Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 54-17 (2021) 129–134

Sensitivity of Chaotic Dynamics Prediction to Observation Noise

Matteo Sangiorgio [*] **Fabio Dercole** [*] **Giorgio Guariso** [*]

[*] *Department of Electronics, Information, and Bioengineering, Politecnico di Milano, I-20133 Italy (e-mail: matteo.sangiorgio, fabio.dercole, giorgio.guariso @polimi.it).*

Abstract: Recent advances in nonlinear time-series prediction demonstrated the ability of recurrent neural network to forecast chaotic time series on a multi-step horizon, outperforming previous approaches. Researches considered chaotic systems with different degree of complexity, but the analysis was mainly limited to the noise-free case. In this work, we extend the analysis to a noisy environment, in order to fill the gap between deterministic and real-world time series. We consider four archetypal deterministic chaotic systems each with different levels of additive noise, representing the observation uncertainty always affecting practical applications. A time series of solar irradiance is also taken into account as a real-world case study. Various neural architectures, including feed-forward and recurrent networks, are adopted as predictors. LSTM cells are used as recurrent neurons, with a special focus on the training approach. As in the noise-free case, LSTM trained without the traditional teacher forcing, i.e., with a training that replicates the forecasting conditions, proved to be the best architecture. The experiments on the archetypal systems also shows that the error due to the model identification is negligible if compared to the one caused by a small observation noise. In other words, system identification and predictions are well distinct tasks.

Keywords: Forecasting, Fuzzy and neural systems relevant to control and identification, Modeling and identification of environmental systems, Machine learning for environmental applications, Deterministic chaos, Recurrent neural networks.

1. INTRODUCTION

Machine learning techniques have recently emerged as efficient tools in the context of the data-driven prediction of chaotic systems. In particular, recurrent neural networks (RNNs) achieved significant results in terms of forecasting accuracy and robustness when trained with backpropagation algorithms as well as following the reservoir computing approach (Pathak et al., 2018; Sangiorgio and Dercole, 2020; Vlachas et al., 2020).

The works in the field of chaotic dynamics forecasting, besides few recent exceptions (Brajard et al., 2020; Chen et al., 2020), usually study only noise-free time series. Under this assumption, machine learning approaches can reproduce with high accuracy the dynamics of chaotic systems up to 6-7 Lyapunov times. Since this limit has been found in a wide range of systems of different degree of complexity and chaoticity, both in continuous and discrete time, it can be somehow considered as the predictability threshold nowadays achievable.

In this work, we analyze how the forecasting skills of four feed-forward and recurrent neural predictors are affected by additive noise, which mimics the role of observation uncertainty always involved in practical applications. In particular, we superimpose different levels of artificially-generated white noise to the time series. The experiment is performed on four time series obtained from archetypal chaotic systems, which have been analyzed in the noise-free case in Sangiorgio and Dercole (2020).

Finally, we evaluate the neural predictors on a real-world application: a solar irradiance time series whose behavior has been proved to be chaotic.

The rest of this paper is organized as follows. In Section 2 we present the neural architectures used as predictors. Section 3 investigates the sensitivity of the predictive performance in presence of an increasing level of noise on the archetypal chaotic systems. In Section 4 we report the results obtained on the solar irradiance time series. Section 5 summarizes the contributions of this work and draws some concluding remarks.

2. NEURAL PREDICTORS

The problem of forecasting a time series $y(t)$ on a multi-step horizon consists of identifying a predictor that takes as input the m lags, $y(t-m+1), \ldots, y(t-1), y(t)$, and produces as output the forecasted values for the subsequent h leads, $\hat{y}(t+1), \hat{y}(t+2), \ldots, \hat{y}(t+h)$ corresponding to the target values $y(t+1), y(t+2), \ldots, y(t+h)$.

The learning problem is feasible if the number m of lags—the autoregressive terms that the model takes as input—is large enough to establish a relation between the input and the one-step target $\hat{y}(t+1)$ (Takens, 1981). In the nonlinear system terminology, the minimum of such m's is the dataset embedding dimension.

10.1016/j.ifacol.2021.11.037

We focus on neural architectures either composed of purely feed-forward (FF) layers, or with the addition of recurrent neurons. The former try to reproduce the mapping between the input and output sets in a static way, the latter tackle the problem under a dynamical perspective. In both cases, it is necessary to frame the time series, $y(1), y(2), \dots, y(t), \dots$, in N pairs of input and output vectors, $\big[y(t-m+1), \dots, y(t-1), y(t)\big]$ and $\big[y(t+1), y(t+2), \dots, y(t+h)\big]$. The learning task is thus to reproduce the mapping between the m-dimensional input space and the h-dimensional output space minimizing a loss function which measures the distance between target and predicted output values.

The traditional loss function in regression tasks is the mean squared error (MSE). We consider N target samples $\mathbf{y} = \big[y_1, y_2, \dots, y_N\big]$, and the corresponding i-step ahead predictions $\hat{\mathbf{y}}^{(i)} = \big[\hat{y}_1^{(i)}, \hat{y}_2^{(i)}, \dots, \hat{y}_N^{(i)}\big]$. 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 $\big[y(t_k+1), y(t_k+2), \dots, y(t_k+h)\big]$ computed on the input $\big[y(t_k-m+1), \dots, y(t_k-1), y(t_k)\big]$.

The i-step-ahead MSE is then defined as:

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

However, the MSE is not a suitable performance metric, since it is not normalized to the variability of the data. To assess the prediction quality, it is preferable to use a relative metric, such as the R^2 score:

$$R^2\big(\mathbf{y}, \hat{\mathbf{y}}^{(i)}\big) = 1 - \frac{\sum_{k=1}^{N}\big(y_k - \hat{y}_k^{(i)}\big)^2}{\sum_{k=1}^{N}\big(y_k - \bar{y}\big)^2}, \tag{2}$$

where $\bar{y}$ is the mean of the target data. The R^2 score is a normalized version of the MSE. It varies in the range $(-\infty, 1]$, with the upper bound corresponding to a perfect forecasting, and 0 to the predictive power of the trivial model which always forecasts the data mean value.

Both MSE and R^2 score, defined for the i^{th} step ahead can be averaged over the entire forecasting horizon:

$$\begin{aligned}\langle\text{MSE}\rangle &= \frac{1}{h}\sum_{i=1}^{h}\text{MSE}\big(\mathbf{y}, \hat{\mathbf{y}}^{(i)}\big),\\ \langle R^2\rangle &= \frac{1}{h}\sum_{i=1}^{h}R^2\big(\mathbf{y}, \hat{\mathbf{y}}^{(i)}\big),\end{aligned} \tag{3}$$

In the context of chaotic data forecasting, the predictive horizon is usually rescaled in terms of the Lyapunov time (LT)—the inverse of the largest Lyapunov exponent (LLE) measuring the average exponential rate of divergence of nearby trajectories. This allows the fair comparison of the predictive power among both artificial and real systems with different degrees of chaoticity.

2.1 Feed-forward architectures

The simplest and most common forecasting approach consists of identifying the best one-step predictor and then use it in a recursive way, feeding the previous step predictions as input for the subsequent steps. Despite the dynamic nature of the time series, this identification is a static task, which requires to reproduce the relation from the m lags to a single output. It can therefore solved with a feed-forward network (see Table 1, FF-recursive).

The main advantage of this approach is that, once the one-step predictor is trained, it can be used recursively to forecast an arbitrarily long future sequence. This is, at the same time, the main drawback of the FF-recursive approach, as it is not optimized for a multi-step prediction.

To this purpose, it is necessary to define a model that predicts multiple values. Within the FF class, we can use the same structure used for the one-step predictor, with the only difference of increasing, from 1 to h, the number of nodes in the output layer. Each output neuron focuses on the prediction at a specific time step ahead (see Table 1, FF-multi-output). The main issue with this architecture is that it does not take into account that the outputs are sequential—i.e., the same variable at different time steps—but it acts as if the outputs were different variables (Guariso et al., 2020).

2.2 Recurrent architectures

The idea behind RNNs is to explicitly take into account the sequentiality of the time series. To do so, the hidden nodes are endowed with an internal state that is recurrently updated using the sequence of the m lags and the predictions up to $i-1$ steps ahead to sequentially contribute to the prediction at step i, $i \geq 2$. We consider, in particular, the widely used LSTM cells (Hochreiter and Schmidhuber, 1997), advanced recurrent cells in which the input-to-state, state-to-state, and state-to-output contributions are regulated by extra internal variables, called gates, whose dynamics are optimized during training.

This architecture potentially overcomes the limitations of the FF-recursive and multi-output predictors, because it is trained to reproduce the entire sequence of output variables and explicitly takes into account that these h outputs are sequential values of the same variable (Sangiorgio, 2021).

The issue with RNNs is that they are systematically trained using a technique known as teacher forcing (TF) (Williams and Zipser, 1989). It consists of using the target data as input, rather than the predictions produced by the network from 1 to $i-1$ steps ahead, to compute the loss of the prediction at step i (see Table 1, LSTM-TF).

In inference mode, the target data are obviously unknown beyond the current time, so that the RNN network feeds back its own predictions to complete the horizon of $h \geq 2$ leads. TF therefore introduces a discrepancy between training and inference, that is known as exposure bias.

TF proved to be efficient on natural language processing related tasks, granting faster convergence and avoiding the accumulation of errors in the initial phase of training (Bengio et al., 2015). For these reasons, all the high-level application programming interfaces (API) for deep learning adopt TF by default (Mihaylova and Martins, 2019), whereas implementing a training without TF requires to code with a low-level API, such as TensorFlow or PyTorch.

However, TF is not a good idea in the context of time series forecasting (Sangiorgio and Dercole, 2020). Basically, training with TF is somehow analogue to train a FF-

Table 1. Training and inference phases of the four considered neural predictors for $m = 2$ and $h = 2$. $y(t)$ is the variable to be predicted, and $s(t)$ the internal state of the recurrent architecture. $s(t-m)$ is the initial internal state vector, whose elements are set to 0.

Predictor	Training phase	Inference phase
FF-recursive	$\hat{y}(t+1) = f_{\text{FF-rec}}\big(y(t), y(t-1)\big)$	$\hat{y}(t+1) = f_{\text{FF-rec}}\big(y(t), y(t-1)\big)$
	$\hat{y}(t+2) = f_{\text{FF-rec}}\big(y(t+1), y(t)\big)$	$\hat{y}(t+2) = f_{\text{FF-rec}}\big(\hat{y}(t+1), y(t)\big)$
FF-multi-output	$\big[\hat{y}(t+2), \hat{y}(t+1)\big] = f_{\text{FF-mo}}\big(y(t), y(t-1)\big)$	$\big[\hat{y}(t+2), \hat{y}(t+1)\big] = f_{\text{FF-mo}}\big(y(t), y(t-1)\big)$
LSTM-TF	$\big[\hat{y}(t), s(t-1)\big] = f_{\text{LSTM-TF}}\big(y(t-1), s(t-2)\big)$	$\big[\hat{y}(t), s(t-1)\big] = f_{\text{LSTM-TF}}\big(y(t-1), s(t-2)\big)$
	$\big[\hat{y}(t+1), s(t)\big] = f_{\text{LSTM-TF}}\big(y(t), s(t-1)\big)$	$\big[\hat{y}(t+1), s(t)\big] = f_{\text{LSTM-TF}}\big(y(t), s(t-1)\big)$
	$\big[\hat{y}(t+2), s(t+1)\big] = f_{\text{LSTM-TF}}\big(y(t+1), s(t)\big)$	$\big[\hat{y}(t+2), s(t+1)\big] = f_{\text{LSTM-TF}}\big(\hat{y}(t+1), s(t)\big)$
LSTM-no-TF	$\big[\hat{y}(t), s(t-1)\big] = f_{\text{LSTM-no-TF}}\big(y(t-1), s(t-2)\big)$	$\big[\hat{y}(t), s(t-1)\big] = f_{\text{LSTM-no-TF}}\big(y(t-1), s(t-2)\big)$
	$\big[\hat{y}(t+1), s(t)\big] = f_{\text{LSTM-no-TF}}\big(y(t), s(t-1)\big)$	$\big[\hat{y}(t+1), s(t)\big] = f_{\text{LSTM-no-TF}}\big(y(t), s(t-1)\big)$
	$\big[\hat{y}(t+2), s(t+1)\big] = f_{\text{LSTM-no-TF}}\big(\hat{y}(t+1), s(t)\big)$	$\big[\hat{y}(t+2), s(t+1)\big] = f_{\text{LSTM-no-TF}}\big(\hat{y}(t+1), s(t)\big)$

recursive predictor, as the optimized losses essentially contain only one-step-ahead predictions (more precisely, predictions obtained with input target values up to the previous step, though with the contribution of the internal state). Similar to FF-recursive predictors, TF therefore suffers the accumulation of errors along the sequence of predictions (Ranzato et al., 2015; Bengio et al., 2015).

As done in the noise-free case in Sangiorgio and Dercole (2020), we thus trained the LSTM architecture also without TF (LSTM-no-TF in Table 1). Coupling these two elements, recurrent neurons and no-TF, solves at the same time the drawbacks of the FF predictors and of LSTM-TF. Indeed, this architecture is trained to predict the entire sequence of output variables taking into account their temporal connection, and makes the training and inference modes coherent, so that the network can limit the propagation of prediction errors through the sequence of predictions.

2.3 Networks' hyper-parameters

To limit the number of networks' hyper-parameters to be tuned we fixed the number of hidden layers (3), and the number of neurons per layer (10). Other hyper-parameters have been optimized through a grid search. We tested a batch size of 256, 512, and 1024; learning rates equal to 0.1, 0.01, and 0.001; decay factor of 0.001, and 0. The number of epochs has been fixed to 5000 to guarantee the convergence of each training. The gradient descent have been performed adopting Adam optimizer by Kingma and Ba (2014), an algorithm that is extensively used in deep learning applications.

3. TIME SERIES FROM ARCHETYPAL SYSTEMS

We test the robustness of the neural predictors to the observation noise on four archetypal chaotic systems: the logistic map, the Hénon map, and two versions of the generalized Hénon map. We limit the analysis to a single output variable for each system.

The one-dimensional logistic map describes growth processes in many fields. It is defined by the following recurrence:

$$y(t+1) = r \cdot y(t) \cdot \big(1 - y(t)\big), \tag{4}$$

where r is the growth rate at low density. This map has a chaotic behavior for most of the r values in the range 3.6-4 (we use $r = 3.7$ in the following computations).

We then consider three versions of the Hénon map. The classical two-dimensional map and a 3D and 10D generalized Hénon maps that can be written as a m-dimensional nonlinear autoregression:

$$y(t+1) = 1 - a \cdot y(t-m+2)^2 + b \cdot y(t-m+1), \tag{5}$$

where m is the system's dimension. The parameters a and b assume the values 1.4 and 0.3 for the 2D case as in Hénon (1976), and $a = 1.9$ and $b = -0.03$ for the generalized maps which exhibit a hyperchaotic behavior (Baier and Klein, 1990; Richter, 2002).

The results relative to the noise-free case are taken from a previous work by Sangiorgio and Dercole (2020), based on the same experimental setting. Here, we retrain the neural predictors on noisy data, adding a white Gaussian noise to the deterministic systems' simulations. Three noise levels are considered, with standard deviation equal to 0.5%, 1%, and 5% of the process standard deviation.

Fig. 1 reports the results obtained with the intermediate noise level (1%). FF-recursive and LSTM-TF predictors exhibit nearly the same performance in the four cases. Their accuracy is high in the initial part of the forecasting horizon, and rapidly degrades to -1, meaning that the actual and predicted trajectories have become uncorrelated (Dercole et al., 2020). FF-multi-output ensures a positive (at worst null) R^2 score, providing acceptable performances only when the multi-step forecasting horizon is short. LSTM-no-TF shows the same performance of FF-recursive and LSTM-TF in the first part of the horizon, its R^2 score is always positive as the FF-multi-output (see the last part of the horizon) and it outperforms the three competitors in the central part of the horizon.

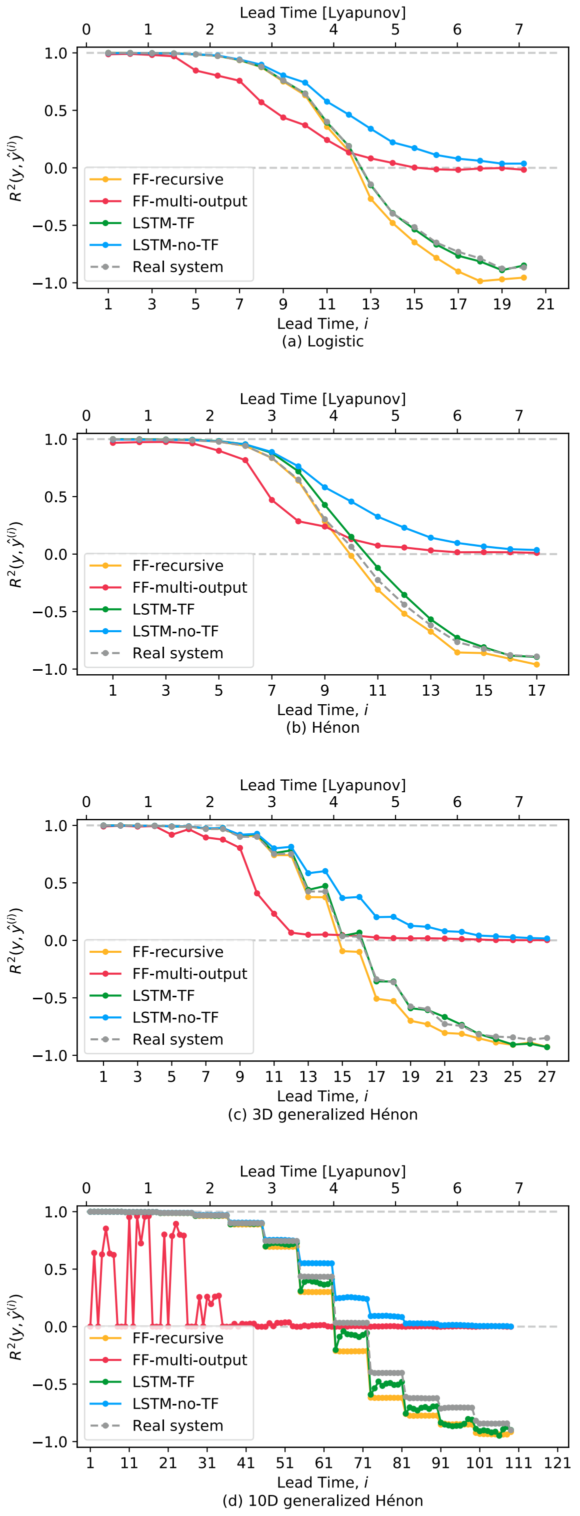

Fig. 1. $R^2(\mathbf{y}, \hat{\mathbf{y}}^{(i)})$ score for the four neural nets and the real system used as predictor on the logistic (a), Hénon (b), 3D (c), and 10D (d) generalized Hénon maps. Test dataset with a 1% noise level.

An interesting benchmark is the real system used as predictor (real system, in Fig. 1 and Table 2). In the noise-free case, the real system always provides a perfect forecast; R^2 score is always equal to one. When considering noisy data, predictions are obtained by simulating the system from a perturbed initial condition set by the d noisy lags, so that the obtained trajectory diverges from the one generating the target data due to the system's chaoticity. In other words, even if one could identify the perfect model of the system, it would not be possible to prevent the multi-step error divergence.

Table 2. Performance of the predictors with different levels of noise

System	Predictor	# LT $R^2 > 0.5$			
		noise free	0.5% noise	1% noise	5% noise
Logistic	FF-recursive	6.37	4.25	3.54	2.12
	FF-multi-output	3.18	2.83	2.83	2.12
	LSTM-TF	> 7.07	4.25	3.54	2.12
	LSTM-no-TF	> 7.07	4.60	3.89	2.12
	Real system	∞	4.24	3.53	2.12
Hénon	FF-recursive	6.43	3.86	3.43	2.14
	FF-multi-output	3.00	3.00	2.57	2.14
	LSTM-TF	> 7.28	3.86	3.43	2.14
	LSTM-no-TF	> 7.28	4.29	3.86	2.57
	Real system	∞	3.86	3.43	2.16
3D gen. Hénon	FF-recursive	6.07	3.86	3.31	2.21
	FF-multi-output	2.76	2.76	2.48	2.21
	LSTM-TF	6.62	4.42	3.31	2.21
	LSTM-no-TF	> 7.45	4.42	3.86	2.21
	Real system	∞	4.42	3.31	2.21
10D gen. Hénon	FF-recursive	6.23	3.97	3.40	2.27
	FF-multi-output	0.00	0.01	0.00	0.01
	LSTM-TF	6.23	3.97	3.40	2.27
	LSTM-no-TF	6.80	4.54	3.97	2.27
	Real system	∞	4.54	3.40	2.27

Looking across the different panels of Fig. 1, it is clear that the decreasing trends are quite similar for all the chaotic maps considered. This fact is interesting since it proves that adopting the system's Lyapunov time as temporal unit (horizontal axis) and the R^2 score as metric (vertical axis) is an appropriate way to standardize the dimension of the problem.

Table 2 reports the sensitivity analysis performed by testing three noise levels in terms of number of LTs for which the R^2 score is higher than 0.5. As expected, the performances are considerably worse than those obtained in the noise-free case since the chaotic behavior of the considered maps exponentially amplify the noise on the initial condition. The predictors' performances decrease faster when the noise level is higher.

The comparison between colored and grey curves allows to analyze the differences between the four neural networks and the real system used as predictor. FF-recursive and LSTM-TF exhibit almost the same trend of the real system used as predictor because they minimize the 1-step-ahead prediction error. They thus solve a system identification task, instead of specifically searching for the best multi-step predictor. Conversely, the LSTM-no-TF is optimized for the considered horizon and, for this reason, provides a better performance then the other competitors on the forecasting task.

The comparison also allows to separately evaluate the two components of the prediction error, the first caused by the uncertainty in the identification process (identification error), the second due to the propagation of the observation noise from the input data to the output (observation error). The fact that the FF-recursive and LSTM-TF architectures

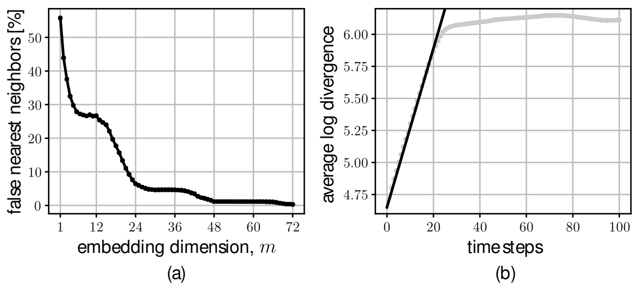

Fig. 2. Output of the false nearest neighbors algorithm (a) and estimation of the LLE = 0.062 (b) for the solar irradiance time series.

have a predictive power similar to the real system used as predictor, shows that identification error is a minor component. This means that the knowledge the actual system does not help improving the predictive accuracy when dealing with a chaotic dataset affected by observation noise.

4. REAL-WORLD TIME SERIES

We now test our neural predictors on a real case study, considering a solar irradiance dataset (2014-2019) measured in Como, Italy, and provided by CML (`www.centrometeolombardo.com`).

We first evaluate a suitable embedding dimension for the dataset, using the false nearest neighbors algorithm Kennel et al. (1992). The fraction of false neighbors becomes negligible starting from $m = 48$ (see Fig. 2a), meaning that such value is appropriate to reconstruct the system dynamics (Takens, 1981). 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 48-dimensional space of delayed coordinates. Essentially, starting from similar windows of 48 consecutive data, the algorithm estimates the average exponential rate of divergence of the data that follow the considered windows. The rate is obtained as the slope of the linear part of the log-divergence plot reported in Fig. 2b. The resulting value is LLE = 0.062, that confirms the caoticity of the series, as already shown by Fortuna et al. (2016).

The comparison of the multi-step prediction of solar irradiance with LSTM and FF networks is performed using data from 2014 to 2017 for parameters training, 2018 for the validation, and 2019 for testing.

Before presenting the results, it is worth noticing that the classical indicators may overestimate the actual performances of models when applied to the complete time series. When dealing with solar irradiance, there is always a strong bias due to the presence of many null values. In the case at hand, they are about 57% of the sample due to different factors: the rotation and revolution motion of earth, the additional shadowing of the nearby mountains and the sensitivity of the sensors. When the recorded value is zero, also the forecast is zero (or very close) and all these small errors substantially reduce the average errors, increasing the R^2 score. Additionally, forecasting the solar irradiance

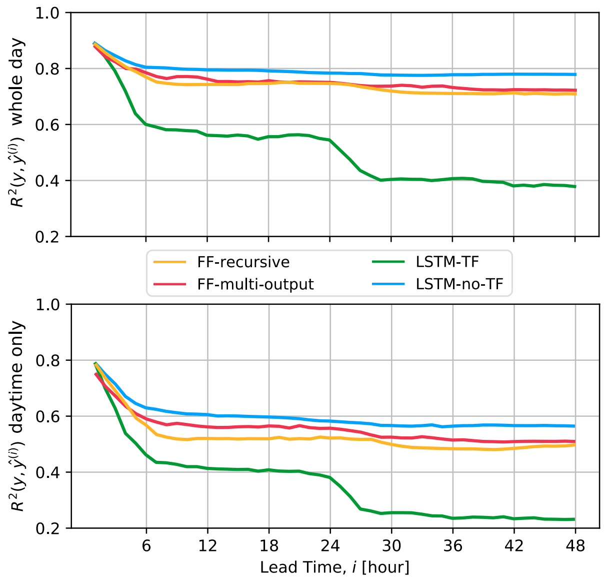

Fig. 3. $R^2(\mathbf{y}, \hat{\mathbf{y}}^{(i)})$ score for the four predictors of the solar irradiance. Performance in the whole day (top), and daytime only (bottom). Test dataset.

during the night is useless, and the power network manager that may use the forecast, for instance, to balance electricity production may well turn the forecasting model off. In order to overcome this common deficiency and allow the models' performances to be compared when they may indeed be useful, we have also computed the performance indicators considering only values above 25 Wm^{-2} (daytime, in what follows).

Fig. 3 reports the results obtained with the four neural predictors in terms of R^2 score both in the whole day and on daytime samples only. Again, the LSTM-no-TF predictor provides the best performances. Comparing the latter with the FF-multi-output predictor, it emerges that the two have a similar accuracy on the 12-step horizon, while FF-multi-output predictor performs slightly worse after 24 steps ahead. Adopting the FF-recursive approach, the R^2 score decreases, specifically after 5 hours. Finally, considering the LSTM-TF predictor the performance drops dramatically. This is probably due to the highly periodic trend characterizing the solar irradiance, which requires to properly propagate the information through the internal state of the recurrent cells. The LSTM architecture fails in such task if it is trained with the traditional teacher forcing procedure. For all the considered neural predictors, the difference between the whole time series (average value 140.37 Wm^{-2}) and the daytime case (average 328.62 Wm^{-2}), emerges clearly, given that during all nights the forecasting errors are negligible.

5. CONCLUSION

In this paper, we analyzed how the forecasting skills of feed-forward and recurrent neural predictors on a multi-step horizon are affected by the presence of observation uncertainty. First, we investigated the effect of artificially-generated additive noise on four deterministic time series of different complexity and chaoticity, including a case of hyperchaos. We then considered a solar irradiance time series to evaluate how the situation changes when going from artificial to real-world systems. The finite embedding and the positive (though small) Lyapunov exponent suggest that the series is generated by a nonlinear chaotic process.

Whatever the system or the level of noise, our results show that LSTM-no-TF is the most performing multi-step predictor. This confirms the same conclusion recently reached in the noise-fee case (Sangiorgio and Dercole, 2020) and makes it robust to observation uncertainty.

Our experiments also proved that the error due to the model identification process is negligible if compared to the one caused by the observation noise, even when this latter is really limited (its dispersion is in the order of 0.5% of the process standard deviation). In other words, knowing the actual model of the system is practically useless, in terms of predictive accuracy, if one can properly identify a neural predictor.

The results also confirm the suitability of FF and LSTM networks in the forecasting of time series related to environmental variables. FF-multi-output provides a forecasting accuracy almost identical to LSTM-no-TF, while the FF-recursive and LSTM-TF predictors have a lower predictive power. Focusing on LSTM nets, it emerges the importance of the training approach used also in real-world dataset. Another interesting conclusion is that between the FF-recursive and FF-multi-output, the former exhibits the lowest performances. This is due to the fact that its parameters are optimized over a time horizon of 1 step, but then used for a longer horizon, thus propagating the error. Therefore, the common practice to identify a single-step predictor, and then to use it in a recursive way to forecast the sequence $\hat{y}(t+1), \hat{y}(t+2), \ldots, \hat{y}(t+h)$, may not be the best choice. A FF-multi-output may represent a more suitable alternative.

Expanding this idea to a broader context, we can conclude that prediction and system identification are related but different tasks. A FF network trained on the one-step prediction can mimic fairly well the behavior of a chaotic system but other configurations exhibit greater predictive power and robustness over multiple steps.

This paper represents the first attempt of systematically evaluating the effect of observation noise in forecasting chaotic dynamics. The other two papers on the same topic (Brajard et al., 2020; Chen et al., 2020) only 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.

ACKNOWLEDGEMENTS

The authors would like to thank Prof. G. Nunnari, University of Catania, Italy, for many fruitful discussions on solar irradiance forecasting.

REFERENCES

Baier, G. and Klein, M. (1990). Maximum hyperchaos in generalized hénon maps. *Phys Lett A*, 151(6-7), 281–284.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In *Adv Neur In*, 1171–1179.

Brajard, J., Carrassi, A., Bocquet, M., and Bertino, L. (2020). Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: A case study with the lorenz 96 model. *J Comput Sci-neth*, 44, 101171.

Chen, P., Liu, R., Aihara, K., and Chen, L. (2020). Autoreservoir computing for multistep ahead prediction based on the spatiotemporal information transformation. *Nat Commun*, 11(1), 1–15.

Dercole, F., Sangiorgio, M., and Schmirander, Y. (2020). An empirical assessment of the universality of ANNs to predict oscillatory time series. *IFAC-PapersOnLine*, 53(2), 1255–1260.

Fortuna, L., Nunnari, G., and Nunnari, S. (2016). *Nonlinear modeling of solar radiation and wind speed time series*. Springer.

Guariso, G., Nunnari, G., and Sangiorgio, M. (2020). Multi-step solar irradiance forecasting and domain adaptation of deep neural networks. *Energies*, 13(15), 3987.

Hénon, M. (1976). A two-dimensional mapping with a strange attractor. In *The Theory of Chaotic Attractors*, 94–102. Springer.

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

Kennel, M.B., Brown, R., and Abarbanel, H.D. (1992). Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Phys Rev A*, 45(6), 3403.

Kingma, D.P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Mihaylova, T. and Martins, A.F. (2019). Scheduled sampling for transformers. *arXiv preprint arXiv:1906.07651*.

Pathak, J., Hunt, B., Girvan, M., Lu, Z., and Ott, E. (2018). Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys Rev Lett*, 120(2), 024102.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2015). Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Richter, H. (2002). The generalized henon maps: Examples for higher-dimensional chaos. *Int J Bifurcat Chaos*, 12(06), 1371–1384.

Sangiorgio, M. (2021). *Deep learning in multi-step forecasting of chaotic dynamics*. Ph.D. thesis, Politecnico di Milano.

Sangiorgio, M. and Dercole, F. (2020). Robustness of LSTM neural networks for multi-step forecasting of chaotic time series. *Chaos Soliton Fract*, 139, 110045.

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

Vlachas, P., Pathak, J., Hunt, B., Sapsis, T., Girvan, M., Ott, E., and Koumoutsakos, P. (2020). Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks*, 126, 191–217.

Williams, R.J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Comput*, 1(2), 270–280.