable variety of natural phenomena are thought to be chaotic, from meteorology (the origin of Lorenz's studies) to the orbits of celestial bodies, to chemical reactions, to electronic circuits [57].

We evaluate the four neural predictors on a real-world dataset, considering an ozone (O_3) concentration time series recorded in Chiavenna, Italy, from 2008 to 2017 with an hourly time step. Such a dataset is maintained and made available by the regional environmental agency (ARPA) of the Lombardy region (https://www.arpalombardia.it).

Ground-level ozone formation is known to be a complex phenomenon. O_3 is a secondary pollutant: there are no ozone emissions, but its presence is due to chemical reactions involving other pollutants (called "precursors") and activated by ultraviolet radiation. This means ozone peaks are only during daylight, particularly where solar radiation is stronger, e.g., in the mountains. These photochemical reactions involve nitrogen oxides (NO_X), mainly emitted by road transport and domestic heating, and volatile organic compounds (VOC), due to industrial plants, road transport and agriculture. The formation process takes few hours, and thus the ozone concentration may reach high values at kilometers of distance from the precursors' sources, depending on the meteorological conditions.

The tropospheric ozone formation is known to be a nonlinear process [58], and thus represents an interesting study case to test the neural predictors presented above. A detail of the time series recorded in Chiavenna is shown in Fig. 7.

Like almost all the environmental processes, the ozone concentration has daily and annual periodic trends caused by the Earth's rotation on its axis and revolution around the Sun (Fig. 7).

To prove that ozone concentration can be considered the output of a chaotic system, it is necessary to check that the LLE is greater than 0. For the four archetypal systems presented above, we could compute the Lyapunov exponents with the traditional algorithm

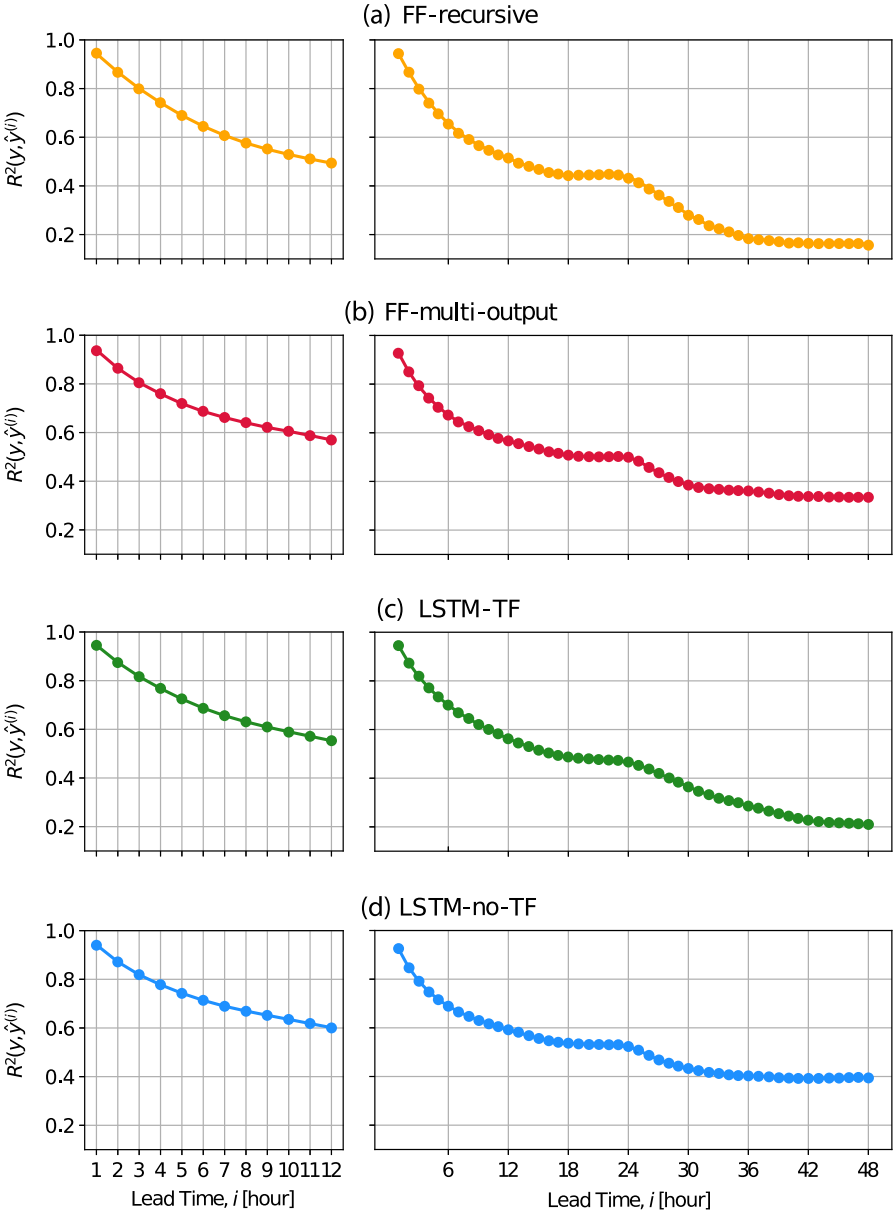

Fig. 9. $R^2(\mathbf{y}, \hat{\mathbf{y}}^{(i)})$ score of the four predictors on the ozone concentration time series. Performances computed on the test dataset (Chiavenna, 2016–2017) for $h = 12$ (first column), and $h = 48$ (second column).

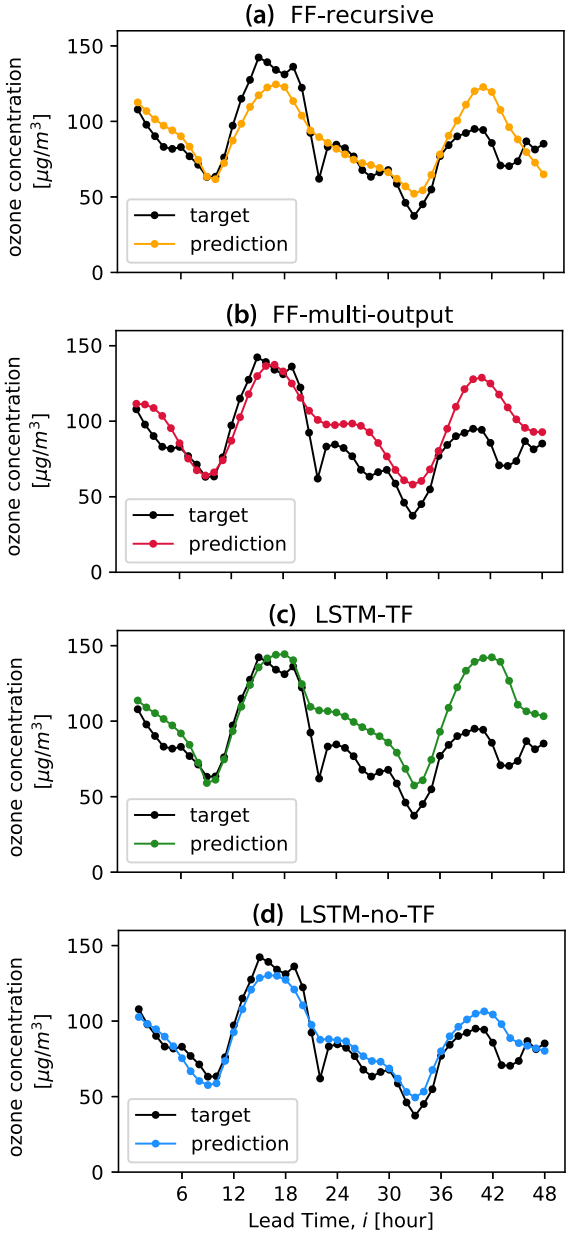

Fig. 10. Target sequence and corresponding predictions for a specific 2-day horizon ($h = 48$) from the ozone concentration test set.

based on the Jacobian matrix since state equations are known. On the contrary, when the model of the system is unknown, it is necessary to make use of statistical methods for detecting chaos [59].

A widely used approach requires, first, to evaluate a suitable embedding dimension for the dataset, using the false nearest neighbors algorithm [60]. The fraction of false neighbors becomes negligible after $m = 24$ (see Fig. 8a), meaning that such value is appropriate for reconstructing the system dynamics [41]. In other words, the finite embedding allows us to describe the time series as produced by a deterministic nonlinear process with the addition of an observation noise.

To check whether the process is chaotic, we estimate the LLE in the 24-dimensional space of delayed coordinates. Starting from similar windows of 24 consecutive data, the algorithm numerically computes the average exponential rate of divergence of the data that follow the considered windows [61]. The rate is obtained as the slope of the linear part of the log-divergence plot reported in Fig. 8(b).

The resulting LLE value, equal to 0.057, assesses the chaoticity of the series. Other analyses of ozone time series in different locations: Cincinnati, Ohio [62], Arosa, Switzerland [63], and Berlin, Germany [64], confirm that the ozone concentration dynamics can be characterized as a chaotic system.

As it is common habit in the field of environmental process forecasting, we compare the neural nets with the trivial persistent predictor: $\hat{y}(t+i) = y(t)$, $i = 1, 2, \ldots, h$ [47,65].

The comparison is performed using the data from 2008 to 2013 as the training set, 2014-15 for the validation, and 2016-17 for testing. As expected, the neural predictors widely outperform the persistent predictor that provides good performance only in the very short term (R^2 score: after one hour 0.90, two hours 0.78, three hours 0.63, six hours 0.26). The results obtained with the neural predictors for $h = 12$ and $h = 48$ are presented in Fig. 9. Considering the 12-hour horizon, the four predictors provide similar performances. A relevant gap emerges in the second half of the forecasting horizon between the LSTM-no-TF and the FF-recursive predictors: the 12-step ahead R^2 score is 0.60 for the former, and 0.49 for the latter. FF-multi-output and LSTM-TF ensure intermediate performances.

Increasing the length of the forecasting horizon to 48 steps (second column of Fig. 9), we spot the same ranking: LSTM-no-TF ensures the best performances, followed by FF-multi-output, LSTM-TF, and FF-recursive. In particular, it emerges that for a long forecasting horizon, it is crucial to optimize the predictor on the whole horizon (LSTM-no-TF and FF-multi-output) rather than on the single-step (FF-recursive and LSTM-TF). In the first case, the R^2 score tends to 0.4, while in the second to 0.2. Thus, the difference is remarkable even if, in all the cases, the R^2 scores are under the traditionally accepted lower bound of 0.5 [55].

From the qualitative point of view, all the predictors have a common trend. The performance is almost flat between 18 and 24 hours ahead, and then decreases again, reaching another plateau at the end of the horizon. This behavior is most probably because the ozone concentration dynamic in a given day is somewhat decoupled from the following day due to the absence of the ultraviolet radiation during nighttime.

Fig. 10 shows the trajectories predicted by the four neural nets in a specific case (the 2-day target is the same in all the panels). As expected, all the predictors have a better performance in the first part of the horizon and have considerable difficulties predicting the final part, except perhaps LSTM-no-TF that limits the maximum error to about 30% of the actual value at step 43.

The comparison reported in Fig. 10 shows that the prediction sequences are smoother than the observed data. This proves that the predictor is not modeling the high-frequency measurement noise that probably affects the dataset as in all real-world cases.

4. Discussion and conclusions

We compared the multi-step forecasting performance of four neural predictors, based on feed-forward and recurrent architectures, on artificial noisy time series generated by chaotic oscillators, and on a 10-year-long dataset of ozone concentration. We considered the same architectures used in a recent study of the noise-free case: two feed-forward (recursive and multi-output) and one recurrent (LSTM) [27]. This allowed us to precisely quantify the effect of different noise sources on artificial data and generalize these findings for real applications.

Regarding the LSTM nets, we considered two training approaches: the traditional teacher forcing, consisting of training the network using the target data from the previous time steps along the predictive horizon, and the case without teacher forcing, where

the network predictions are fed back as input for the following steps. Extending the analysis to artificial noisy systems and to a real-world environment, we confirmed that teacher forcing is not effective in the context of time series forecasting, as shown in Sangiorgio and Dercole [27] on noise-free artificial data.

In the context of artificial systems, we considered two different sources of noise. One is the traditional observation noise, which we model with an additive disturbance on the noise-free time series produced by four archetypal chaotic systems (logistic, Hénon and two generalized Hénon maps with low and high dimension). The other is a structural noise, i.e., a slow perturbation of the process generating the data, which we introduced through a modified version of the logistic map with a periodic dynamic for the growth-rate parameter.

The results obtained for different levels of observation noise confirmed the noise-free ranking. The FF-multi-output predictor remains the worst predictor and particularly suffers long multi-step horizons due to its fully static approach. The FF-recursive predictor guarantees accurate forecasting for approximately 5 LTs in the deterministic case and 2.5 LTs with a 1% noise level. The LSTM-TF is notably better than the FF-recursive in the noise-free cases, but the two predictors have almost identical performances in a noisy environment. The LSTM-no-TF is always the most performing predictor, proving its robustness to both the system complexity and the presence of noise. However, its horizon of perfect forecasting is more than halved with a 1% noise level. The fact that the predictors rank the same despite the different complexity of the considered systems proves the robustness of the results.

The results also show that the difference between the predictors' accuracy (except the FF-multi-output) rapidly reduces for increasing noise levels. For the considered systems, a noise level of 5% represents a sort of threshold beyond which there are no substantial differences between the predictors. A measurement affected by a 5% uncertainty is precise or inaccurate depending on the specific application (note that the noise level is unknown in practice), but the take-home message that emerges is probably general: the distances between different predictors are more prominent when high-quality datasets are available. Conversely, when we are aware that a given dataset is very inaccurate, it is reasonable to adopt a single easy and fast predictor instead of investing lots of resources in implementing many complex models.

The numerical experiments in a stochastic environment also allowed to prove that the error due to the model identification procedure is negligible if compared to that caused by the observation noise, even when this latter is limited (for instance, when its dispersion is in the order of 0.5% of the process standard deviation). In other words, having available the actual model of the system is of little use for the prediction of chaotic time series affected by observation noise.

Regarding the second typology of noise, we considered the structural non-stationarity of the system generating the data, which is another aspect usually affecting real-world time series. In our experiment, the slow dynamic is somewhat similar to the environmental variables' annual cycle. In contrast, the fast dynamic of the logistic may represent demographic oscillations in ecology or a rapid atmospheric process. The results showed that the gap between LSTM and FF predictors is quite broad, proving that the dynamic nature of recurrent architectures is more suitable than the static ones to model time-varying processes. LSTM nets also proved more robust with respect to the number of time lags included in the input. This feature represents a remarkable advantage given the well-known problem of estimating the actual embedding dimension of a real time series [66].

These results, on deterministic and noisy environments, have been obtained for convenience on discrete-time maps. However, we expect them to hold in a broader context, including continuous-time systems, as long as the issues related to the numerical integration are carefully handled, and real-world datasets (e.g., signals related to meteorology, fluid dynamics, biometry, econometrics) [67,68].

To evaluate how the situation changes when going from artificial to real-world systems, we considered a time series of ozone concentration that we showed to oscillate chaotically. The predictions of this time series are better with the FF-multi-output and LSTM-no-TF than with FF-recursive and LSTM-TF.

In this specific case, the training approach (optimized on the whole horizon, rather than on the single-step-ahead) is more important than the neural network structure (FF or LSTM). Therefore, one of the merits of this study is to clarify that the common practice of identifying a single-step model and then use it recursively is not the best choice under the perspective of forecasting. Expanding this idea to a broader context, we can conclude that prediction and system identification are indeed related but different tasks. FF-multi-output and LSTM-no-TF exhibit greater predictive power and robustness on short-mid-term forecasting but do not generally replicate the climate of the system's attractor satisfactorily. To this aim, it is preferable to use the FF-recursive, LSTM-TF or other models (such as the reservoir computers), which can mimic fairly well the long-term behavior of the chaotic systems [10,36].

Note that the three predictors that explicitly take into account a multi-step horizon (FF-multi-output and the two LSTMs) have been trained using the average performance on the forecasting horizon as the objective function. A weighted average that gives more importance to the errors in the first time steps could be used instead, e.g., when using the predictor within modern receding horizon control schemes that require implementing only the first control action and then plan again for the next horizon. Different weighting can be easily implemented in our framework. However, the actual costs and benefits of the distribution of the forecasting accuracy over the forecasting horizon are highly application-specific.

This paper represents the first attempt of systematically evaluating the effect of observation and structural noise on the short-mid-term forecasting of chaotic dynamics. It complements recent works [34,35] that investigate the effect of noise in Lorenz 96-like systems in the case in which the full state vector—or a large part of it—is accessible. This is a critical issue because the system's state is not accessible and even unknown in most applications. In such situations, we believe that neural predictors directly derived from the single (or the few) time series of interest provide more robust results to be transferred to real-world datasets.

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.

Appendix A. Hyper-parameters tuning and training algorithm

The structure of the neural architectures is the same for all the considered predictors and is formed by three hidden layers of 10 neurons each. For the two LSTM predictors, we used two LSTM hidden layers with one FF layer on top. However, this is not a critical issue because the same performances can be obtained adopting a different configuration with a single LSTM layer with two FF layers on top. Working with a fixed structure limits the number of hyper-parameters that have to be tuned, and allows to compare the performances obtained with different predictors.

The remaining hyper-parameters, which basically control the learning process, have been tuned using a traditional exhaustive search on a grid defined by the following values:

- batch size: 256, 512, and 1024;
- learning rate: 0.1, 0.01, and 0.001;
- learning rate decay factor: 0.001, and 0.

In order to avoid unfortunate initializations of the parameters, we repeated the optimization twice with different random values for each node of the grid.

The training has been performed using Adam (adaptive moment estimation) optimizer [69], a state-of-the-art algorithm for deep learning derived from the standard stochastic gradient descent algorithm. As suggested by its name, Adam adaptively estimates the first- and second-order moments of the gradients using running averages with decay rates equal to $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for the first and second moments, respectively. The estimates of the two moments are then used to evolve the learning rate during the training procedure.

We performed the training for 5000 epochs, a value that guarantees the convergence of the optimization in all the considered cases. Both training and validation losses are monitored during the learning process, so that the parametrization selected for a specific combination of the hyper-parameters is that having the best performance on the validation dataset (instead of the one obtained in the last epoch).

Appendix B. Archetypal chaotic systems

We consider four synthetic disrete-time dynamical systems, universally known for being chaotic, to test the predictive power of the various architectures: the logistic map, the Hénon map, and two versions of the generalized Hénon maps. They were adopted by Sangiorgio and Dercole [27] to study the noise-free case.

The logistic map is a simple quadratic map that describes the population growth of different species, and also a variety of processes in many scientific fields (e.g., economy and policy). It has a single state variable—the density of the considered population—that usually coincides with the output variable $y(t)$, so that the systems is defined by the following difference equation:

$$y(t+1) = r \cdot y(t) \cdot \Big(1 - y(t)\Big). \tag{B.1}$$

The positive parameter r, usually greater than 1, represents the growth rate of the population under ideal conditions (i.e., unlimited resources). Despite of its simplicity, the logistic map generates very complicated (chaotic) dynamics for almost all the values of r between 3.6 and 4. In the numerical experiment performed in this paper, we set $r = 3.7$.

The Hénon map and its generalized version introduced by Baier and Klein [70] can be defined in a generic n-dimensional case using the following set of state equations:

$$\begin{cases} x_1(t+1) = 1 - a\,x_{n-1}(t)^2 + x_n(t) \\ x_j(t+1) = x_{j-1}(t), \quad j = 2, \ldots, n-1 \\ x_n(t+1) = b\,x_{n-1}(t). \end{cases} \tag{B.2}$$

This system goes back to the formulation of the traditional Hénon map for $n = 2$, with parameters a and b equal to 1.4 and 0.3 as in the classic paper by Hénon [71].

To extend the analysis to hyperchaotic attractors (i.e., attractors with at least two positive Lyapunov exponents), we consider the generalized Hénon map with parameters $a = 1.9$ and $b = -0.03$. This map is specifically thought to have $n-1$ positive Lyapunov exponents [72]. A low-dimensional ($n = 3$) and a high-dimensional ($n = 10$) version of the generalized Hénon map are studied.

The system in Eq. (B.2) can be written as a nonlinear regression of order $m = n$ (m is thus the embedding dimension) taking the first state variable as output, i.e., $y(t) = x_1(t)$:

$$y(t+1) = 1 - a \cdot y(t-m+2)^2 + b \cdot y(t-m+1). \tag{B.3}$$

CRediT authorship contribution statement

Matteo Sangiorgio: Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing, Visualization. **Fabio Dercole:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Supervision. **Giorgio Guariso:** Conceptualization, Methodology, Writing – review & editing, Supervision.

References

[1] Farmer JD, Sidorowich JJ. Predicting chaotic time series. Phys Rev Lett 1987;59(8):845.
[2] Casdagli M. Nonlinear prediction of chaotic time series. Physica D 1989;35(3):335–56.
[3] Jones RD, Lee Y-C, Barnes C, Flake GW, Lee K, Lewis P, Qian S. Function approximation and time series prediction with neural networks. In: 1990 IJCNN international joint conference on neural networks. IEEE; 1990. p. 649–65.
[4] Principe JC, Rathie A, Kuo J-M. Prediction of chaotic time series with neural networks and the issue of dynamic modeling. Int J Bifurcation Chaos 1992;2(04):989–96.
[5] Principe JC, Kuo J-M. Dynamic modelling of chaotic time series with neural networks. In: Advances in neural information processing systems; 1995. p. 311–18.
[6] Navone H, Ceccatto H. Learning chaotic dynamics by neural networks. Chaos Solitons Fractals 1995;6:383–7.
[7] Verdes P, Granitto P, Navone H, Ceccatto H. Forecasting chaotic time series: global vs. local methods. Novel Intell Autom Control Sys 1998;1:129–45.
[8] Principe JC, Wang L, Kuo J-M. Non-linear dynamic modelling with neural networks. In: Signal analysis and prediction. Springer; 1998. p. 275–90.
[9] Shukla J. Predictability in the midst of chaos: a scientific basis for climate forecasting. Science 1998;282(5389):728–31.
[10] Sangiorgio M. Deep learning in multi-step forecasting of chaotic dynamics. Department of Electronics, Information and Bioengineering, Politecnico di Milano; 2021.
[11] Cannas B, Cincotti S, Marchesi M, Pilo F. Learning of chua's circuit attractors by locally recurrent neural networks. Chaos Solitons Fractals 2001;12(11):2109–15.
[12] Han M, Xi J, Xu S, Yin F-L. Prediction of chaotic time series based on the recurrent predictor neural network. IEEE Trans Signal Process 2004;52(12):3409–16.
[13] Cechin AL, Pechmann DR, de Oliveira LP. Optimizing markovian modeling of chaotic systems with recurrent neural networks. Chaos Solitons Fractals 2008;37(5):1317–27.
[14] Chandra R, Zhang M. Cooperative coevolution of elman recurrent neural networks for chaotic time series prediction. Neurocomputing 2012;86:116–23.
[15] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT press; 2016.
[16] Butcher JB, Verstraeten D, Schrauwen B, Day C, Haycock P. Reservoir computing and extreme learning machines for non-linear time-series data analysis. Neural Netw 2013;38:76–89.
[17] Pathak J, Lu Z, Hunt BR, Girvan M, Ott E. Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. Chaos: An Interdisciplinary J Nonlinear Sci 2017;27(12):121102.
[18] Lu Z, Pathak J, Hunt B, Girvan M, Brockett R, Ott E. Reservoir observers: model-free inference of unmeasured variables in chaotic systems. Chaos: An Interdisciplinary J Nonlinear Sci 2017;27(4):041102.
[19] Lu Z, Hunt BR, Ott E. Attractor reconstruction by machine learning. Chaos: An Interdisciplinary J Nonlinear Sci 2018;28(6):061104.
[20] Pathak J, Hunt B, Girvan M, Lu Z, Ott E. Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach. Phys Rev Lett 2018;120(2):024102.
[21] Jiang J, Lai Y-C. Model-free prediction of spatiotemporal dynamical systems with recurrent neural networks: role of network spectral radius. Phys rev res 2019;1(3):033056.
[22] Haluszczynski A, Räth C. Good and bad predictions: assessing and improving the replication of chaotic attractors by means of reservoir computing. Chaos: An Interdisciplinary J Nonlinear Sci 2019;29(10):103143.
[23] Fan H, Jiang J, Zhang C, Wang X, Lai Y-C. Long-term prediction of chaotic systems with machine learning. Phys Rev Res 2020;2(1):012080.
[24] Wan ZY, Vlachas P, Koumoutsakos P, Sapsis T. Data-assisted reduced-order modeling of extreme events in complex dynamical systems. PLoS ONE 2018;13(5):e0197704.
[25] Vlachas PR, Byeon W, Wan ZY, Sapsis TP, Koumoutsakos P. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. Proc Royal Soc A: Math Phys Eng Sci 2018;474(2213):20170844.
[26] Madondo M, Gibbons T. Learning and modeling chaos using lstm recurrent neural networks. In: MICS 2018 proceedings; 2018.
[27] Sangiorgio M, Dercole F. Robustness of LSTM neural networks for multi-step forecasting of chaotic time series. Chaos Solitons Fractals 2020;139:110045.
[28] Hassanzadeh P, Chattopadhyay A, Palem K, Subramanian D. Data-driven prediction of a multi-scale lorenz 96 chaotic system using a hierarchy of deep learning methods: reservoir computing, ann, and rnn-lstm. Bull. Am Phys Soc 2019.

[29] Vlachas P, Pathak J, Hunt B, Sapsis T, Girvan M, Ott E, Koumoutsakos P. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. Neural Netw 2020.
[30] Yang F-P, Lee S-J. Applying soft computing for forecasting chaotic time series. In: 2008 IEEE international conference on granular computing. IEEE; 2008. p. 718–23.
[31] Li Q, Lin R-C. A new approach for chaotic time series prediction using recurrent neural network. Math Problems Eng 2016;2016.
[32] Hussein S, Chandra R, Sharma A. Multi-step-ahead chaotic time series prediction using coevolutionary recurrent neural networks. In: 2016 IEEE congress on evolutionary computation (CEC). IEEE; 2016. p. 3084–91.
[33] Mariet Z, Kuznetsov V. Foundations of sequence-to-sequence modeling for time series. In: The 22nd international conference on artificial intelligence and statistics; 2019. p. 408–17.
[34] Brajard J, Carrassi A, Bocquet M, Bertino L. Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: a case study with the lorenz 96 model. J Comput Sci 2020;44:101171.
[35] Chen P, Liu R, Aihara K, Chen L. Autoreservoir computing for multistep ahead prediction based on the spatiotemporal information transformation. Nat Commun 2020;11(1):1–15.
[36] Patel D, Canaday D, Girvan M, Pomerance A, Ott E. Using machine learning to predict statistical properties of non-stationary dynamical processes: system climate, regime transitions, and the effect of stochasticity. Chaos: An Interdisciplinary J Nonlinear Sci 2021;31(3):033149.
[37] Sangiorgio M, Dercole F, Guariso G. Sensitivity of chaotic dynamics prediction to observation noise. IFAC-PapersOnLine 2021;54.
[38] Pathak J, Wikner A, Fussell R, Chandra S, Hunt BR, Girvan M, et al. Hybrid forecasting of chaotic processes: using machine learning in conjunction with a knowledge-based model. Chaos: An Interdisciplinary J Nonlinear Sci 2018;28(4):041101.
[39] Romanuke V. Time series smoothing improving forecasting. Appl Comp Sys 2021;26(1):60–70.
[40] Cheng W, Wang Y, Peng Z, Ren X, Shuai Y, Zang S, Liu H, Cheng H, Wu J. High-efficiency chaotic time series prediction based on time convolution neural network. Chaos, Solitons & Fractals 2021;152:111304.
[41] Takens F. Detecting strange attractors in turbulence. In: Dynamical systems and turbulence, Warwick 1980. Springer; 1981. p. 366–81.
[42] Bakker R, Schouten JC, Giles CL, Takens F, Bleek CMvd. Learning chaotic attractors by neural networks. Neural Comput 2000;12(10):2355–83.
[43] Galván IM, Isasi P. Multi-step learning rule for recurrent neural models: an application to time series forecasting. Neural Process Lett 2001;13(2):115–33.
[44] Lim TP, Puthusserypady S. Error criteria for cross validation in the context of chaotic time series prediction. Chaos: An Interdisciplinary J Nonlinear Sci 2006;16(1):013106.
[45] Wu X, Wang Y, Mao J, Du Z, Li C. Multi-step prediction of time series with random missing data. Appl Math Model 2014;38(14):3512–22.
[46] Shi X, Feng Y, Zeng J, Chen K. Chaos time-series prediction based on an improved recursive levenberg–marquardt algorithm. Chaos Solitons Fractals 2017;100:57–61.
[47] Guariso G, Nunnari G, Sangiorgio M. Multi-step solar irradiance forecasting and domain adaptation of deep neural networks. Energies 2020;13(15):3987.
[48] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80.
[49] Williams RJ, Zipser D. A learning algorithm for continually running fully recurrent neural networks. Neural Comput 1989;1(2):270–80.
[50] Ranzato M, Chopra S, Auli M, Zaremba W. Sequence level training with recurrent neural networks. arXiv preprint arXiv:151106732 2015.
[51] He T, Zhang J, Zhou Z, Glass J. Quantifying exposure bias for neural language generation. arXiv preprint arXiv:190510617 2019.
[52] Mihaylova T, Martins AF. Scheduled sampling for transformers. arXiv preprint arXiv:190607651 2019.
[53] Bengio S, Vinyals O, Jaitly N, Shazeer N. Scheduled sampling for sequence prediction with recurrent neural networks. In: Advances in neural information processing systems; 2015. p. 1171–9.
[54] Dercole F, Sangiorgio M, Schmirander Y. An empirical assessment of the universality of ANNs to predict oscillatory time series. IFAC-PapersOnLine 2020;53(2):1255–60.
[55] Santhi C, Arnold JG, Williams JR, Dugas WA, Srinivasan R, Hauck LM. Validation of the swat model on a large river basin with point and nonpoint sources. JAWRA 2001;37(5):1169–88.
[56] Rossetto B, Lenzini T, Ramdani S, Suchey G. Slow-fast autonomous dynamical systems. Int J Bifurcation Chaos 1998;8(11):2135–45.
[57] Van Truc N, Anh DT. Chaotic time series prediction using radial basis function networks. In: 2018 4th international conference on green technology and sustainable development (GTSD). IEEE; 2018. p. 753–8.
[58] Lin X, Trainer M, Liu S. On the nonlinearity of the tropospheric ozone production. J Geophy Res: Atmos 1988;93(D12):15879–88.
[59] Ellner S, Turchin P. Chaos in a noisy world: new methods and evidence from time-series analysis. Am Nat 1995;145(3):343–75.
[60] Kennel MB, Brown R, Abarbanel HD. Determining embedding dimension for phase-space reconstruction using a geometrical construction. Phys Rev A 1992;45(6):3403.
[61] Wolf A, Swift JB, Swinney HL, Vastano JA. Determining lyapunov exponents from a time series. Physica D 1985;16(3):285–317.
[62] Chen J-L, Islam S, Biswas P. Nonlinear dynamics of hourly ozone concentrations: nonparametric short term prediction. Atmos Environ 1998;32(11):1839–48.
[63] Chattopadhyay G, Chattopadhyay S. A probe into the chaotic nature of total ozone time series by correlation dimension method. Soft Comput 2008;12(10):1007–12.
[64] Haase P, Schlink U, Richter M. Non-parametric short-term prediction of ozone concentration in berlin: preconditions and justification. In: Air pollution modelling and simulation. Springer; 2002. p. 527–36.
[65] Meyer PG, Kantz H, Zhou Y. Characterizing variability and predictability for air pollutants with stochastic models. Chaos: An Interdisciplinary J Nonlinear Sci 2021;31(3):033148.
[66] Bradley E, Kantz H. Nonlinear time-series analysis revisited. Chaos: An Interdisciplinary J Nonlinear Sci 2015;25(9):097610.
[67] Matsumoto T, Nakajima Y, Saito M, Sugi J, Hamagishi H. Reconstructions and predictions of nonlinear dynamical systems: a hierarchical bayesian approach. IEEE Trans Signal Process 2001;49(9):2138–55.
[68] Siek M. Predicting storm surges: chaos, computational intelligence, data assimilation and ensembles: UNESCO-IHE PhD thesis. CRC Press; 2011.
[69] Kingma DP, Ba J. Adam: a method for stochastic optimization. arXiv preprint arXiv:14126980 2014.
[70] Baier G, Klein M. Maximum hyperchaos in generalized hénon maps. Phys Lett A 1990;151(6–7):281–4.
[71] Hénon M. A two-dimensional mapping with a strange attractor. In: The Theory of Chaotic Attractors. Springer; 1976. p. 94–102.
[72] Richter H. The generalized henon maps: examples for higher-dimensional chaos. Int J Bifurcation Chaos 2002;12(06):1371–84.