Neural Networks 167 (2023) 715–729

Contents lists available at ScienceDirect

Neural Networks

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

An adaptive embedding procedure for time series forecasting with deep neural networks

Federico Succetti, Antonello Rosato, Massimo Panella *

Department of Information Engineering, Electronics and Telecommunications (DIET), University of Rome "La Sapienza", Via Eudossiana 18, 00184 Rome, Italy

ARTICLE INFO

Article history:
Received 4 January 2023
Received in revised form 30 May 2023
Accepted 28 August 2023
Available online 9 September 2023

Keywords:
Deep neural network
Adaptive embedding
Long Short-Term Memory
Forecasting
Time series

ABSTRACT

Nowadays, solving time series prediction problems is an open and challenging task. Many solutions are based on the implementation of deep neural architectures, which are able to analyze the structure of the time series and to carry out the prediction. In this work, we present a novel deep learning scheme based on an adaptive embedding mechanism. The latter is exploited to extract a compressed representation of the input time series that is used for the subsequent forecasting. The proposed model is based on a two-layer bidirectional Long Short-Term Memory network, where the first layer performs the adaptive embedding and the second layer acts as a predictor. The performances of the proposed forecasting scheme are compared with several models in two different scenarios, considering both well-known time series and real-life application cases. The experimental results show the accuracy and the flexibility of the proposed approach, which can be used as a prediction tool for any actual application.

1. Introduction

For several years, time series forecasting has covered a wide range of real-life problems. It has become fundamental in many fields, such as, weather forecasting (Succetti, Rosato, Araneo, & Panella, 2020; Zhao et al., 2016), energy consumption (Deb, Zhang, Yang, Lee, & Kwok Wei, 2017; Rosato, Panella, Andreotti, Mohammed, & Araneo, 2021), financial indexes (Chen & Chen, 2015), anomaly detection (Ashok, Govindarasu, & Ajjarapu, 2018; Ceschini et al., 2021), and so on. At the same time, the inherent characteristics of time series data make their analysis a challenging task. This is mainly due to the causality constraints of their time component, which must always be taken into account when working with this type of data. Other important aspects of time series are trends, seasonal variations and correlation among observed values that can be close or far in time. These issues are related to the nature of the time series themselves.

In the past few years, Deep Neural Networks (DNNs) have become one of the most popular approaches for solving time series forecasting problems thanks to their ability to map complex non linear feature interactions (Reyes & Ventura, 2019). Among DNNs, the ones exploiting gating mechanisms, as the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) layers, represent the state of the art for working with time series (Chung, Gulcehre, Cho, & Bengio, 2014; Hochreiter & Schmidhuber, 1997). Despite the well-known capabilities of LSTM networks, they are not always able to solve the time series prediction problem accurately. One of the main reasons that hinders the prediction performance of such models is related to their training process; in fact, the inherently huge number of hyperparameters to be optimized can make the whole process unfeasible because of its heavy computational load. Also, it is well-known that the final performance of such models is extremely sensitive to the random initialization of model parameters. Another reason why LSTM networks sometimes fail to achieve a good level of accuracy can be related to the highly stochastic and/or nonlinear and/or non-stationary nature of any real-world time series itself (Gers, Eck, & Schmidhuber, 2002; Makridakis, Spiliotis, & Assimakopoulos, 2018). In these cases, it is important to follow the data evolution as accurately as possible, taking into account that the 'physical' system generating the time series is usually unknown and its internal dynamics (i.e., its internal state) is not observable; the only information we have in this regard is conveyed by the measured time series under analysis.

In this paper, a novel deep learning approach for time series forecasting is presented: we propose to adopt a DNN model in order to obtain a compressed representation of the time series that can be associated to the inner evolution of the unknown system that generates them, therefore acting as an Adaptive Embedding (AE) procedure for the subsequent estimation of future values

* Corresponding author.
E-mail addresses: federico.succetti@uniroma1.it (F. Succetti), antonello.rosato@uniroma1.it (A. Rosato), massimo.panella@uniroma1.it (M. Panella).

https://doi.org/10.1016/j.neunet.2023.08.051

of the input sequences (Kennel, Brown, & Abarbanel, 1992). The proposed approach is based on a two-layer bidirectional LSTM (biLSTM) network, where the first layer exploits the AE mechanism to extract the salient features associated with the input time series in an unsupervised way, while the second layer acts as a predictor. The main novelty of this technique relies right on the union between the proposed AE of the input time series and the DNN, which act together as a single system. This allows to eliminate the necessity to adopt feature extraction algorithms as a further preprocessing step before performing the training procedure, while obtaining their advantages (Yang, Wan, Zhang, & Xiong, 2022; Zhao, Deng, Li, Wang, & Wei, 2020). Moreover, the proposed approach is based on a two-fold training process, namely 'pre-training' and 'prediction', in which the AE is exploited in order to obtain with a high level of accuracy the salient information extracted from the input time series.

The guess of this work is that a DNN based on an AE scheme can improve the system identification process in order to obtain a more accurate prediction of future time series behaviors. Furthermore, as proved in the following of this paper, the hyperparameters' optimization can be facilitated by using a greedy layer-wise pre-training of the AE stage, followed by the fine-tuning of the predictor stage in order to solve the underlying forecasting problem.

The proposed approach is compared with several models in two different types of tests. The first one is carried out on three well-known time series prediction benchmarks, whereas the second one is carried out on several real-world time series. Three of them are taken from the M4 competition (Makridakis, Spiliotis, & Assimakopoulos, 2020), while the others are related to real-world application cases. The experimental results show that the proposed DNN can be used as an accurate prediction tool sporting a high level of flexibility; it can be applied to any kind of time series while also considering different types of DNN layers, which may result more suited to the specific time series to be predicted.

The paper is organized as follows. The related works about the use of similar AE techniques and both LSTM and biLSTM networks for solving time series forecasting problems are summarized in Section 2. The theoretical AE concepts along with a detailed analysis of the proposed model are introduced in Section 3. The assessment of the proposed AE technique on three benchmark time series is illustrated in Section 4, while the experimental results are shown in Section 5, where the performance of the proposed approach is also compared with those of seven benchmark models for time series prediction. Finally, some concluding remarks and future research directions are reported in Section 6.

2. Related work

The performances of LSTM networks are widely assessed in the literature. In recent years, LSTM networks have established themselves as the state of the art in time series modeling. They are designed to work with data sequences. In fact, they solve the vanishing gradient problem (Bengio, Simard, & Frasconi, 1994) and they are able to retain information for long periods of time. This makes them suitable for managing the long-term dependencies present in time series.

Despite the advantages of LSTM networks, they provide poor performance when there are strong variations in time series (Makridakis et al., 2018). This problem is usually solved through the use of autoencoders, because they are able to extract an encoded representation of the data automatically (Gensler, Henze, Sick, & Raabe, 2016; Li, Yu, Shahabi, & Liu, 2017).

In literature there are several studies concerning LSTM and biLSTM networks and autoencoders, also considering different combinations of them for solving time series forecasting problems. Considering LSTM networks, the authors in Bae, Kim, and Lee (2021) predict the future trend of significant nuclear power plant parameters to detect a human error in a short time or even prevent it. They use three different architectures and multiple parameters to carry out their multivariate analysis. The experimental results prove that the LSTM network is the most accurate.

In Aggarwal and Toshniwal (2021), Aggarwal and Toshniwal use a combination of particle swarm optimization for hyperparameters calibration and LSTM networks to predict the air quality of 15 locations in India. They compare the performance of their model with traditional sequential models, and they also use several existing benchmark dataset samples to further prove the superiority of their approach.

LSTM networks are also used in tasks pertained to the energy field. For example, the authors in Li and Becker (2021) use a combination of LSTM networks and feature selection algorithms to predict the electricity price under the consideration of market coupling. In another study in Rong Liu and Huang (2021), Liu et al. combined a Convolutional Neural Network (CNN) and a LSTM network to perform an accurate short-term power load forecasting obtaining good results.

As stated before, autoencoders play an important role in time series modeling. Therefore, it is necessary to underline that the proposed approach differs from the classic autoencoder mechanism. In fact, it is based on the synergy given by the joint use of the AE procedure and the deep neural architecture. The latter work together as a single system, providing an accurate unsupervised prediction as output.

Regarding autoencoders, the authors in Kim and Cho (2021) use a deep autoencoder to forecast the energy demand and provide an explanation of how it works. Experimental results show that they reach a mean squared error of 0.376 in predicting the electricity demand every one minute for a time interval of one hour. In Takahashi, Ooka, and Ikeda (2021) Takahashi et al. use autoencoders and other machine learning algorithms to address the anomaly detection problem regarding the electrical demand in a hospital located in Japan. Experimental results prove that their methodologies can increase energy savings and reduce peak building loads.

The combination of LSTM and/or biLSTM networks with autoencoders can be found in several fields. For example, in Tong et al. (2022) Tong et al. propose an LSTM-Autoencoder model that integrates long-term and short-term features for load forecasting. The encoder part is used to extract time series features and generate latent vectors, whereas the decoder tries to reconstruct the input sequence while outputting the prediction results. They reach a mean absolute error less then 52 MW on the Alberta Electric System Operator dataset.

In another study in Yang, Zhai, and Li (2021), the authors focus their attention on solving the problem related to the online AC False Data Injection Attack detection in smart grids by using an LSTM-Autoencoder network. Experimental results demonstrate satisfactory attack detection accuracy of the model on IEEE 14 and 118-bus systems.

In Sagheer and Kotb (2019), the authors proposed a methodology where an autoencoder is pre-trained, then the encoder is detached and used as a feature extractor before feeding an LSTM network. Afterward, only the LSTM network is trained. They verify that this approach improves the performance of deep LSTM networks and leads to fast convergence.

A research conducted in Padnekar, Kumar, and Deepak (2020) uses a biLSTM autoencoder for stance prediction, that is, the process of automatically classifying the stance of a news article towards a target into one of the following classes: 'Agree', 'Disagree', 'Discuss', 'Unrelated'. The authors demonstrated that the method is reasonably accurate at predicting stance, achieving a

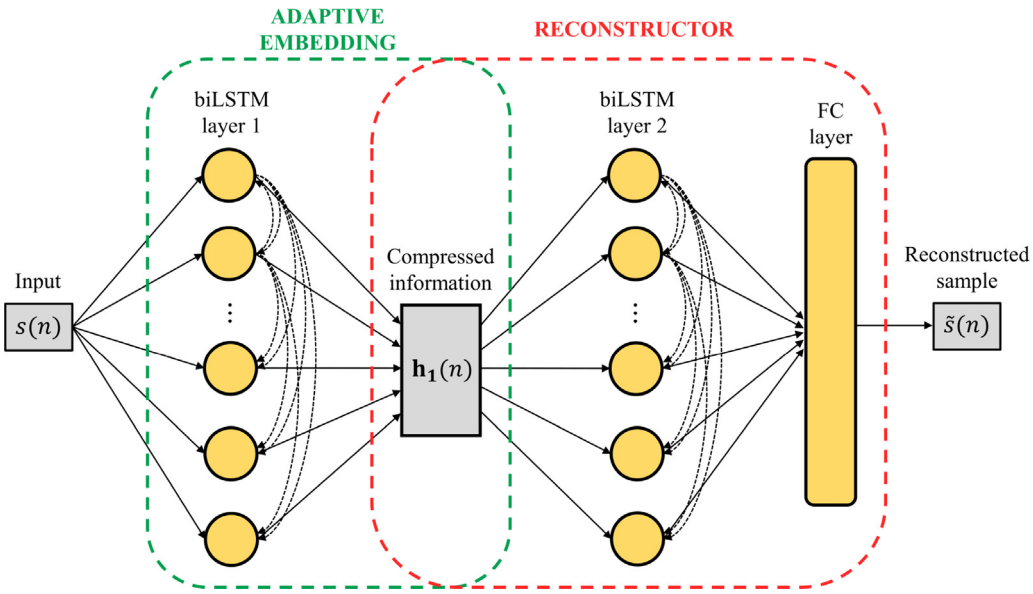

Fig. 1. biLSTM-AE network in the pre-training phase: all layers are trained by using as target values the same samples at the input; solid and dashed lines in black represent feed-forward and recurrent connections, respectively.

classification accuracy as high as 94%. Another study in Lee et al. (2021) deals with the anomaly detection problem. A biLSTM-based autoencoder is used to find the anomalous data point considering metering data corresponding to 4 types of energy sources electricity/water/heating/hot water collected from 985 households reaching a level of accuracy of 99.5%. Finally, the authors in Khan et al. (2021) proposed a one-step forecast of renewable energy generation for short-term horizons by incorporating an autoencoder with a biLSTM network. They reached state of the art results in terms of error metrics, with RMSE of 0.103 and 0.019 considering the prediction of solar power and wind speed, respectively.

Following the previous discussion, it seems to the best of our knowledge that no works in the literature actually investigated an AE mechanism coupled with a biLSTM network for univariate time series forecasting, although they can use similar tools and stacked networks (i.e., LSTM and/or biLSTM layers and autoencoders). Nonetheless, there are several works that improve the potential of standard LSTM networks by using several methodologies, such as, the use of CNN layers to capture the dependencies among different time series, the use of optimization algorithms to fit the hyperparameters and the use of autoencoders. However, autoencoders not based on recurrent neural networks are unable to recognize the long-term dependencies present in time series data, as well as DNNs that leverage autoencoders and LSTM/biLSTM networks are not always trained with a two-phase approach, as proposed in this paper.

3. Prediction system based on adaptive embedding

A typical situation arising when a DNN is trained regards a strongly non-convex objective function, with many distinct local minima in the model's parameter space. In these cases, the main difficulty lies in the fact that not all of these local minima provide an equivalent generalization error. Furthermore, standard DNN training schemes are mainly based on the random initialization of parameters, with the possibility to place them in regions of the parameter space that yield a poor generalization capability (Bottou, Chapelle, DeCoste, & Weston, 2007).

The approach that has enabled DNNs to be effectively trained is the one based on the greedy layer-wise unsupervised pre-training followed by supervised fine-tuning. In this approach, each layer is pre-trained in an unsupervised way, learning a non-linear transformation of its input (the output of the previous layer). This unsupervised pre-training represents the starting point of the final training phase, where the deep architecture is fine-tuned in a supervised manner. A possible reason to explain why the greedy layer-wise unsupervised pre-training will result so effective is that it initializes the model's parameters in a point of the parameter space where the parameters are 'restricted' (Erhan, Courville, Bengio, & Vincent, 2010).

The greedy principle can be exploited by the AE mechanism in order to learn a compressed representation of a set of data by training the network to ignore unimportant data (i.e., noise). The AE must be able to capture the statistical structure in the training set to minimize the reconstruction error, that is, the difference between the input and the output as the reconstructed input.

In order to take advantage of the greedy principle and the AE concepts in the context of time series forecasting, we apply the AE scheme to a recurrent DNN made up of two stacked biLSTM layers. The proposed network will be referred to in the following as 'biLSTM-AE'. The novelty of the proposed approach consists in exploiting the aforementioned concepts to realize an adaptive data embedding procedure based on a two-step training approach; the steps are referred to as 'pre-training' and 'prediction', respectively, and they are illustrated in the following.

3.1. Pre-training phase

Let $s(n)$, $n > 0$, be the scalar time series to be predicted, where the sample at time step n and the previous ones are the known samples that can be used for prediction. It is important to point out that we decided to use here a scalar input $s(n)$ as input to the model, instead of a vector of past samples, in order to stress the AE capability of the proposed approach. Nevertheless, in Section 5 we use a window of past samples to perform the multi-step-ahead predictions on real-life time series. In the pre-training phase, the proposed DNN carries out the AE procedure. The first biLSTM layer receives as input $s(n)$ and learns at its output the compressed information $\mathbf{h}_1(n) \in \mathbb{R}^{H_1}$, which contains the hidden states (at time n) associated with the H_1 units of the layer. The second biLSTM layer is fed at the input by $\mathbf{h}_1(n)$ and produces as output the vector $\mathbf{h}_2(n) \in \mathbb{R}^{H_2}$, which consists of

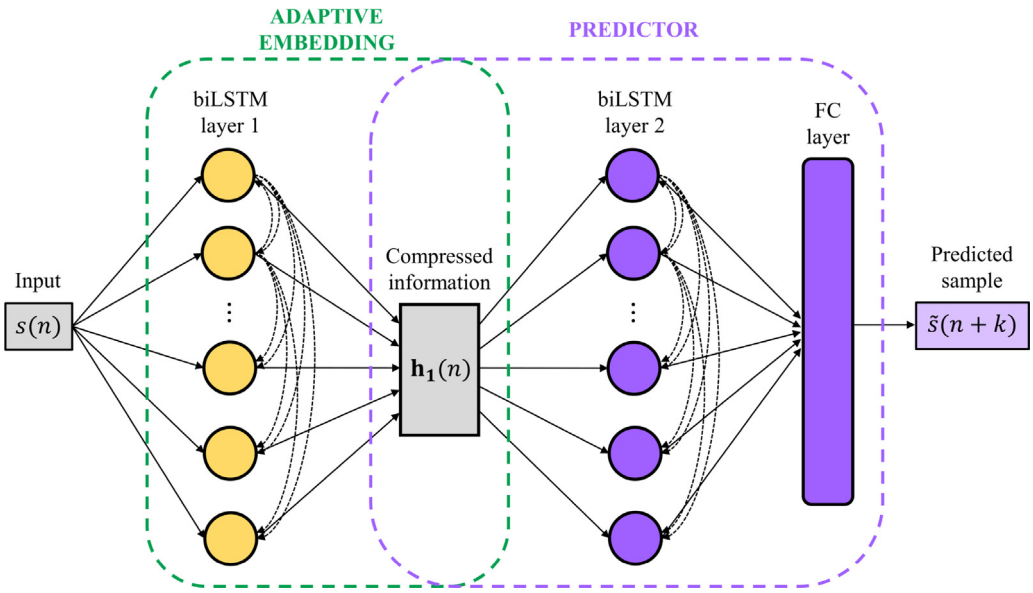

Fig. 2. biLSTM-AE network in the prediction phase: the first biLSTM layer is not trained, its weights are the ones obtained during pre-training; the second biLSTM and the FC layers are trained by using as target values future samples at a prediction distance k with respect to the ones present at the input.

the hidden states associated with the H_2 units of this second layer. Finally, a Fully Connected (FC) layer is used to compile linearly the reconstructed scalar sequence $\tilde{s}(n)$ receiving $\mathbf{h}_2(n)$ at its input. This way, the proposed biLSTM-AE network performs a low-dimensional identity mapping (i.e., embedding) over the input time series. The pre-training configuration of the biLSTM-AE network is reported in Fig. 1.

From a modeling point of view, the embedding function $e : \mathbb{R} \to \mathbb{R}^{H_1}$ such that $\mathbf{h}_1(n) = e\,(s(n))$ is obtained by the following pair of recurrent state equations associated with the first biLSTM layer:

$$\mathbf{c}_1(n) = f\left(s(n), \mathbf{h}_1(n-1), \mathbf{h}_1(n+1), \mathbf{c}_1(n-1);\ \boldsymbol{\theta}_{c_1}\right), \tag{1a}$$

$$\mathbf{h}_1(n) = g\left(s(n), \mathbf{h}_1(n-1), \mathbf{h}_1(n+1), \mathbf{c}_1(n-1);\ \boldsymbol{\theta}_{h_1}\right), \tag{1b}$$

where subscript '1' refers to the first biLSTM layer; $\mathbf{c}_1(n)$ and $\mathbf{h}_1(n)$ are the 'cell' and 'hidden' state vectors of the biLSTM layer, respectively; $f(\cdot)$ and $g(\cdot)$ are general functions obtained by the combination of the gate equations of the biLSTM model, depending on the chosen activation functions, etc.; $\boldsymbol{\theta}_{c_1}$ and $\boldsymbol{\theta}_{h_1}$ are the layer's weights that are set by the training algorithm.

The reconstruction function $r : \mathbb{R}^{H_1} \to \mathbb{R}$ such that $s(n) = r\,(\mathbf{h}_1(n))$ is obtained by the following pair of recurrent state equations associated with the second biLSTM layer and the feed-forward equation of the FC layer:

$$\mathbf{c}_2(n) = f\left(\mathbf{h}_1(n-1), \mathbf{h}_1(n+1), \mathbf{h}_2(n-1), \mathbf{h}_2(n+1), \mathbf{c}_2(n-1);\ \boldsymbol{\theta}_{c_2}\right), \tag{2a}$$

$$\mathbf{h}_2(n) = g\left(\mathbf{h}_1(n-1), \mathbf{h}_1(n+1), \mathbf{h}_2(n-1), \mathbf{h}_2(n+1), \mathbf{c}_2(n-1);\ \boldsymbol{\theta}_{h_2}\right), \tag{2b}$$

$$\tilde{s}(n) = \mathbf{w}^t \mathbf{h}_2(n) + b\,, \tag{2c}$$

where, in this case, subscript '2' refers to the second biLSTM layer; $\mathbf{c}_2(n)$ and $\mathbf{h}_2(n)$ are the state vectors of this layer; $\boldsymbol{\theta}_{c_2}$ and $\boldsymbol{\theta}_{h_2}$ are the layer's weights that are set by the training algorithm as well, together with the weight vector $\mathbf{w}$ and the bias b of the FC layer. We denoted in (2c) a desired estimate $\tilde{s}(n)$ of the whole network input $s(n)$.

3.2. Prediction phase

In the prediction phase, the biLSTM-AE network is re-trained to perform the actual time series forecasting. In this stage, the weights of the first biLSTM layer are locked at the values obtained during the pre-training phase. Hence, the vector $\mathbf{h}_1(n)$ will be identical to the one obtained before so as to maintain the embedding of the input sequence that will be exploited for the actual prediction. The latter is carried out by the second biLSTM layer and the FC layer as well, which are re-trained to estimate a future value of the time series at a distance $k > 0$, namely $\tilde{s}(n+k)$. The prediction configuration of the biLSTM-AE network is reported in Fig. 2.

At this stage, the embedding's model is identical to the one introduced in (1), so the related equations holds with the same parameter vectors $\boldsymbol{\theta}_{c_1}$ and $\boldsymbol{\theta}_{h_1}$. Conversely, the second part of the stack is trained and so the model's equations in (2) are replaced by the following ones:

$$\mathbf{c}'_2(n) = f\left(\mathbf{h}_1(n-1), \mathbf{h}_1(n+1), \mathbf{h}'_2(n-1), \mathbf{h}'_2(n+1), \mathbf{c}'_2(n-1);\ \boldsymbol{\theta}'_{c_2}\right), \tag{3a}$$

$$\mathbf{h}'_2(n) = g\left(\mathbf{h}_1(n-1), \mathbf{h}_1(n+1), \mathbf{h}'_2(n-1), \mathbf{h}'_2(n+1), \mathbf{c}'_2(n-1);\ \boldsymbol{\theta}'_{h_2}\right), \tag{3b}$$

$$\tilde{s}(n+k) = \mathbf{w}'^t \mathbf{h}'_2(n) + b'\,, \tag{3c}$$

where the superscript denotes different values with respect to the pre-training phase, as the learning algorithm will estimate new parameters $\boldsymbol{\theta}'_{c_2}$, $\boldsymbol{\theta}'_{h_2}$, $\mathbf{w}'$, and b' based on the different target values $\tilde{s}(n+k)$.

It is important to outline that, while they are both needed to reach a good prediction, the two phases can be used with different timing; namely, pre-training can be used only when it is needed to get a grasp of the underlying evolution of the time series by means of adaptive embedding (for example to follow seasonal drifts), while prediction layers could be re-trained at every time slot in order to follow local variations of the time series without changing the embedding. Consequently, as the prediction phase is adopted more frequently and the number of hyperparameters to be optimized in this case is reduced, the whole optimization process is simplified and what we previously claimed in the Introduction is confirmed.

As a final consideration, we decided to use a DNN made up of only two biLSTM layers and one FC layer because we want to highlight the potential of the proposed AE procedure regardless of the underlying neural architecture. From this point of view,

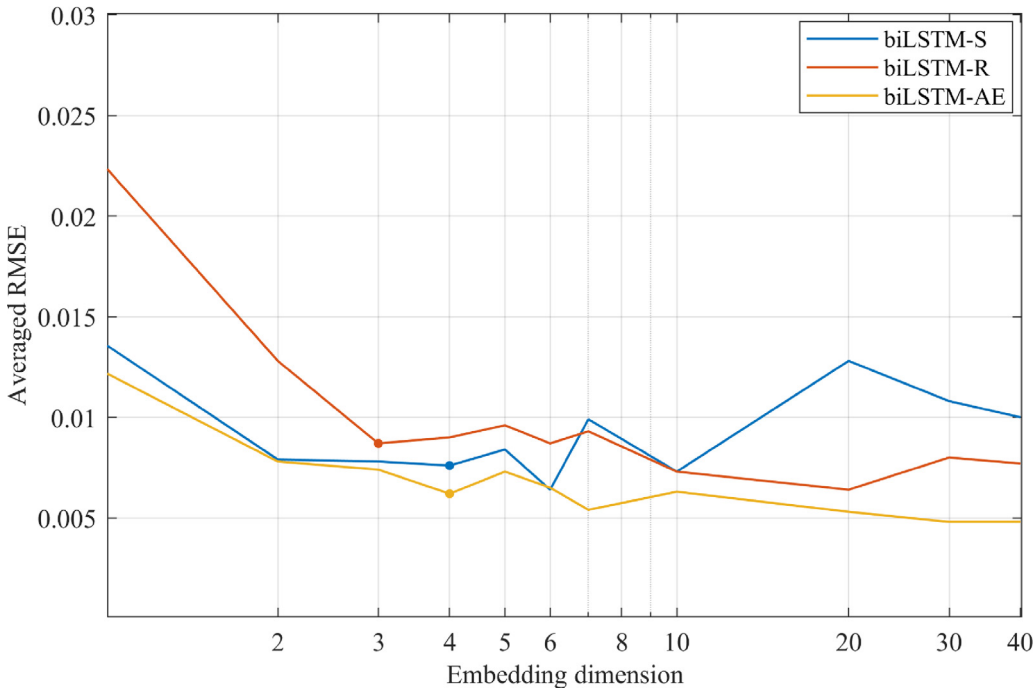

Fig. 3. Average RMSE vs. embedding dimension for the Mackey-Glass time series. The bold points on each curve represent an estimate of the optimal embedding dimension.

the choice of the neural architecture is made with the goal of maximizing the model's ability to learn from the data and make accurate predictions thanks to the effectiveness of the proposed AE procedure. At the same time, this does not limit the possibility of using in the future the AE procedure with more complex neural network architectures in order to face more challenging scenarios.

4. Validation of the network stack for time series embedding

In this Section, the biLSTM-based architecture we propose for the AE of the input time series is assessed on some benchmark time series. Our guess is that the network behavior will be optimal when the hidden dimension H_1 is close to the theoretical embedding dimension of $s(n)$ and that no improvements or rather overfitting is obtained if such a dimension is increased over the right limit. To this end, the following performances will be evaluated against different values of H_1.

Of course, this property may be a characteristic of any other architecture based on two biLSTM layers. In order to prove the absolute quality of the proposed approach, which is based on AE in the pre-training phase and then on forecasting during the second phase after the embedding is achieved, we compare the final performance of the biLSTM-AE network with respect to two other training approaches based on a similar DNN stack. Namely, a standard two-layer biLSTM (biLSTM-S) network where these two layers and the FC one are trained directly in the prediction phase by using as target values the ones to be predicted (i.e., $\tilde{s}(n+k)$), and a randomized (biLSTM-R) version of the latter one, where only the second biLSTM and the FC layers are trained while the first biLSTM layer is randomized.

Standardization is applied to normalize each time series before training, subtracting the mean and scaling by the standard deviation of the training set. A grid search procedure is carried out on the training data in order to set the optimal value of H_2 for the second biLSTM layer and the initial learning rate of the training algorithm. This procedure is applied considering a fixed value of H_1 as it will be varied successively; at this stage, its value is set to the theoretical embedding dimension given by the False Nearest Neighbors (FNN) method (Kennel et al., 1992). It is important to remark that the FNN algorithm is applied considering a time lag $T = 1$ between two consecutive samples of $s(n)$, that is $s(n)$ will not undergo any data reduction or decimation, so as to compare in a similar situation the embedding obtained by the FNN algorithm and the one associated with the proposed biLSTM-AE.

All networks are trained using the ADAM algorithm (Kingma & Ba, 2014) with a gradient decay factor of 0.9, a mini batch size equal to 1, and 300 epochs. In all cases, the prediction distance is set to $k = 1$. Furthermore, each network is trained on each dataset considering 10 different runs, each related to a different (random) initialization of the network's parameters. The prediction performance is reported in terms of average Root Mean Squared Error (RMSE) over these runs. The RMSE is given by:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2}, \tag{4}$$

where y and $\hat{y}$ represent the real value and the actual prediction, respectively, and N is the number of samples in the test sequence. In this case, we decide to use only the RMSE as a quality metric since it gives an appropriate indication of the goodness of the proposed approach. However, in the experimental results reported in Section 5 we evaluate the performance of the models by considering additional metrics to provide a more general assessment.

We report in the following the results obtained by applying this experimental setup to well-known chaotic toy problems related to the Mackey-Glass, Lorenz and Rossler time series.

All the experiments reported in the following were performed using Matlab® R2021b on a machine provided with an AMD Ryzen™ 7 5800X 8-core CPU at 3.80 GHz and with 64 GB of RAM, using for training and inference an NVIDIA® GeForce™ RTX 3080 Ti GPU at 1.365 GHz and 12288 MB of GDDR6X RAM.

Table 1
Average RMSE and embedding dimension for the Mackey-Glass time series.

H_1	biLSTM-S	biLSTM-R	biLSTM-AE
1	1.36	2.24	1.22
2	0.79	1.28	0.78
3	0.78	0.87	0.74
4	0.76	0.90	0.62
5	0.84	0.96	0.73
10	0.73	0.73	0.63
20	1.28	0.64	0.53
30	1.08	0.80	0.48
40	1.00	0.77	0.48
50	0.97	0.63	0.60

Table 2
Average RMSE and embedding dimension for the Lorenz attractor.

H_1	biLSTM-S	biLSTM-R	biLSTM-AE
1	32.98	86.75	31.07
2	26.00	30.68	26.58
3	22.79	26.92	21.48
4	20.28	30.70	26.44
5	31.18	23.84	20.47
10	28.72	23.26	19.86
20	30.85	28.72	21.32
30	35.51	23.10	21.75
40	30.78	22.05	22.05
50	31.42	19.26	21.90

4.1. Mackey-glass time series

The Mackey-Glass time series data refers to the following delayed differential equation (Mackey & Glass, 1977):

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1+x(t-\tau)^{10}} - bx(t), \tag{5}$$

where τ is the time delay. The parameters in (5) are set as follows: $a = 0.2$, $b = 0.1$ and $\tau = 17$. It is important to underline that for $\tau \geq 17$ the time series shows chaotic behavior. Data are generated with an integral step of 0.1 and by setting the initial condition $x(0) = 1.2$.

A number of 2000 samples is considered, where 1800 are used for training and 200 for testing. The grid search procedure is applied to each network using the training set only (1600 samples for training and 200 for validating) and a fixed value $H_1 = 2$, which is the embedding dimension estimated by the FNN method; the optimal values obtained at the end are $H_2 = 50$ hidden units for the second biLSTM layer and 0.009 for the initial learning rate. Successively, with these optimal values fixed, all networks are trained considering different values of H_1 to evaluate different embedding conditions[1]; then, the final networks are tested and the average RMSE is reported.

The results obtained with the three DNNs are shown in Fig. 3. We note that the first local minimum on each curve (represented by a bold point) can be considered as the actual estimate of the embedding dimension and therefore as the optimal hidden dimension produced by the biLSTM-AE. In fact, after the first minimum each RMSE curve saturates with unless subsequent oscillations within a negligible range of values. In this experiment, the first local minimum occurs at $H_1 = 4$ for biLSTM-S and biLSTM-AE, while it occurs at $H_1 = 3$ for biLSTM-R. These values are very close to the theoretical embedding dimensions obtained by the FNN method.

The numerical results, multiplied by the factor 10^{-2}, are reported in Table 1. Once the first local minimum is reached, there are no significant improvements of the RMSE as the embedding dimension increases. Furthermore, we remark that the performance obtained by the biLSTM-AE network is better than those obtained by means of the other two DNN models.

4.2. Lorenz attractor

The Lorenz attractor is described by a system of three ordinary differential equations known as the Lorenz equations (Lorenz, 1963):

$$\frac{dx}{dt} = \sigma(y-x), \tag{6a}$$

$$\frac{dy}{dt} = x(\rho - z) - y, \tag{6b}$$

$$\frac{dz}{dt} = xy - \beta z, \tag{6c}$$

where σ is the Prandtl number and ρ is the Rayleigh number. To show the chaotic behavior, the parameters are set as follows: $\sigma = 10$, $\rho = 28$, $\beta = 8/3$, initial condition $[x(0), y(0), z(0)] = [0, 1, 1.05]$ with an integral step of 0.01.

Only the chaotic time series $x(t)$ is considered; out of a total of 15000 samples, 13000 are used for training and 2000 for testing. The grid search is run on the training set (11000 samples for training and 2000 for validating) with $H_1 = 2$ given by the FNN method, which is right close to the actual Lyapunov dimension 2.06 usually adopted for the Lorenz attractor, yielding as final outcome $H_2 = 30$ for the second biLSTM layer and 0.01 for the initial learning rate.

The results obtained with the three considered DNNs as H_1 varies are illustrated in Fig. 4. In this case, the first local minimum occurs at $H_1 = 3$ for biLSTM-R and biLSTM-AE, and $H_1 = 4$ for biLSTM-S. Also in this case, the values are very close to the theoretical embedding dimension obtained with the FNN method.

The numerical results, multiplied by the factor 10^{-2}, are reported in Table 2 and it is confirmed there are no longer significant improvements as H_1 increases, after the first local minimum of the RMSE is reached. Also in this case, the performance obtained by the biLSTM-AE network is better on average than those obtained by the biLSTM-S and biLSTM-R networks.

4.3. Rossler attractor

The Rossler attractor is described by a system of three non linear ordinary differential equations (Rössler, 1976):

$$\frac{dx}{dt} = -(y+z), \tag{7a}$$

$$\frac{dy}{dt} = x + ay, \tag{7b}$$

$$\frac{dz}{dt} = b + z(x-c). \tag{7c}$$

To obtain the chaotic behavior, the time series is generated with $a = 0.5$, $b = 0.2$, $c = 10$, initial condition $[x(0), y(0), z(0)] = [0.5, 1.5, 0.1]$ and an integral step of 0.01.

As for the Lorenz attractor, only the chaotic time series $x(t)$ is considered. In this case, 20000 samples are used for training and 5000 for testing, with a grid search on the training set only (15000 samples for training and 5000 for validating) using $H_1 = 3$ given

[1] Although a fine-grained search was carried out for all values of H_1 from 1 to 50, for the sake of conciseness, we report in the following figures and tables the results obtained only for some meaningful values of H_1, mainly aiming at the identification of the first minimum and of the asymptotic behavior of the RMSE.

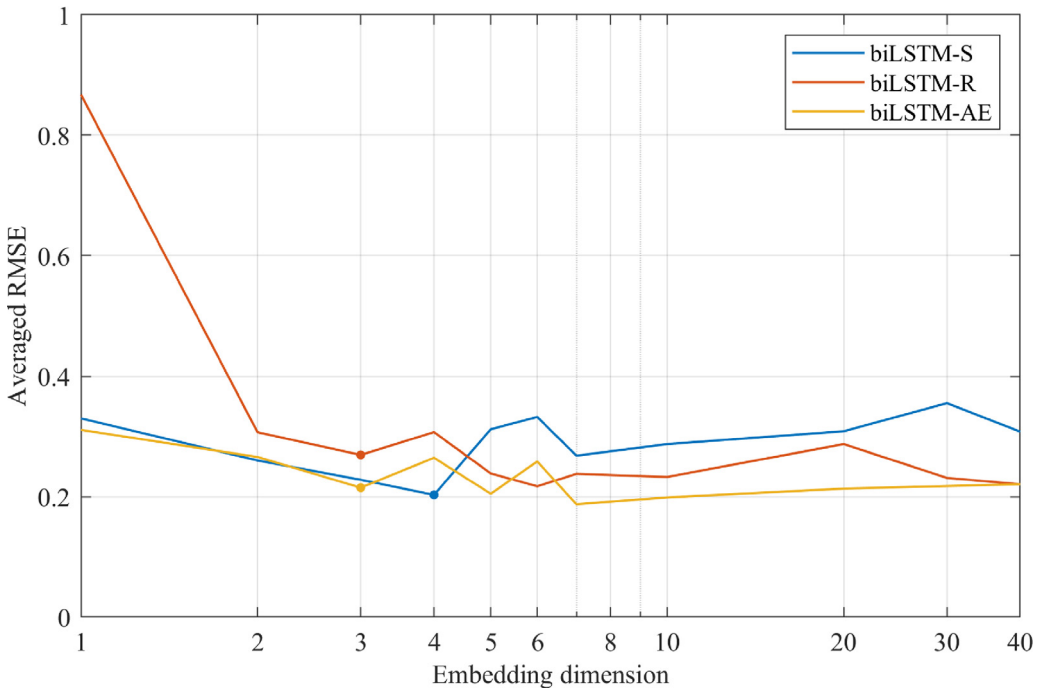

Fig. 4. Average RMSE vs. embedding dimension for the Lorenz time series $x(t)$. The bold points on each curve represent the optimal embedding dimensions.

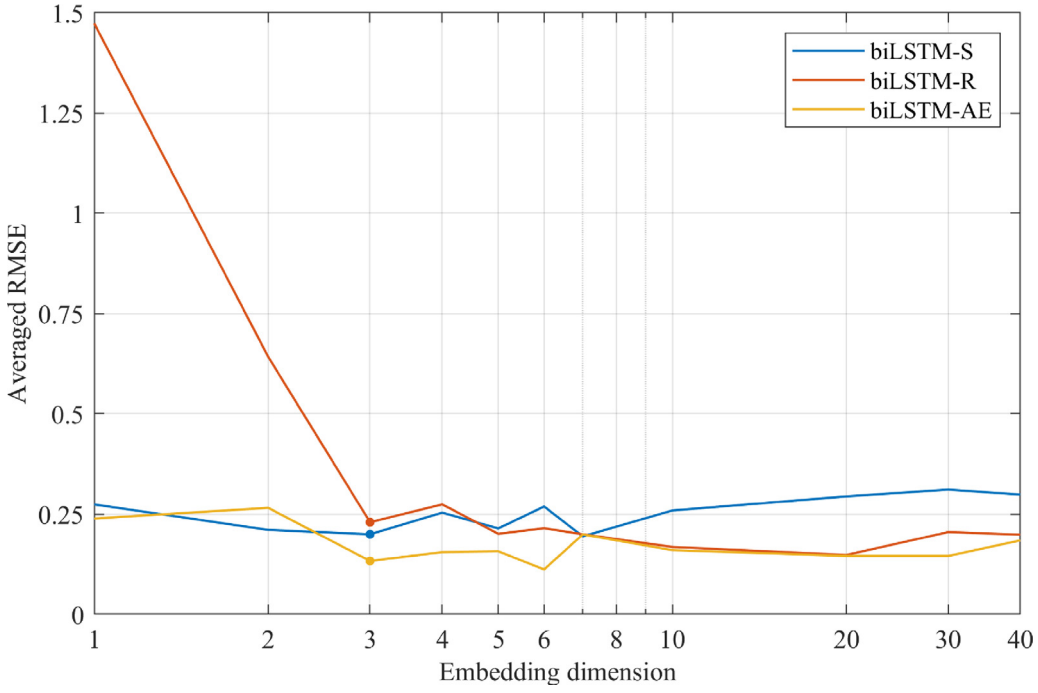

Fig. 5. Average RMSE vs. embedding dimension for the Rossler time series $x(t)$. The bold points on each curve represent the optimal embedding dimensions.

by the FNN method. The optimal values obtained in this case are $H_2 = 20$ and initial learning rate 0.01.

By the graphical results reported in Fig. 5, we observe that the first local minimum occurs with $H_1 = 3$ for all of the DNN models and it also coincides with the theoretical embedding dimension obtained by the FNN method.

The numerical results, multiplied by the factor 10^{-2}, are reported in Table 3 and they are on the same line of the previous ones, in particular confirming the convergent trend of all RMSE curves and the better performance of the proposed biLSTM-AE network with respect to the other ones considered herein.

5. Experiments

The performances of the proposed biLSTM-AE approach are evaluated experimentally on five real-world univariate time series. Three of them are taken from the M4 competition (Makridakis et al., 2020). The first one (S_1) is hourly sampled and goes from July 2015 to August 2015; the second one (S_2) is daily sampled and consists of values from July 2011 to June 2015 while the last one (S_3), which is quarterly sampled, goes from January 1979 to July 2016. The remaining two time series are both hourly sampled. The first one (S_4) represents a case study related to

Table 3
Average RMSE and embedding dimension for the Rossler attractor.

H_1	biLSTM-S	biLSTM-R	biLSTM-AE
1	27.39	147.36	23.88
2	21.05	64.19	26.53
3	19.96	22.96	13.30
4	25.34	27.40	15.50
5	21.41	20.07	15.71
10	25.88	16.80	16.00
20	29.39	14.80	14.56
30	31.09	20.50	14.55
40	29.84	19.86	18.44
50	25.65	14.57	8.92

Table 4
Details of the dataset used.

Input data	# of samples in the training set	# of samples in the test set	Sampling interval
S_1	700	48	hourly
S_2	1429	14	daily
S_3	143	8	quarterly
S_4	120	24	hourly
S_5	456	24	hourly

the prediction of electrical power usage of a 2-storey residential house located in Houston, TX, USA (Polu, 2022); it contains hourly power usage (in kW) starting from June 2016 to August 2020. The last one (S_5) contains a full year of observations related to the output power (in kW) of a PV plant located in San Francisco, CO, USA, in 2015.

All the time series have been chosen on purpose with different characteristics (lengths, shapes, sampling) in order to make a fair comparison. Before applying the learning procedure, a standardization is carried out on the data, subtracting the mean and scaling by the standard deviation of the training set. The latter is different for each time series, especially considered S_1, S_2 and S_3 where both the training and test sets are defined a priori in the competition. On the other hand, S_4 and S_5 are not subject to any constraint, so we decide the number of samples in the training and test sets. Further details are reported in Table 4 together with the sampling step, for each time series.

For the performance evaluation of the proposed approach, seven well-known models are considered for comparison: biLSTM (standard), GRU, DeepAR (Flunkert, Salinas, & Gasthaus, 2017), NBEATS (Oreshkin, Carpov, Chapados, & Bengio, 2019), Temporal Fusion Transformer (TFT) (Lim, Arik, Loeff, & Pfister, 2019), Support Vector Machine (SVM), Zhu, Ye, Wang, Chevallier, and Wei (2022) and Random Forest (RF) (Fan, Zhang, Yu, Hong, & Dong, 2022). The biLSTM network has the same architecture of the proposed biLSTM-AE but it is trained in a standard fashion, without the two-phase approach. The same goes for the GRU, which consists of two-stacked layers. The DeepAR model is made up of two LSTM layers with H_1 and H_2 hidden units, while NBEATS consists of different blocks organized into several stacks, where each block contains 4 FC layers with ReLU non-linearities. Finally, the TFT architecture consists in a combination of LSTM-based encoder–decoder structure, Gated Residual Networks (GRNs) and multi-head attention block. The five DNNs, together with the proposed biLSTM-AE, are trained using the ADAM algorithm with a gradient decay factor of 0.9. On the other hand, the SVM model makes use of the sequential minimal optimization algorithm (Fan, Chen, Lin, & Joachims, 2005) and a linear kernel function.

All models are optimized by using a grid search procedure applied on the training set. In this case, the optimization also takes into account H_1 (considering the proposed biLSTM-AE) in addition to the other hyperparameters. The optimal setups of RF and SVM models are reported in Table 5 whereas the ones related to the DNNs are reported in Table 6. Each predictor is trained considering 10 different runs, each related to a different random initialization of the model's parameters, and the prediction performance is represented by the average error on the test set, including the standard deviation as well. In order to carry out a detailed analysis on models' performance, four different error metrics are considered: Mean Absolute Error (MAE), RMSE, R^2 and Signal to Noise Ratio (SNR).

Table 5
Training hyperparameters for SVM and RF.

Model	Input data	Box constraint	Kernel scale	# of trees	Minimum leaf size
SVM	S_1	3	11	–	-
	S_2	2	11	–	-
	S_3	7	50	–	-
	S_4	2	1	–	-
	S_5	3	25	–	-
RF	S_1	–	-	400	1
	S_2	–	-	200	10
	S_3	–	-	500	20
	S_4	–	-	500	3
	S_5	–	-	300	20

The MAE is given by:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right), \tag{8}$$

where y represents the real value, $\hat{y}$ is the actual prediction and N is the number of samples in the test sequence. It shows how much inaccuracy should be expected from the forecast on average (the lower its value, the better the model). However, because MAE does not reveal the proportional scale of the error, it can be difficult to distinguish between large and little errors. This problem is dammed by the RMSE since it penalizes greater errors more. Thus, it can be compared to the MAE to see whether there are any substantial but uncommon inaccuracies in the forecast. Furthermore, both metrics present the same unit measure of the original series and thus they are easy to comprehend.

Another important metric is the R^2, which is defined by:

$$\mathrm{R}^2 = 1 - \frac{\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{N}\left(y_i - \bar{y}\right)^2}, \tag{9}$$

where $\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2$ denotes the sum of squared residuals from expected values, whereas $\sum_{i=1}^{N}\left(y_i - \bar{y}\right)^2$ represents the sum of squared deviations from the dependent variable's sample mean $\bar{y}$. This metric shows whether the model is a good fit for the observed values: $R^2 = 1$ shows a perfect match of predictions with no errors; $R^2 = 0$ represents the baseline where predictions are always equal to the mean $\bar{y}$; negative values of R^2 exhibit even worse situations.

The last metric, i.e. the SNR, is defined by the following equation:

$$\mathrm{SNR} = 10\log_{10}\frac{\sum_{i=1}^{N} y_i^2}{\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2}, \tag{10}$$

which shows how well the original time series is represented with respect to the prediction error and it is measured in dB (the higher, the better).

In all of the experiments a prediction distance $k = 1$ is used together with a multi-step ahead forecasting approach where all samples in each test set are predicted at the same time reflecting a real-life context, contrary to Section 4 where the samples are predicted with a fine-grain approach (i.e., one-step ahead) to

Table 6
Training hyperparameters for all DNNs.

Model	Input data	H_1	H_2	State size	# of heads	# of stacks	# of blocks	Epochs	Learning rate	Batch size
biLSTM-AE	S_1	60	40	–	–	–	–	25	0.03	1
	S_2	80	80	–	–	–	–	200	0.01	1
	S_3	20	20	–	–	–	–	50	0.01	1
	S_4	20	80	–	–	–	–	50	0.01	1
	S_5	70	20	–	–	–	–	75	0.05	1
biLSTM	S_1	80	20	–	–	–	–	50	0.01	1
	S_2	90	70	–	–	–	–	400	0.01	1
	S_3	20	20	–	–	–	–	50	0.01	1
	S_4	20	40	–	–	–	–	100	0.01	1
	S_5	20	20	–	–	–	–	200	0.01	1
GRU	S_1	20	40	–	–	–	–	50	0.005	1
	S_2	80	40	–	–	–	–	100	0.01	1
	S_3	40	80	–	–	–	–	200	0.001	1
	S_4	50	70	–	–	–	–	200	0.001	1
	S_5	50	20	–	–	–	–	300	0.01	1
DeepAR	S_1	40	40	–	–	–	–	10	0.01	32
	S_2	80	80	–	–	–	–	5	0.01	32
	S_3	60	60	–	–	–	–	10	0.001	32
	S_4	60	60	–	–	–	–	5	0.001	32
	S_5	40	40	–	–	–	–	5	0.001	32
TFT	S_1	-	–	20	4	–	–	10	0.01	32
	S_2	-	–	20	10	–	–	10	0.001	32
	S_3	-	–	20	2	–	–	5	0.001	32
	S_4	-	–	20	10	–	–	5	0.001	32
	S_5	-	–	40	1	–	–	5	0.005	32
NBEATS	S_1	-	–	–	–	30	1	10	0.001	32
	S_2	-	–	–	–	40	1	20	0.0001	128
	S_3	-	–	–	–	30	1	10	0.001	32
	S_4	-	–	–	–	30	1	10	0.0001	1024
	S_5	-	–	–	–	30	1	10	0.001	512

Table 7
Average RMSE and standard deviation of adopted predictors.

Model	S_1	S_2	S_3	S_4	S_5
biLSTM-AE	**24.488 ± 2.746**	104.390 ± 11.462	369.021 ± 23.047	**0.324 ± 0.029**	**27.463 ± 7.078**
biLSTM	31.972 ± 3.419	109.056 ± 16.765	359.327 ± 27.800	0.471 ± 0.111	63.008 ± 16.582
GRU	40.352 ± 4.771	141.134 ± 38.952	203.391 ± 23.364	0.620 ± 0.079	82.921 ± 13.731
DeepAR	55.290 ± 25.947	250.026 ± 38.205	182.565 ± 58.049	0.427 ± 0.066	196.800 ± 37.273
TFT	38.225 ± 15.319	**95.974 ± 16.941**	**61.748 ± 11.594**	0.516 ± 0.111	121.832 ± 76.517
NBEATS	29.947 ± 2.660	239.304 ± 46.242	237.845 ± 37.819	0.607 ± 0.015	228.825 ± 12.790
RF	32.921 ± 2.304	121.998 ± 3.406	192.810 ± 7.094	0.914 ± 0.067	146.416 ± 19.578
SVM	34.841 ± 6.107	170.536 ± 49.633	147.624 ± 80.249	0.737 ± 0.080	106.892 ± 47.782

prove the validity of the AE procedure. Some clarifications must be done considering the multi-step ahead forecasting. The deep learning models are trained to predict all samples of the test set at the same time, i.e. in a 'one-shot' manner, whereas for the machine learning models (RF and SVM) the situation is slightly different. They are trained to predict one sample of the test set at a time, using the last predicted sample to forecast the next one, and reiterating the process on all samples of the test set to carry out the multi-step ahead forecasting.

The numerical results are reported in Tables 7 to 10, by considering the different quality metrics. Bold numbers evidence the best results for every experiment. It is worth noting that in three out of five cases the proposed biLSTM-AE outperforms the other models. This highlights the importance of the AE procedure that provides a better starting point for the network parameters to carry out the final forecast. Tables 7 to 10 also point out that there can be a high difference in terms of the metrics among the models; this is mainly due to high absolute values that characterize some time series. Furthermore, the smallest/highest quality metric does not always reflect the best performance, as the predicted curve could not approximate exactly the real one from a visual point of view as reported in Figs. 6 to 10.

Considering the S_1 test set, our model performs better than the others. In fact, it achieves performance values that are less than half of those obtained by DeepAR and are lower, albeit by a lesser amount, than the ones obtained by the other models. On the other hand, in the S_2 test set TFT achieves the best performance; it performs slightly better than both the proposed biLSTM-AE and the standard biLSTM and outperforms, together with the latter ones, the other models. It is noteworthy that the biLSTM-AE and the biLSTM show similar performance, even if the former is more accurate and robust. It is also important to underline that, considering the MAE, the R^2 and the SNR, the performances of the proposed approach with respect to TFT are quite similar. Nevertheless, in this case, the quality metrics that provide the best understanding of performance are the RMSE and the R^2, since the former is able to penalize greater errors more, whereas the latter gives an idea about the variance of the models. The S_3 test set represents the only case where the proposed model has the worst performance by considering all the quality metrics. Even in this case, TFT achieves the best performance and it is followed by the SVM model, which is the second best. Here, the standard biLSTM performs slightly better than the biLSTM-AE, even if the latter has a lower variance that underlines its robustness. The worst performance of the biLSTM-AE could be related to the nature of the

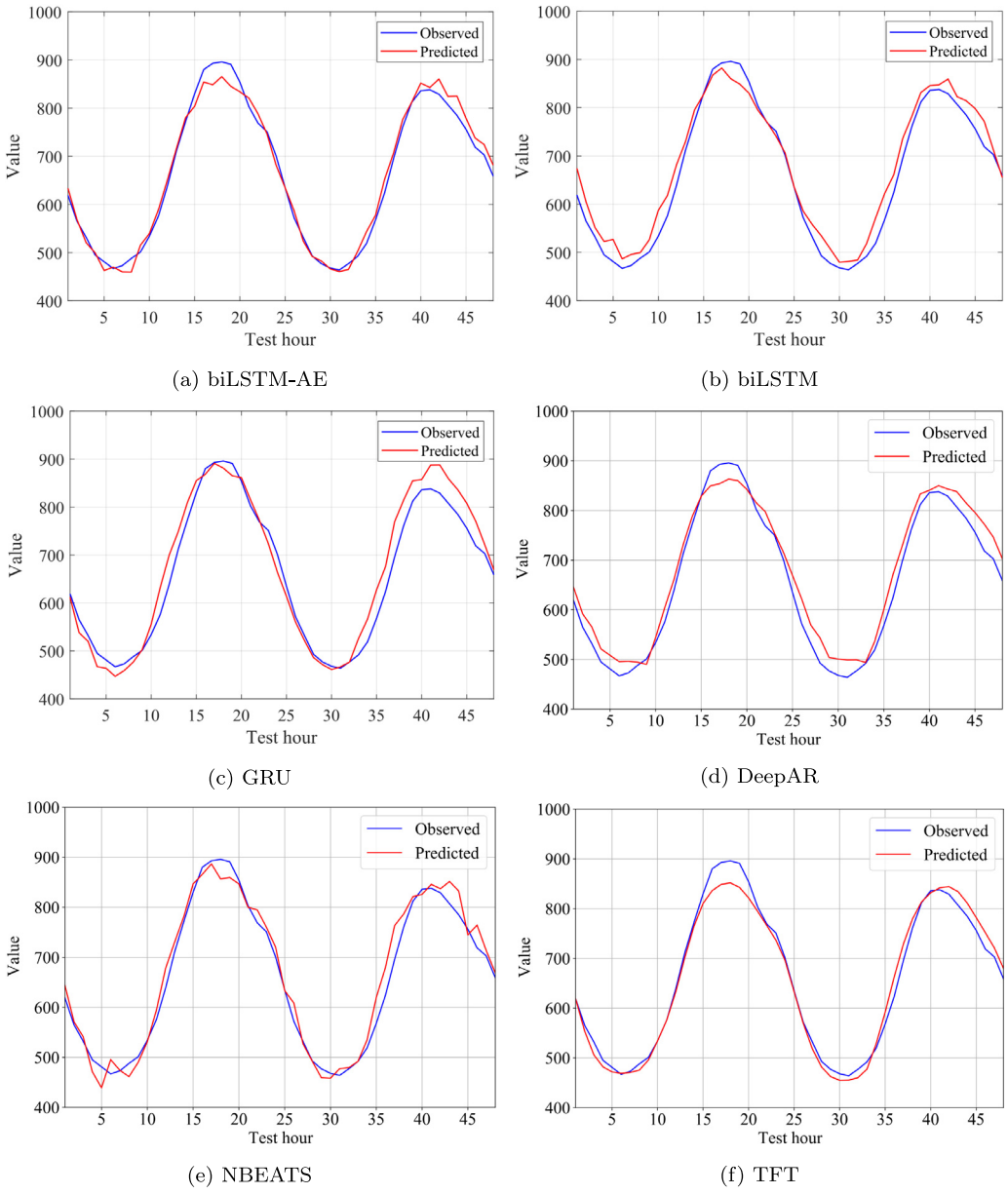

Fig. 6. Predicted (red) and observed (blue) values of S_1 test set.

Table 8
Average MAE and standard deviation of adopted predictors.

Model	S_1	S_2	S_3	S_4	S_5
biLSTM-AE	**19.531 ± 2.455**	85.228 ± 9.256	356.161 ± 21.588	**0.243 ± 0.019**	**17.041 ± 5.040**
biLSTM	26.564 ± 3.158	87.684 ± 11.517	346.108 ± 26.636	0.310 ± 0.067	40.077 ± 11.150
GRU	32.055 ± 3.126	121.083 ± 36.588	177.276 ± 22.792	0.396 ± 0.053	50.676 ± 7.498
DeepAR	48.915 ± 23.060	224.285 ± 32.791	169.699 ± 57.535	0.318 ± 0.065	126.594 ± 21.936
TFT	31.801 ± 14.191	**84.339 ± 12.871**	**56.692 ± 12.434**	0.319 ± 0.059	77.165 ± 47.893
NBEATS	24.086 ± 2.295	196.788 ± 38.927	200.225 ± 37.619	0.579 ± 0.065	175.171 ± 10.231
RF	28.901 ± 5.536	100.270 ± 3.286	158.443 ± 6.399	0.401 ± 0.028	93.353 ± 12.477
SVM	34.841 ± 6.107	151.283 ± 51.742	133.803 ± 78.911	0.607 ± 0.071	93.694 ± 35.864

time series under analysis. It may present some characteristics that can be better captured with different approaches, such as the multi-head attention block of TFT and/or the auto-regressive part of DeepAR. However, it is quite remarkable that, in all other cases, the biLSTM-AE achieves the best performance considering all metrics. In particular, in the S_4 test set the performances of the models seem very similar. This is due to the fact that the time series under analysis has relatively low values, as reported in Fig. 9. Finally, considering the S_5 test set, it is noteworthy that the proposed approach outperforms the other ones and this highlights the consistency of the AE mechanism.

Another important aspect to underline is that the proposed model achieves better performance by considering different training and test set lengths. This is another advantage given by the

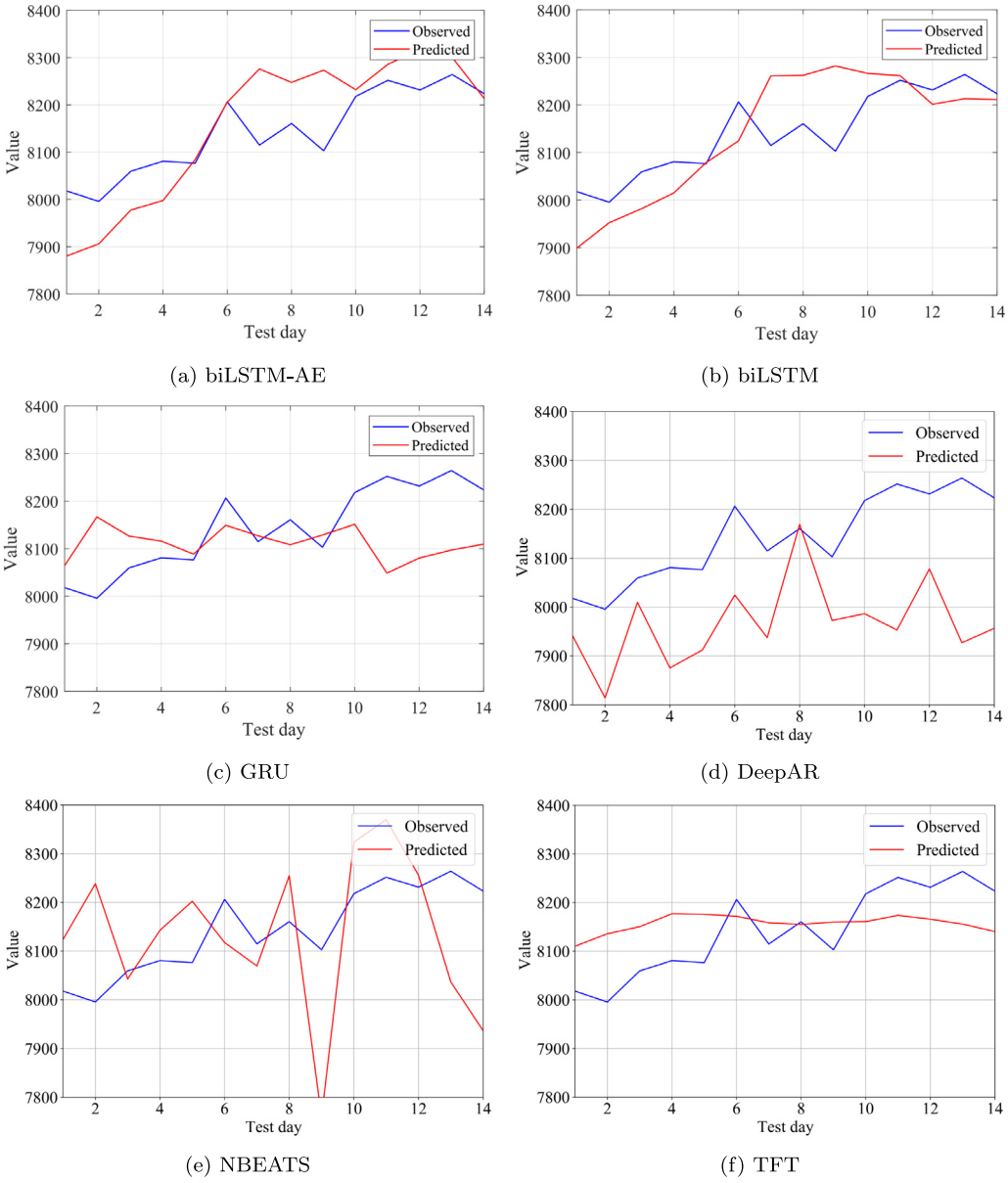

Fig. 7. Predicted (red) and observed (blue) values of S_2 test set.

Table 9
Average R^2 and standard deviation of adopted predictors.

Model	S_1	S_2	S_3	S_4	S_5
biLSTM-AE	$\mathbf{0.971 \pm 0.006}$	-0.458 ± 0.309	-9.413 ± 1.298	$\mathbf{0.887 \pm 0.019}$	$\mathbf{0.995 \pm 0.003}$
biLSTM	0.951 ± 0.012	-0.607 ± 0.493	-8.892 ± 1.541	0.751 ± 0.116	0.971 ± 0.015
GRU	0.921 ± 0.018	-1.816 ± 1.707	-2.190 ± 0.723	0.582 ± 0.110	0.951 ± 0.016
DeepAR	0.822 ± 0.191	-7.465 ± 2.678	-1.796 ± 1.472	0.800 ± 0.061	0.723 ± 0.097
TFT	0.919 ± 0.06	$\mathbf{-0.257 \pm 0.497}$	$\mathbf{0.699 \pm 0.112}$	0.697 ± 0.103	0.857 ± 0.155
NBEATS	0.957 ± 0.008	-6.860 ± 2.867	-3.450 ± 1.339	0.412 ± 0.124	0.638 ± 0.040
RF	0.948 ± 0.007	-0.971 ± 0.108	-1.836 ± 0.203	0.605 ± 0.019	0.850 ± 0.035
SVM	0.940 ± 0.020	-3.142 ± 2.190	-1.102 ± 2.042	0.101 ± 0.128	0.907 ± 0.088

AE procedure, which demonstrates that it can be applied on different datasets producing accurate results. At the same time, as highlighted by the numerical results, it could be prone to errors. This is probably due to the specific characteristics of the time series, such as the number of samples, the trend, etc. This is evident by considering the results obtained with the S_3 test set, where all the other models perform better than the proposed one.

As a general remark, it is of particular interest that the proposed approach does not present the worst standard deviation, even in the case when it shows the worst performance. This highlights the robustness of the biLSTM-AE. The model that has the best standard deviation on average is the RF, while the worst one is the SVM. Nevertheless, the numerical results reported

Table 10
Average SNR and standard deviation of adopted predictors.

Model	S_1	S_2	S_3	S_4	S_5
biLSTM-AE	**28.856 ± 0.998**	37.892 ± 0.983	24.395 ± 0.543	**12.746 ± 0.815**	**25.425 ± 2.201**
biLSTM	26.529 ± 0.851	37.556 ± 1.338	24.634 ± 0.667	9.697 ± 2.058	18.237 ± 2.333
GRU	24.523 ± 1.039	35.481 ± 2.149	29.607 ± 1.011	7.138 ± 1.063	15.677 ± 1.403
DeepAR	22.498 ± 3.487	30.354 ± 1.286	31.105 ± 3.589	10.429 ± 1.365	8.238 ± 1.777
TFT	25.626 ± 3.436	**38.689 ± 1.365**	**40.061 ± 1.629**	8.695 ± 1.632	14.247 ± 6.241
NBEATS	27.092 ± 0.764	30.811 ± 1.770	28.312 ± 1.450	5.637 ± 0.975	6.770 ± 0.491
RF	26.257 ± 0.653	36.493 ± 0.246	30.024 ± 0.328	7.269 ± 0.213	10.721 ± 1.361
SVM	25.873 ± 1.615	33.964 ± 2.837	33.555 ± 4.916	3.733 ± 0.659	14.107 ± 3.783

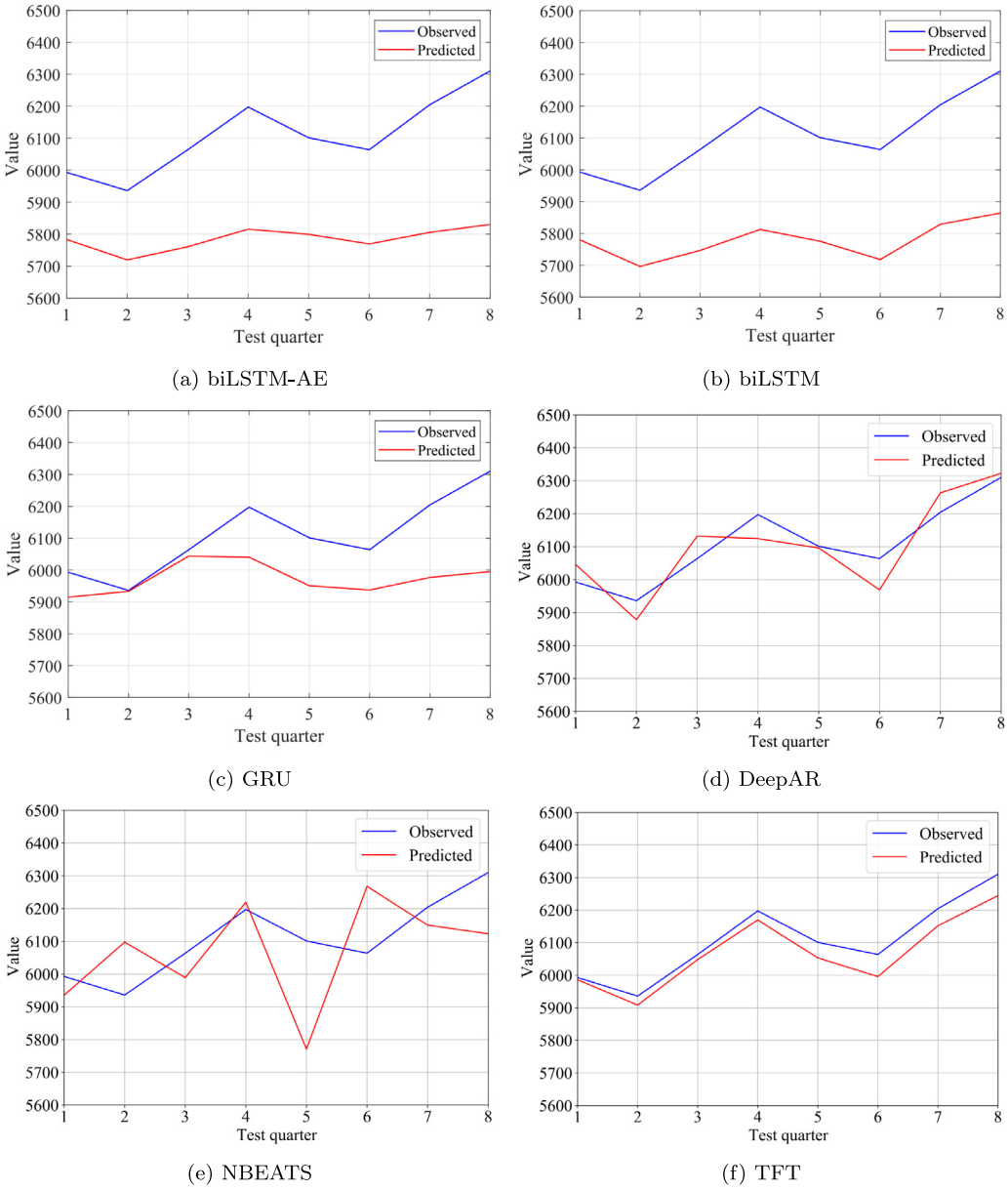

Fig. 8. Predicted (red) and observed (blue) values of S_3 test set.

in Tables 7 to 10 should be analyzed by considering a trade-off between the goodness and robustness. In these terms, it is evident that the biLSTM-AE achieves the best results, proving the efficiency of the proposed AE procedure. This is further confirmed by the visual results reported from Figs. 6 to 10. In Fig. 6(a) it is clear that the prediction related to the S_1 test set is accurate despite a little underestimation of the first peak. At the same time, the predicted curves related to the other models in Figs. 6(b), 6(c), 6(d), 6(e) and 6(f) are able to follow the real one, despite a worsening in performance. Conversely, in the S_2 test set represented in Fig. 7, the forecasting is not so good; all models struggle to correctly predict the real data. This is the case where the numerical values reported in Tables 7 to 10 do not reflect perfectly the visual results, except for the R^2 in Table 9 that, by measuring the variance of the models, gives some hint about the curves. The prediction obtained by the TFT model in

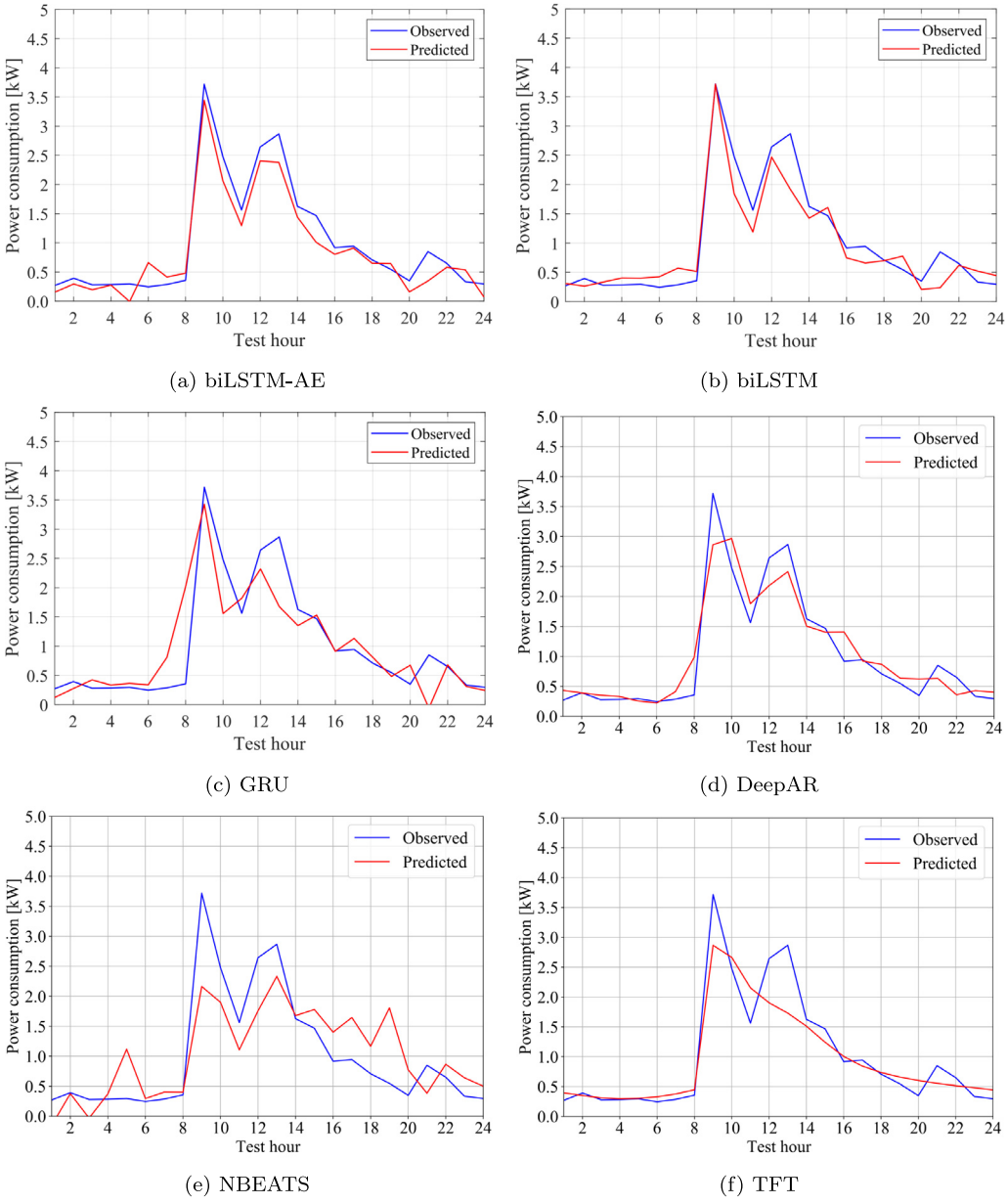

Fig. 9. Predicted (red) and observed (blue) values of S_4 test set.

Fig. 7(f), which is the best one, is not able to follow the real curve, as it is first underestimated and then overestimated; even the shape is not recognized properly. Nevertheless, the prediction falls within a range of values reasonably close to the real curve and this explain the lowest numerical values. This is not true for the S_3 test set of Fig. 8. In this case, the prediction achieves by the biLSTM-AE represented in Fig. 8(a) does not fall into a range of values near the real curve but it is able to resemble its shape. At the same time, it is important to remember that the results reported in Tables 7 to 10 are averaged over 10 runs and thus, from this point of view, it is clear that the results obtained by the other models are superior, especially considering the TFT. On the contrary, considering the test sets related to S_4 and S_5 reported in Figs. 9 and 10, respectively, it is clear that the proposed model carries out accurate forecasts. In particular, Fig. 9(a) shows how the predicted curve is able to recognize the peaks of the real curve while maintaining a good accuracy during the first and last hours of the day. The same goes for the biLSTM and the GRU, albeit to a lesser extent. In this case, despite small differences in terms of the different quality metrics, it is clear that the proposed model achieves the best result even from a visual point of view. Finally, the prediction reported in Fig. 10(a) related to the S_5 test set shows the high accuracy of the proposed model. Here, the predicted curve follows almost perfectly the real one. Only biLSTM, GRU and TFT, whose results are reported in Figs. 10(b), 10(c) and 10(f), respectively, provide a similar performance. In particular, the biLSTM underestimates the real peak whereas TFT slightly overestimates it. This is not true for the GRU, where the predicted curve slightly anticipates the real one.

6. Conclusions

A novel deep learning approach is presented in this paper. The novelty lies in the exploitation of an adaptive data embedding procedure obtained through the use of a DNN architecture based on a dual-stacked biLSTM network. The first biLSTM layer carries out the AE procedure by learning a compressed representation of the input time series, that is, the dynamics of the unknown

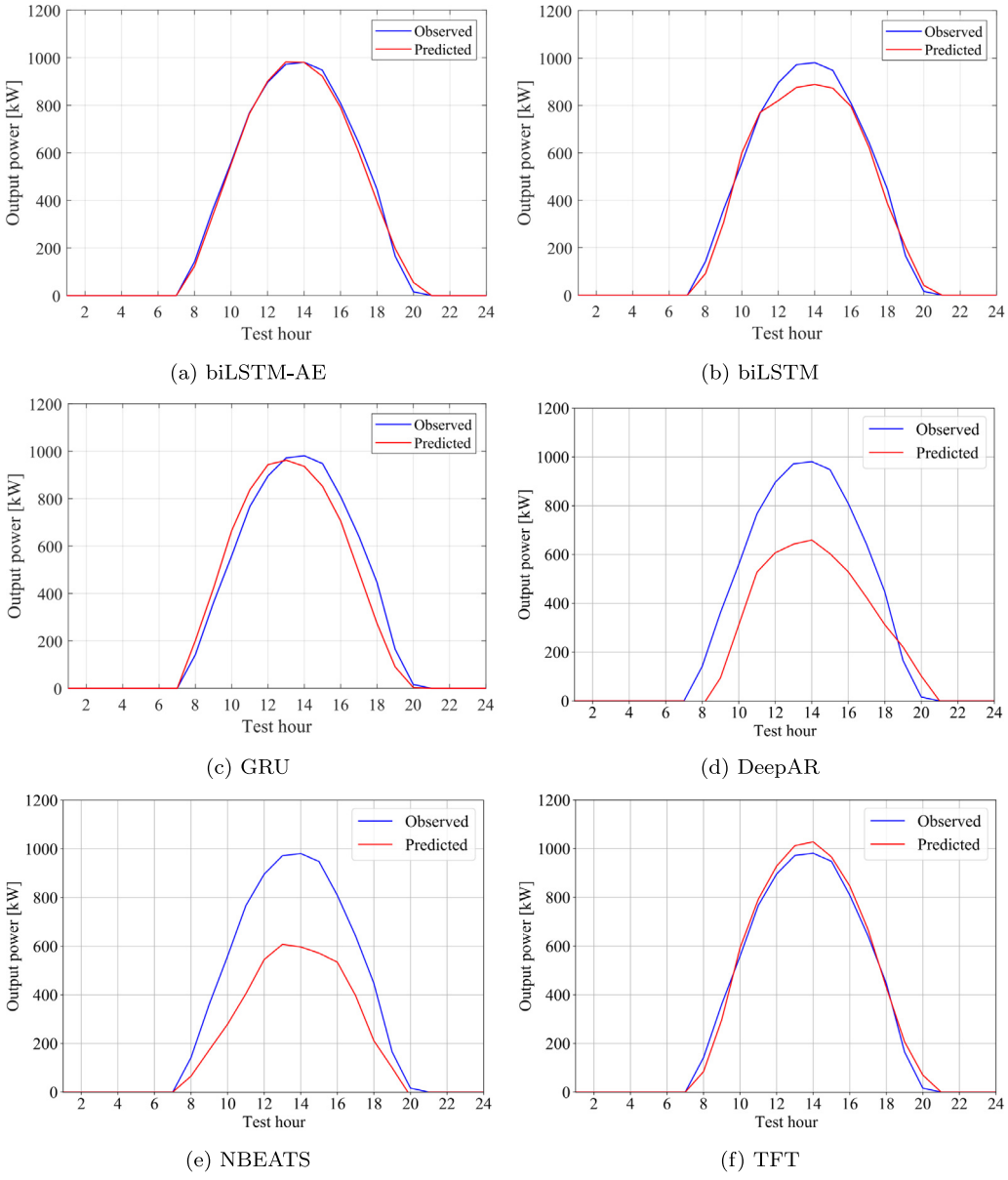

Fig. 10. Predicted (red) and observed (blue) values of S_5 test set.

system generating the observed data. Then, the second biLSTM layer is used to carry out the prediction task. The DNN is trained with a two stages approach, where the first one is used to fit the first biLSTM layer and the second one is used for the actual forecasting.

The proposed approach is extensively tested by using both benchmark and real-world time series data. The benchmarks are used to demonstrate the effectiveness of the AE procedure, whereas the real-world data are used to show the goodness and robustness of the obtained predictor. It is also compared with several state-of-the-art forecasting models, showing better performance anyway.

Future works and investigations should consider the application of the unsupervised adaptive embedding approach to different deep neural architectures and more complex scenarios. Namely, the adaptive embedding methodology can be implemented into more complex models that combines several techniques (i.e, season and trend analysis, attention mechanism, etc.) and can be tested on more challenging scenarios that consider several data sources, as for multivariate time series and distributed contexts.

CRediT authorship contribution statement

Federico Succetti: Conceptualization, Methodology, Software, Validation, Investigation, Writing – original draft, Visualization. **Antonello Rosato:** Methodology, Validation, Formal analysis, Writing – original draft, Visualization. **Massimo Panella:** Conceptualization, Methodology, Software, Data curation, Supervision.

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.

Data availability

Data will be made available on request.

References

Aggarwal, A., & Toshniwal, D. (2021). A hybrid deep learning framework for urban air quality forecasting. *Journal of Cleaner Production*, *329*, Article 129660. http://dx.doi.org/10.1016/j.jclepro.2021.129660, URL https://www.sciencedirect.com/science/article/pii/S0959652621038373.

Ashok, A., Govindarasu, M., & Ajjarapu, V. (2018). Online detection of stealthy false data injection attacks in power system state estimation. *IEEE Transactions on Smart Grid*, *9*(3), 1636–1646. http://dx.doi.org/10.1109/TSG.2016.2596298.

Bae, J., Kim, G., & Lee, S. J. (2021). Real-time prediction of nuclear power plant parameter trends following operator actions. *Expert Systems with Applications*, *186*, Article 115848. http://dx.doi.org/10.1016/j.eswa.2021.115848, URL https://www.sciencedirect.com/science/article/pii/S0957417421012094.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166. http://dx.doi.org/10.1109/72.279181.

Bottou, L., Chapelle, O., DeCoste, D., & Weston, J. (2007). Scaling learning algorithms toward AI. In *Large-scale kernel machines* (pp. 321–359). MIT Press.

Ceschini, A., Rosato, A., Succetti, F., Luzio, F. D., Mitolo, M., Araneo, R., et al. (2021). Deep neural networks for electric energy theft and anomaly detection in the distribution grid. In *2021 IEEE International conference on environment and electrical engineering and 2021 IEEE industrial and commercial power systems Europe* (pp. 1–5). http://dx.doi.org/10.1109/EEEIC/ICPSEurope51590.2021.9584796.

Chen, M.-Y., & Chen, B.-T. (2015). A hybrid fuzzy time series model based on granular computing for stock price forecasting. *Information Sciences*, *294*, http://dx.doi.org/10.1016/j.ins.2014.09.038.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555.

Deb, C., Zhang, F., Yang, J., Lee, S., & Kwok Wei, S. (2017). A review on time series forecastingtechniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, *74*, http://dx.doi.org/10.1016/j.rser.2017.02.085.

Erhan, D., Courville, A., Bengio, Y., & Vincent, P. (2010). Why does unsupervised pre-training help deep learning? In Y. W. Teh, & M. Titterington (Eds.), *Proceedings of machine learning research*: *vol. 9*, *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 201–208). Chia Laguna Resort, Sardinia, Italy: PMLR, URL https://proceedings.mlr.press/v9/erhan10a.html.

Fan, R.-E., Chen, P.-H., Lin, C.-J., & Joachims, T. (2005). Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, *6*(12).

Fan, G.-F., Zhang, L.-Z., Yu, M., Hong, W.-C., & Dong, S.-Q. (2022). Applications of random forest in multivariable response surface for short-term load forecasting. *International Journal of Electrical Power & Energy Systems*, *139*, Article 108073.

Flunkert, V., Salinas, D., & Gasthaus, J. (2017). DeepAR: Probabilistic forecasting with autoregressive recurrent networks, CoRR abs/1704.04110. URL http://arxiv.org/abs/1704.04110.

Gensler, A., Henze, J., Sick, B., & Raabe, N. (2016). Deep learning for solar power forecasting — An approach using AutoEncoder and LSTM neural networks. In *2016 IEEE international conference on systems, man, and cybernetics* (pp. 002858–002865). http://dx.doi.org/10.1109/SMC.2016.7844673.

Gers, F. A., Eck, D., & Schmidhuber, J. (2002). Applying LSTM to time series predictable through time-window approaches. In R. Tagliaferri, & M. Marinaro (Eds.), *Neural nets WIRN Vietri-01* (pp. 193–200). London: Springer London.

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

Kennel, Brown, & Abarbanel (1992). Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical Review. A, Atomic, Molecular, and Optical Physics*, *45 6*, 3403–3411.

Khan, N., Ullah, F. U. M., Haq, I., Khan, S., Lee, M., & Baik, S. (2021). AB-net: A novel deep learning assisted framework for renewable energy generation forecasting. *Mathematics*, *9*, http://dx.doi.org/10.3390/math9192456.

Kim, J.-Y., & Cho, S.-B. (2021). Explainable prediction of electric energy demand using a deep autoencoder with interpretable latent space. *Expert Systems with Applications*, *186*, Article 115842. http://dx.doi.org/10.1016/j.eswa.2021.115842.

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. In *International conference on learning representations*.

Lee, S., Jin, H., Nengroo, S. H., Doh, Y., Lee, C., Heo, T., et al. (2021). Smart metering system capable of anomaly detection by bi-directional LSTM autoencoder, CoRR abs/2112.03275. URL https://arxiv.org/abs/2112.03275.

Li, W., & Becker, D. M. (2021). Day-ahead electricity price prediction applying hybrid models of LSTM-based deep learning methods and feature selection algorithms under consideration of market coupling. *Energy*, *237*, Article 121543. http://dx.doi.org/10.1016/j.energy.2021.121543, URL https://www.sciencedirect.com/science/article/pii/S0360544221017916.

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2017). Graph convolutional recurrent neural network: Data-driven traffic forecasting, CoRR abs/1707.01926. URL http://arxiv.org/abs/1707.01926.

Lim, B., Arik, S. O., Loeff, N., & Pfister, T. (2019). Temporal fusion transformers for interpretable multi-horizon time series forecasting. http://dx.doi.org/10.48550/ARXIV.1912.09363, URL https://arxiv.org/abs/1912.09363.

Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, *20*(2), 130–141. http://dx.doi.org/10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2, URL https://journals.ametsoc.org/view/journals/atsc/20/2/1520-0469_1963_020_0130_dnf_2_0_co_2.xml.

Mackey, M. C., & Glass, L. (1977). Oscillation and chaos in physiological control systems. *Science*, *197*(4300), 287–289. http://dx.doi.org/10.1126/science.267326.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. *PLoS One*, *13*(3), 1–26. http://dx.doi.org/10.1371/journal.pone.0194889.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, *36*(1), 54–74. http://dx.doi.org/10.1016/j.ijforecast.2019.04.014, M4 Competition, URL https://www.sciencedirect.com/science/article/pii/S0169207019301128.

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2019). N-BEATS: neural basis expansion analysis for interpretable time series forecasting, CoRR abs/1905.10437. URL http://arxiv.org/abs/1905.10437.

Padnekar, S. M., Kumar, G. S., & Deepak, P. (2020). Bilstm-autoencoder architecture for stance prediction. In *2020 international conference on data science and engineering* (pp. 1–5). http://dx.doi.org/10.1109/ICDSE50459.2020.9310133.

Polu, S. (2022). Residential power usage 3years data – timeseries. https://www.kaggle.com/srinuti/residential-power-usage-3years-data-timeseries. (Accessed 01 February 2022).

Reyes, O., & Ventura, S. (2019). Performing multi-target regression via a parameter sharing-based deep network. *International Journal of Neural Systems*, http://dx.doi.org/10.1142/S012906571950014X.

Rong Liu, W. H., & Huang, Q. (2021). Short-term load forecasting based on LSTNet in power system. *International Transactions on Electrical Energy Systems*, *31*, http://dx.doi.org/10.1002/2050-7038.13164.

Rosato, A., Panella, M., Andreotti, A., Mohammed, O., & Araneo, R. (2021). Two-stage dynamic management in energy communities using a decision system based on elastic net regularization. *Applied Energy*, *291*, Article 116852. http://dx.doi.org/10.1016/j.apenergy.2021.116852.

Rössler, O. (1976). An equation for continuous chaos. *Physics Letters. A*, *57*(5), 397–398. http://dx.doi.org/10.1016/0375-9601(76)90101-8, URL https://www.sciencedirect.com/science/article/pii/0375960176901018.

Sagheer, A., & Kotb, M. (2019). Unsupervised pre-training of a deep LSTM-based stacked autoencoder for multivariate time series forecasting problems. *Scientific Reports*, *9*, 19038. http://dx.doi.org/10.1038/s41598-019-55320-6.

Succetti, F., Rosato, A., Araneo, R., & Panella, M. (2020). Deep neural networks for multivariate prediction of photovoltaic power time series. *IEEE Access*, *8*, 211490–211505. http://dx.doi.org/10.1109/ACCESS.2020.3039733.

Takahashi, K., Ooka, R., & Ikeda, S. (2021). Anomaly detection and missing data imputation in building energy data for automated data pre-processing. *Journal of Physics: Conference Series*, *2069*(1), Article 012144. http://dx.doi.org/10.1088/1742-6596/2069/1/012144.

Tong, X., Wang, J., Zhang, C., Wu, T., Wang, H., & Wang, Y. (2022). LS-LSTM-AE: Power load forecasting via long-short series features and LSTM-autoencoder. *Energy Reports*, *8*, 596–603. http://dx.doi.org/10.1016/j.egyr.2021.11.172, 2021 The 8th International Conference on Power and Energy Systems Engineering, URL https://www.sciencedirect.com/science/article/pii/S2352484721013196.

Yang, X., Wan, C., Zhang, T., & Xiong, Z. (2022). Feature extraction of sequence data based on LSTM and its application to fault diagnosis of industrial process. In *2022 IEEE 11th data driven control and learning systems conference* (pp. 693–698). http://dx.doi.org/10.1109/DDCLS55054.2022.9858505.

Yang, L., Zhai, Y., & Li, Z. (2021). Deep learning for online AC false data injection attack detection in smart grids: An approach using LSTM-autoencoder. *Journal of Network and Computer Applications*, *193*, Article 103178. http://dx.doi.org/10.1016/j.jnca.2021.103178, URL https://www.sciencedirect.com/science/article/pii/S1084804521001880.

Zhao, H., Deng, K., Li, N., Wang, Z., & Wei, W. (2020). Hierarchical spatial-spectral feature extraction with long short term memory (LSTM) for mineral identification using hyperspectral imagery. *Sensors*, *20*(23), http://dx.doi.org/10.3390/s20236854, URL https://www.mdpi.com/1424-8220/20/23/6854.

Zhao, Y., Ye, L., Li, Z., Song, X., Lang, Y., & Su, J. (2016). A novel bidirectional mechanism based on time series model for wind power forecasting. *Applied Energy*, *177*, 793–803. http://dx.doi.org/10.1016/j.apenergy.2016.03.096.

Zhu, B., Ye, S., Wang, P., Chevallier, J., & Wei, Y.-M. (2022). Forecasting carbon price using a multi-objective least squares support vector machine with mixture kernels. *Journal of Forecasting*, *41*(1), 100–117.