Expert Systems With Applications 205 (2022) 117721

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Online learning of windmill time series using Long Short-term Cognitive Networks

Alejandro Morales-Hernández [a], Gonzalo Nápoles [b,*], Agnieszka Jastrzebska [c], Yamisleydi Salgueiro [d], Koen Vanhoof [a]

[a] Business Informatics Research Group, Hasselt University, Belgium
[b] Department of Cognitive Science & Artificial Intelligence, Tilburg University, The Netherlands
[c] Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland
[d] Department of Computer Science, Faculty of Engineering, Universidad de Talca, Campus Curicó, Chile

ARTICLE INFO

Keywords:
Long Short-term Cognitive Network
Recurrent Neural Network
Multivariate time series
Forecasting

ABSTRACT

Forecasting windmill time series is often the basis of other processes such as anomaly detection, health monitoring, or maintenance scheduling. The amount of data generated by windmill farms makes online learning the most viable strategy to follow. Such settings require retraining the model each time a new batch of data is available. However, updating the model with new information is often very expensive when using traditional Recurrent Neural Networks (RNNs). In this paper, we use Long Short-term Cognitive Networks (LSTCNs) to forecast windmill time series in online settings. These recently introduced neural systems consist of chained Short-term Cognitive Network blocks, each processing a temporal data chunk. The learning algorithm of these blocks is based on a very fast, deterministic learning rule that makes LSTCNs suitable for online learning tasks. The numerical simulations using a case study involving four windmills showed that our approach reported the lowest forecasting errors with respect to a simple RNN, a Long Short-term Memory, a Gated Recurrent Unit, and a Hidden Markov Model. What is perhaps more important is that the LSTCN approach is significantly faster than these state-of-the-art models.

1. Introduction

Humanity's sustainable development requires the adoption of less environmentally aggressive energy sources. Over the last years, renewable energy sources (RES) have increased their presence in the energy matrix of several countries. These clean energies are less polluting, renewable, and abundant in nature. However, limitations such as volatility and intermittency reduce their reliability and stability for power systems. This hinders the integration of renewable sources into the main grid and increases their generation costs (Sinsel et al., 2020).

Power generation forecasting (Foley et al., 2012) is one of the approaches adopted to facilitate the optimal integration of RES in power systems. Overall, the goal of power generation forecasting is to know in advance the possible disparity between generation and demand due to fluctuations in energy sources (Ahmed & Khalid, 2019). Forecasting methods used for renewable energies are based on physical, statistical, or machine learning (ML) models. Although ML models often achieve the highest performance compared to other models, their deployment in real applications is limited (Jorgensen & Shaker, 2020). On the one hand, most ML models require feature engineering before building the model and lack interpretability. On the other hand, these methods usually assume that the training data is completely available in advance. Hence, most ML methods are unable to incorporate new information into the previously constructed models (Wang et al., 2019).

Within clean energy approaches, wind energy has shown sustained growth in installed capacity and exploitation in recent years (Ahmed et al., 2020). However, wind energy involves some peculiarities to be considered when designing new forecasting solutions. Firstly, wind-based power generation can heavily be affected by weather variability, which means that the power generation fluctuates with extreme weather phenomena (i.e., frontal systems or rapidly evolving low-pressure systems). Weather events are unavoidable, but their impact can be minimized if anticipated in advance. Secondly, wind generators are dynamic systems that behave differently over time (i.e., due to wear of turbine components, maintenance, etc.). Finally, the data generated by windmills is not static since they will continue to operate, thus producing new pieces of data. These characteristics make traditional ML

* Corresponding author.
E-mail addresses: alejandro.moraleshernandez@uhasselt.be (A. Morales-Hernández), g.r.napoles@uvt.nl (G. Nápoles), a.jastrzebska@mini.pw.edu.pl (A. Jastrzebska), ysalgueiro@utalca.cl (Y. Salgueiro), koen.vanhoof@uhasselt.be (K. Vanhoof).

https://doi.org/10.1016/j.eswa.2022.117721
Received 10 July 2021; Received in revised form 25 May 2022; Accepted 31 May 2022
Available online 6 June 2022

methods inadequate to model the dynamics of these systems properly. This means that new approaches are needed to improve the prediction of wind generation. The development of algorithms capable of learning beyond the production phase will also allow them to be kept up-to-date at all times (Losing et al., 2018).

Recently, Nápoles et al. (2021) introduced a recurrent neural system termed *Long Short-term Cognitive Network* (LSTCN) that seems suitable for online learning setting where data might be volatile. Moreover, the cognitive component of such a recurrent neural network allows for interpretability and it is given by two facts. Firstly, neural concepts and weights have a well-defined meaning for the problem domain being modeled. This means that the resulting model can easily be interpreted with little effort. For example, in Nápoles et al. (2021) the authors discussed a measure to compute the relevance of each variable in multivariate time series without the need for any post-hoc method. Secondly, the domain expert can insert knowledge into the network by modifying the prior knowledge matrix, which is not altered during the learning process. For example, we can modify some connections in the weight matrix to manually encode patterns that have not yet been observed in the data or that were observed under exceptional circumstances.

Despite the advantages of the LSTCN model when it comes to its forecasting capabilities, intrinsic interpretability and short training time, it has not yet been applied to a real-world problem, as far as we know. In addition, we have little knowledge of the performance of this brand new model on online learning settings operating with volatile data that might be shortly available. Such a lack of knowledge and the challenges related to the wind prediction described above have motivated us to study the LSTCNs' performance on a real-world problem concerning the power forecasting of four windmills.

More explicitly, this paper elaborates on the task of forecasting power generation in windmills using the LSTCN model. By doing that, we propose an LSTCN-based pipeline to tackle the related online learning problem where each data chuck is processed only once. In this pipeline, each iteration processes a data chunk using a Short-term Cognitive Network (STCN) block (Nápoles et al., 2019) that operates with the knowledge transferred from the previous block. This means that the model can be retrained without compromising what the network has learned from previous data chunks. The numerical simulations show that our solution (i) outperforms state-of-the-art recurrent neural networks when it comes to the forecasting error and (ii) reports significantly shorter training and test times.

The remainder of the paper is organized as follows. Section 2 revises the literature about recurrent neural networks used to forecast windmill time series. Section 3 presents the proposed LSTCN-based power forecasting model for an online learning setting. Section 4 describes the case study, the state-of-the-art recurrent models used for comparison purposes and the simulation results. Finally, Section 5 concludes the paper and suggests further research directions to be explored.

2. Forecasting models with recurrent neural networks

Neural networks are a family of biology-inspired computational models that have found applications in many fields. An example of engineering applications of neural models is the support of wind turbine operation and maintenance. In this area, neural models dedicated to the analysis of temporal data have proven to be quite useful. This is motivated by the fact that typical data describing the operation of a wind turbine are collected by sensors forming a supervisory control and data acquisition (SCADA) system (Du et al., 2017; Weerakody et al., 2021). Such data come in the form of long sequences of numerical values, thus making Recurrent Neural Networks (RNNs) the right choice for processing such data. This section briefly revises the literature on the applications of RNNs on data analysis in the area of wind turbine operation and maintenance support.

RNNs differ from other neural networks in the way the input data is propagated. In standard neural networks, the input data is processed in a feed-forward manner, meaning the signal is transmitted unidirectionally. In RNN models, the signal goes through neurons that can have backward connections from further layers to earlier layers (Che et al., 2018). Depending on a particular neural model architecture, we can restrict the layers with feedback connections to only selected ones. The overall idea is to allow the network to "revisit" nodes, which mimics the natural phenomenon of memory (Kong et al., 2019). RNNs turned out to be useful for accurate time series prediction tasks (Strobelt et al., 2018), including wind turbine time series prediction (Cui et al., 2021).

As reported by Zhang et al. (2020), the task of analyzing wind turbine data often involves building a regression model operating on multi-attribute data from SCADA sensors. Such models can help us understand the data (Delgado & Fahim, 2021; Janssens et al., 2016).

Currently, the most popular variant of RNN in the field of wind turbine data processing is the Long Short-Term Memory (LSTM) model (Hochreiter & Schmidhuber, 1997; Mishra et al., 2020). In this model, the inner operations are defined by neural gates called *cell*, *input gate*, *output gate*, and *forget gate*. The cell acts as the memory, while the other components determine the way the signal propagates through the neural architecture (Zhang et al., 2018). The introduction of these specialized units helped prevent (to some extent) the gradient problems associated with training RNN models (Sherstinsky, 2020).

Existing neural network approaches to wind turbine data forecasting do not pay enough attention to the issue of model complexity and efficiency. In most studies, authors reduce the available set of input variables rather than optimizing the neural architecture used. For example, Feng et al. (2019) used the LSTM model with hand-picked three SCADA input variables, while Riganti-Fulginei et al. (2018) used eleven SCADA variables. Qian et al. (2019) also used LSTM to predict wind turbine data. In their study, the initial set of input variables consisted of 121 series, but this was later reduced to only three variables and then to two variables using the Mahalanobis distance method. The issue of preprocessing and feature selection was also raised by Wang et al. (2018), suggesting Principal Component Analysis to reduce the dimensionality of the data.

LSTM has been found to perform well even when the time series variables are of incompatible types. It is worth citing the study of Lei et al. (2019), who used LSTM to predict two qualitatively different types of time series simultaneously: (i) vibration measurements that have a high sampling rate and (ii) slow varying measurements (e.g., bearing temperature). It should be noted that existing studies bring additional techniques that enhance the capabilities of the standard LSTM model. For example, Cao, Zhang et al. (2019) propose segmenting the data and using segment-related features instead of raw signals. Xiang et al. (2021) also do not use raw signals. Instead, they use Convolutional Neural Networks (CNNs) to extract the dynamic features of the data, which is then fed to LSTM. A similar approach, combining CNN with LSTM, was presented by Xue et al. (2021). Another interesting technique was introduced by Chen et al. (2021), who combined LSTM with an auto-encoder (AE) neural network so that their model can detect and reject anomalies while achieving better results for non-anomalous data. Liu et al. (2020) used wavelet decomposition together with LSTM and found that it achieves better results than standard LSTM, but this comes at the cost of increased time complexity (training time increases by about 30%). Other studies on LSTM and wind power prediction have focused on tuning the LSTM architecture, for example, by testing different transformation functions (Yin et al., 2020) or by adding a specialized imputation module for missing data (Li et al., 2019).

In addition, the bidirectional LSTM model (Gers et al., 2002) has also been applied to forecast wind turbine data. The application of this model was found in the study of Zhen et al. (2020) and, in a deeper architecture, in the study of Cao, Qian et al. (2019).

While most of the recently published studies using neural models to predict multivariate wind turbine time series employ LSTM, there are also several alternative approaches focusing on other RNN variants. For example, there are several papers on the use of Elman neural networks in forecasting multivariate wind turbine data (Lin, 2013, 2016). Kramti et al. (2018) also applied Elman neural networks, but using a slightly modified architecture. Likewise, we should mention the work of López et al. (2020), which involved Echo State Network and LSTM. Finally, it is worth mentioning the work of Kong et al. (2020), in which the task of processing data from wind turbines is implemented using CNNs and Gated Recurrent Unit (GRU) (Cho et al., 2014). The latter neural architecture is a variant of RNN, which can be seen as a simplification of the LSTM architecture. GRU was also used in the study of Niu et al. (2020), which employs attention mechanisms to reduce the forecasting error.

There are other models equipped with reasoning mechanisms similar to the one used by neural networks. In particular, the concept of "neuron" can also be found in Hidden Markov Models (HMMs) (Rabiner, 1989). Such neurons are implemented as *states*, and the set of states essentially plays a role analogous to that of hidden neurons in a standard neural network. HMMs have also found applications in wind power forecasting. The studies of Bhaumik et al. (2019) and Qu et al. (2021) should be mentioned in this context. Both research teams highlight decent predictions and robustness to noise in the data.

As pointed out by Manero et al. (2018), the task of comparing wind energy forecasting approaches described in the literature is challenging due to several factors such as the differences in time series datasets, the alternative forecast horizons, etc. In this paper, we will conduct experiments for key state-of-the-art models for our data alongside the LSTCN formalism. The methodology adopted allows us to draw conclusions about the forecasting accuracy of different models and compare their empirical computational complexity.

3. Long Short-term Cognitive Network

This section elaborates on the LSTCN model used for online learning of multivariate time series. The first sub-section will explain how to prepare the data to simulate an online learning problem, while the remaining ones will introduce the network architecture and the learning algorithm.

3.1. Data preparation for online learning simulations

Let $x \in \mathbb{R}$ be a variable observed over a discrete time scale within a period $t \in \{1, 2, \dots, T\}$ where $T \in \mathbb{N}$ is the number of observations. Hence, a univariate time series can be defined as a sequence of observations $\{x^{(t)}\}_{t=1}^{T} = \{x^{(1)}, x^{(2)}, \dots, x^{(T)}\}$. Similarly, we can define a multivariate time series as a sequence $\{X^{(t)}\}_{t=1}^{T} = \{X^{(1)}, X^{(2)}, \dots, X^{(T)}\}$ of vectors of M variables, such that $X^{(t)} = [x_1^{(t)}, x_2^{(t)}, \dots, x_M^{(t)}]$. A model F is used to forecast the next $L < T$ steps ahead. In this paper, we assume that the model F is built as a sequence of neural blocks with local learning capabilities, each able to capture the trends in the current time patch (i.e., a chunk of the time series) being processed. Both the network architecture and the parameter learning algorithm will be detailed in the following sub-sections.

Let us assume that $X \in \mathbb{R}^{M \times T}$ is a dataset comprising a multivariate time series (Fig. 1(a)). Firstly, we need to transform X into a set of Q tuples with the form $(X^{(t-R)}, X^{(t+L)}), t - R > 0, t + L \leq T$ where R represents how many past steps we will use to forecast the following L steps ahead (see Fig. 1(b)). In this paper, we assume that $R = L$ for the sake of simplicity. Secondly, each component in the tuple is flattened such that we obtain a $Q \times (M(R+L))$ matrix. Finally, we create a partition $P = \{P^{(1)}, \dots, P^{(k)}, \dots, P^{(K)}\}$ from the set of flattened tuples such that $P^{(k)} = (P_1^{(k)}, P_2^{(k)})$ is the kth time patch involving two data pieces $P_1^{(k)}, P_2^{(k)} \in \mathbb{R}^{C \times N}$, where $N = MR$ and C denotes the number of instances in that time patch.

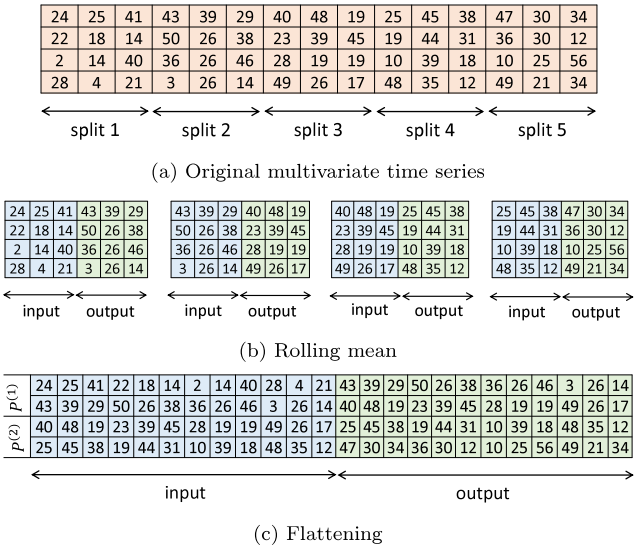


Fig. 1. Data pre-processing using $R = L = 3$. (a) The original multivariate time series $X \in \mathbb{R}^{M \times T}$, with rows as variables and columns as timestamps. (b) Selection of sub-sequences of the time series according to parameters R and L. (c) Each sub-sequence is flattened to obtain the temporal instances. In this example, the flattened dataset is divided into two time parches.

Fig. 1 shows an example of such a pre-processing method. First, the times series is split into chunks of equal length as defined by the L and R parameters. Second, we use the resulting chunks to create a set of input–output pairs. Finally, we flatten these pairs to obtain the tuples with the inputs to the network and the corresponding expected outputs.

It should be highlighted that the forecasting model will have access to a time patch in each iteration, as it usually happens in an online scenario. If the neural model is fed with several time steps, then it will be able to forecast multiple-step ahead of all variables describing the time series.

3.2. Network architecture and neural reasoning

In the online learning setting, we consider a time series (regardless of the number of observed variables) as a sequence of time patches of a certain length. Such a sequence refers to the partition $P = \{P^{(1)}, \dots, P^{(k)}, \dots, P^{(K)}\}$ obtained with the data preparation steps discussed in the previous subsection. Hence, the proposed network architecture consists of an LSTCN model able to process the sequence of time patches.

An LSTCN model can be defined as a collection of STCN blocks, each processing a specific time patch and transferring knowledge to the following STCN block in the form of weight matrices. Fig. 2 shows the recurrent pipeline of an LSTCN involving three STCN blocks to model a multivariate time series decomposed into three time patches. It should be highlighted that learning happens inside each STCN block to prevent the information flow from vanishing as the network processes more time patches. Moreover, weights estimated in the current STCN block are transferred to the following STCN block to perform the next reasoning process (see Fig. 3). These weights will no longer be modified in subsequent learning processes, which allow preserving the knowledge we have learned up to the current time patch. That makes our approach suitable for the online learning setting.

The reasoning within an STCN block involves two gates: the *input gate* and the *output gate*. The input gate operates the prior knowledge matrix $W_1^{(k)} \in \mathbb{R}^{N \times N}$ with the input data $P_1^{(k)} \in \mathbb{R}^{C \times N}$ and the prior bias matrix $B_1^{(k)} \in \mathbb{R}^{1 \times N}$ denoting the bias weights. Both matrices $W_1^{(k)}$ and $B_1^{(k)}$ are transferred from the previous block and remain locked during the learning phase to be performed in that STCN block. The result of

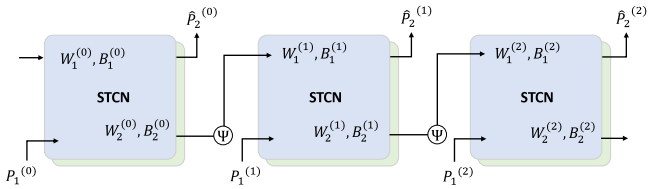


Fig. 2. LSTCN architecture of three STCN blocks. The weights learned in the current block are transferred to the following STCN block as a prior knowledge matrix.

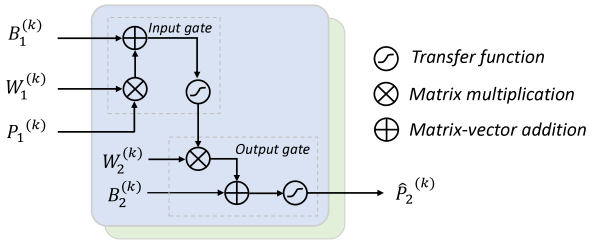


Fig. 3. Reasoning within an STCN block. Firstly, the current time patch is mixed with the prior knowledge matrices $W_1^{(k)}$ and $B_1^{(k)}$. This operation produces a temporal state matrix $H^{(k)}$. Secondly, we operate the $H^{(k)}$ matrix with the matrices $W_2^{(k)}$ and $B_2^{(k)}$. The result of such an operation will be an approximation of the expected output $P_2^{(k)}$.

the input gate is a temporal state $H^{(k)} \in \mathbb{R}^{C\times N}$ that represents the outcome that the block would have produced given $P_1^{(k)}$ if the block would not have been adjusted to the block's expected output $P_2^{(k)}$. Such an adaptation is done in the output gate where the temporal state is operated with the matrices $W_2^{(k)} \in \mathbb{R}^{N\times N}$ and $B_2^{(k)} \in \mathbb{R}^{1\times N}$, which contain learnable weights. Fig. 3 depicts the reasoning process within the kth block.

Eqs. (1) and (2) show the short-term reasoning process of this model in the kth iteration,

$$\hat{P}_2^{(k)} = f\left(H^{(k)}W_2^{(k)} \oplus B_2^{(k)}\right) \tag{1}$$

and

$$H^{(k)} = f\left(P_1^{(k)}W_1^{(k)} \oplus B_1^{(k)}\right) \tag{2}$$

where $f(x) = \frac{1}{1+e^{-x}}$, whereas $\hat{P}_2^{(k)}$ is an approximation of the expected block's output. In these equations, the $\oplus$ operator performs a matrix–vector addition by operating each row of a given matrix with a vector, provided that both the matrix and the vector have the same number of columns. Notice that we assumed that values to be forecast are in the $[0, 1]$ interval.

As mentioned, the LSTCN model consists of a sequential collection of STCN blocks. In this neural system, the knowledge from one block is passed to the next one using an aggregation procedure (see Fig. 2). This aggregation operates on the knowledge learned in the previous block (that is to say, the $W_2^{(k-1)}$ matrix). In this paper, we use the following non-linear operator in all our simulations:

$$W_1^{(k)} = \Psi(W_2^{(k-1)}), k-1 \geq 0 \tag{3}$$

and

$$B_1^{(k)} = \Psi(B_2^{(k-1)}), k-1 \geq 0 \tag{4}$$

such that $\Psi(x) = tanh(x)$. However, we can design operators combining the knowledge in both $W_1^{(k-1)}$ and $W_2^{(k-1)}$.

There is an important detail to be discussed. Once we have processed the available sequence (i.e., performed K short-term reasoning steps with their corresponding learning processes), the whole LSTCN model will narrow down to the last STCN block. Therefore, that network will be used to forecast new data chunks as they arrive and a new learning process will follow, as needed in online learning settings.

3.3. Parameter learning

Training the LSTCN in Fig. 2 means training each STCN block with its corresponding time patch. The learning process within a block is partially independent of other blocks as it only uses the prior weights matrices that are transferred from the previous block. As mentioned, these prior knowledge matrices are used to compute the temporal state and are not modified during the block's learning process.

The learning task within an STCN block can be summarized as follows. Given a temporal state $H^{(k)}$ resulting from the input gate and the block's expected output $P_2^{(k)}$, we need to compute the matrices $W_2^{(k)} \in \mathbb{R}^{N\times N}$ and $B_2^{(k)} \in \mathbb{R}^{1\times N}$.

Mathematically speaking, the learning is performed by solving a system of linear equations that adapt the temporal state to the expected output. Eq. (5) displays the deterministic learning rule solving this regression problem,

$$\begin{bmatrix} W_2^{(k)} \\ B_2^{(k)} \end{bmatrix} = \left(\left(\boldsymbol{\Phi}^{(k)}\right)^\top \boldsymbol{\Phi}^{(k)} + \lambda\Omega^{(k)}\right)^{-1} \left(\boldsymbol{\Phi}^{(k)}\right)^\top f^-\left(P_2^{(k)}\right) \tag{5}$$

where $\boldsymbol{\Phi}^{(k)} = (H^{(k)}|A)$ such that $A_{C\times 1}$ is a column vector filled with ones, $\Omega^{(k)}$ denotes the diagonal matrix of $(\boldsymbol{\Phi}^{(k)})^\top\boldsymbol{\Phi}^{(k)}$, while $\lambda \geq 0$ denotes the ridge regularization penalty. This learning rule assumes that the neuron's activation values inner layer are standardized. When the final weights are returned, they are adjusted back into their original scale.

It shall be noted that we need to specify $W_1^{(0)}$ and $B_1^{(0)}$ in the first STCN block. We can use a transfer learning approach from a previous learning process or it can be provided by domain experts. Since this information is not available, we fit a single STCN block without an intermediate state (i.e., $H^{(0)} = P_1^{(0)}$) on a smoothed representation of the whole (available) time series. The smoothed time series is obtained using the moving average method for a given window size.

Fig. 4 portrays the workflow of the iterative learning process of an LSTCN model. An incoming chunk of data triggers a new training process on the last STCN block using the stored knowledge that the network has learned in previous iterations. After that, the prior knowledge matrices are recomputed using an aggregation operator and stored to be used as prior knowledge when performing reasoning.

4. Numerical simulations

In this section, we will explore the performance (forecasting error and training time) of the proposed LSTCN-based online forecasting model for windmill time series.

4.1. Description of windmill datasets

To conduct our experiments, we adopted four public datasets from the ENGIE web page.[1] Each dataset corresponds to a windmill where measurements were recorded every 10 min from 2013 to 2017. The time series of each windmill contains 264,671 timestamps. Eight variables concerning the windmill and environmental conditions were selected: *generated power, rotor temperature, rotor bearing temperature, gearbox inlet temperature, generator stator temperature, wind speed, outdoor temperature,* and *nacelle temperature*.

As of the pre-processing steps, we removed duplicated timestamps, imputed missing timestamps and values, and applied a min–max normalization. Moreover, the data preparation procedure described in Fig. 1 was applied to each dataset. Table 1 displays a descriptive summary of all datasets after normalization where the minimum, median and maximum of the absolute Pearson's correlation values among the variables are denoted as *min*, *med*, *max*, respectively.

[1] https://opendata-renewables.engie.com/explore/index.

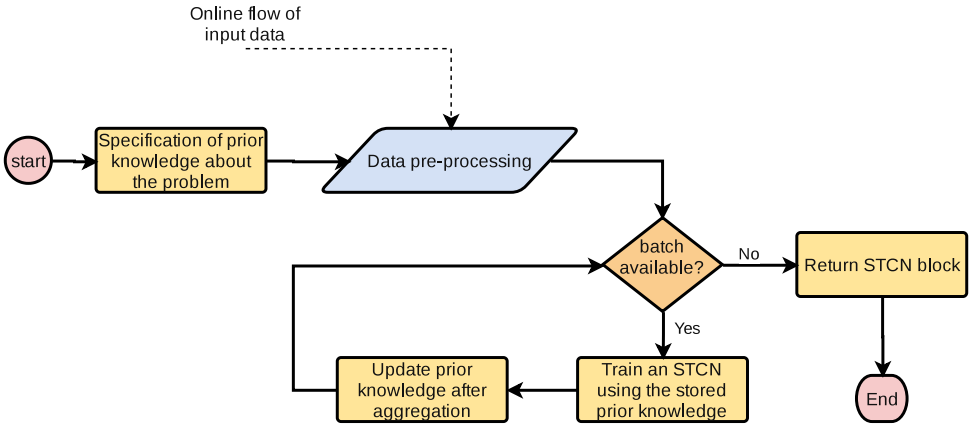


Fig. 4. The LSTCN model can be seen as a sequential collection of STCN blocks that perform iterative learning. When a new chunk of data is available, a new STCN block is trained and the prior knowledge is updated using an aggregation procedure.

Table 1
Descriptive statistics for the windmill datasets.

Dataset	*min*	*med*	*max*
1	0.0708	0.2799	0.9456
2	0.0888	0.3032	0.8848
3	0.0687	0.3014	0.9497
4	0.0835	0.3148	0.9441

We split each dataset using a hold-out approach (80% for training and 20% for testing purposes). As for the performance metric, we use the mean absolute error (MAE) in all simulations reported in this section. In addition, we report the training and test times of each forecasting model. The training time (in seconds) of each algorithm was computed by adding the time needed to train the algorithm in each time patch. Finally, we arbitrarily fix the patch size to 1024.

4.2. Recurrent online learning models

We contrast the LSTCNs' performance against four recurrent learning networks used to handle online learning settings. The models adopted for comparison are GRU, LSTM, HMM, and a fully connected Recurrent Neural Network (RNN) where the output is to be fed back to the input.

The RNN, LSTM and GRU networks were implemented using Keras v2.4.3, while HMM was implemented using the *hmmlearn* library.[2] The training of these models was adapted to online learning scenarios. In practice, this means that RNN, GRU, and LSTM were retrained on each time patch using the prior knowledge structures learned in previous learning steps. In the HMM-based forecasting model, the transition probability matrix is passed from one patch to another, and it is updated based on the new information.

In the LSTCN model, we used $L = \{6, 48, 72\}$ such that $R = L$ (hereinafter we will only refer to L) and $w = 10$. Notice that given the sampling interval of the data, six steps represent one hour while 72 steps represent half a day. We did not perform parameter tuning since the online learning setting demands fast re-training of these recurrent models when a new data chunk arrives. It would not be feasible to fine-tune the hyperparameters in each iteration since such a process is computationally demanding. Instead, we retained the default parameters reported on the corresponding Keras layers. In the HMM-based model, we used four hidden states and Gaussian emissions to generate the predictions. These parameter values were arbitrarily selected without further experimentation.

Table 2
Results for the windmill case study for $L = 6$ (1 h). LSTCN clearly outperforms the other models in both in accuracy and training time.

	Model	Training error	Training time	Test error	Test time
WT1	LSTCN	0.0672	0.0091	**0.0715**	0.0255
	RNN	0.1087	0.6477	0.1321	1.3503
	LSTM	0.1036	1.7681	0.1258	2.6994
	GRU	0.1193	1.7162	0.1396	2.4513
	HMM	0.0567	241.17	0.1267	91.08
WT2	LSTCN	0.0592	0.0091	**0.0647**	0.0261
	RNN	0.1086	0.6516	0.1209	1.3048
	LSTM	0.1035	1.7075	0.1147	2.5682
	GRU	0.1243	1.7718	0.1384	2.3741
	HMM	0.0381	333.61	0.0941	107.54
WT3	LSTCN	0.0474	0.0103	**0.0564**	0.0370
	RNN	0.1221	0.6418	0.1532	1.3171
	LSTM	0.1107	1.6932	0.1455	2.6672
	GRU	0.1219	1.6821	0.1499	2.3471
	HMM	0.0683	318.11	0.1768	118.31
WT4	LSTCN	0.0602	0.0112	**0.0666**	0.0398
	RNN	0.1069	0.6609	0.1398	1.3507
	LSTM	0.0979	1.1241	0.1275	1.4140
	GRU	0.1162	1.6661	0.1405	2.3841
	HMM	0.0534	367.05	0.1445	164.15

Table 3
Results for the windmill case study for $L = 48$ (8 h). LSTCN clearly outperforms the other models in both in accuracy and training time.

	Model	Training error	Training time	Test error	Test time
WT1	LSTCN	0.0701	0.1958	**0.0721**	0.0408
	RNN	0.5963	17.7336	0.5966	4.5753
	LSTM	0.1248	229.1311	0.1290	30.7941
	GRU	0.1641	155.0201	0.1689	13.7719
	HMM	0.0794	1432.65	0.0978	431.02
WT2	LSTCN	0.0594	0.7503	**0.0600**	0.0753
	RNN	0.5340	27.3444	0.5357	5.7755
	LSTM	0.1424	225.2245	0.1444	31.2621
	GRU	0.2317	137.3365	0.2324	15.9500
	HMM	0.0388	1586.91	0.0821	441.19
WT3	LSTCN	0.0474	0.2333	**0.0482**	0.0470
	RNN	0.4991	20.1501	0.5048	5.8768
	LSTM	0.1332	236.3217	0.1401	31.8122
	GRU	0.1887	136.6582	0.1933	16.3957
	HMM	0.0761	1462.68	0.1582	413.93
WT4	LSTCN	0.0600	0.2463	**0.0623**	0.0490
	RNN	0.3407	37.1072	0.3425	5.8727
	LSTM	0.1254	214.9554	0.1318	19.9095
	GRU	0.1634	138.3381	0.1688	16.1307
	HMM	0.0799	1971.42	0.1239	531.94

[2] https://github.com/hmmlearn/hmmlearn.

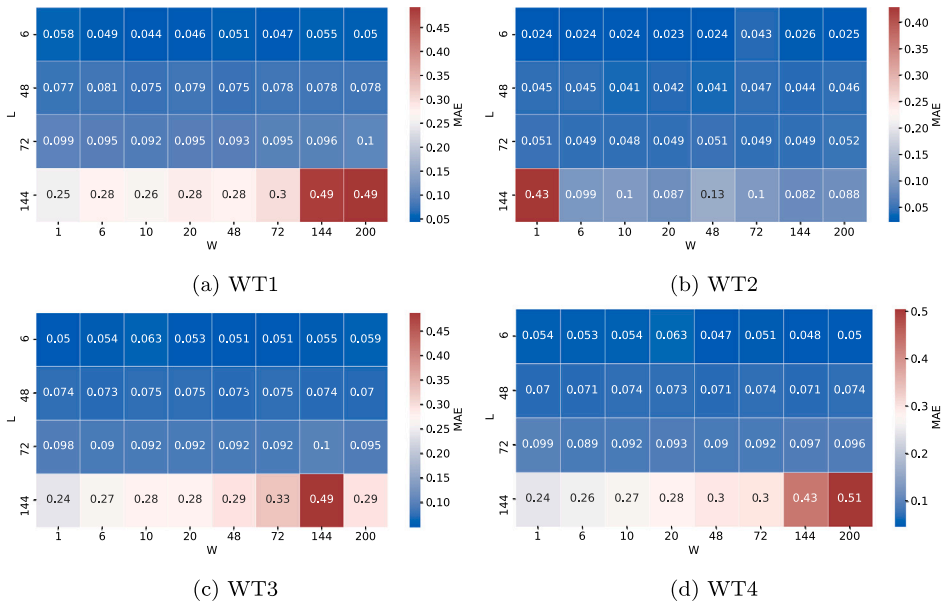


Fig. 5. MAE values obtained by the LSTCN-based model when changing the w and L parameters. As expected, expanding the prediction horizon (that is to say, increasing the number of steps ahead to be predicted) leads to performance degradation of predictions. However, the model does not seem to be especially sensitive to the w parameter, except for larger L values where the error increases as the w gets larger.

Table 4
Results for the windmill case study for $L = 72$ (12 h). LSTCN clearly outperforms the other models in both in accuracy and training time.

	Model	Training error	Training time	Test error	Test time
WT1	LSTCN	0.0706	0.3770	**0.0726**	0.0430
	RNN	0.5187	45.8973	0.5219	9.5791
	LSTM	0.1376	714.2750	0.1418	53.8956
	GRU	0.1891	892.9639	0.1927	31.4892
	HMM	0.0861	2967.08	0.1236	549.49
WT2	LSTCN	0.0597	1.0251	**0.0604**	0.0605
	RNN	0.6490	60.2035	0.6553	8.9644
	LSTM	0.1320	726.8744	0.1345	53.2367
	GRU	0.2304	806.9376	0.2316	38.6462
	HMM	0.0488	2642.24	0.0718	622.14
WT3	LSTCN	0.0476	0.5155	**0.0494**	0.0545
	RNN	0.5524	47.0321	0.5618	9.4718
	LSTM	0.1441	768.9152	0.1518	54.4804
	GRU	0.1846	787.8666	0.1909	36.3055
	HMM	0.0811	2916.17	0.1281	631.11
WT4	LSTCN	0.0605	0.5040	**0.0618**	0.0535
	RNN	0.5313	45.3868	0.5378	10.0499
	LSTM	0.1358	690.0783	0.1404	37.7908
	GRU	0.1813	809.8271	0.1895	31.4764
	HMM	0.0887	2365.61	0.1112	589.81

4.3. Results and discussion

Fig. 5 shows an analysis of the influence of w and L on the model's behavior. The parameters were varied in the discrete set $w = \{1, 6, 10, 20, 48, 72, 144\}$ and $L = \{6, 48, 72, 144\}$, and the MAE computed on the test set was used for comparative purposes. The results did not show a large difference when changing w while keeping L fixed. However, the reduction in model performance was more evident when L increases, which is usual in time series forecasting models (see Figs. 7–10).

As mentioned, the knowledge used by the first STCN is extracted from a smoothed representation of the time series data we have. Nevertheless, we can start with a zero-filled matrix if such knowledge is not available. Fig. 6 shows the MAE of the predictions in the training set of the four windmills in both settings. Starting from scratch (no knowledge about the data), the LSTCN starts predicting with a large MAE in the first time patch. As new data is received, the network updates its knowledge and reduces the prediction error. In this simulation, we used five time patches such that each STCN block is fitted on the newly received data. The LSTCN model using general knowledge of the time series (assumed as a warm-up) generates small errors from the first time patch.

Tables 2–4 show the results for $L = 6, L = 48$ and $L = 72$, respectively. More explicitly, we report the average training and test errors, and the average training and test times (in seconds). The LSTCN model obtained the lowest MAE value in all cases (the lowest average test error for each windmill is highlighted in boldface). Those results allow us to conclude that our approach is able to produce better forecasting results when compared with well-established recurrent neural networks. It should be noted, however, that such a conclusion is attached to the fact that no hyper-parameter tuning was performed in our simulations.

Another clear advantage of LSTCN over these state-of-the-art algorithms is the reduced training and test times. Re-training the model quickly when a new piece of data arrives while retaining the knowledge we have learned so far is a key challenge in online learning settings. Recurrent neural models such as RNN, LSTM, GRU use a backpropagation-based learning algorithm to compute the weights regulating the network behavior. The algorithm needs to iterate multiple times over the data with limited vectorization possibilities.

Overall, there is a trade-off between accuracy and training time when it comes to the batch size. The smaller the batch size in the backpropagation learning, the more accurate the predictions are expected to be. However, smaller batch sizes make the training process slower. Another issue with gradient-based optimization methods is that they usually operate in a stochastic fashion, thus making them quite sensitive to the initial conditions. Notice that HMM also requires several iterations to build the probability transition matrix.

To alleviate the problem caused by larger prediction horizons, we could increase the batch size in our model (or decrease the batch size of other recurrent models used for comparison purposes). In that way, we will have more data in each training process, which will likely lead to models with improved predictive power. Alternatively, we could adopt an incremental learning approach to reuse data concerning previous time patches as defined by a given window parameter. However, we should be aware that many online learning problems operate on volatile data that is just available for a short period.

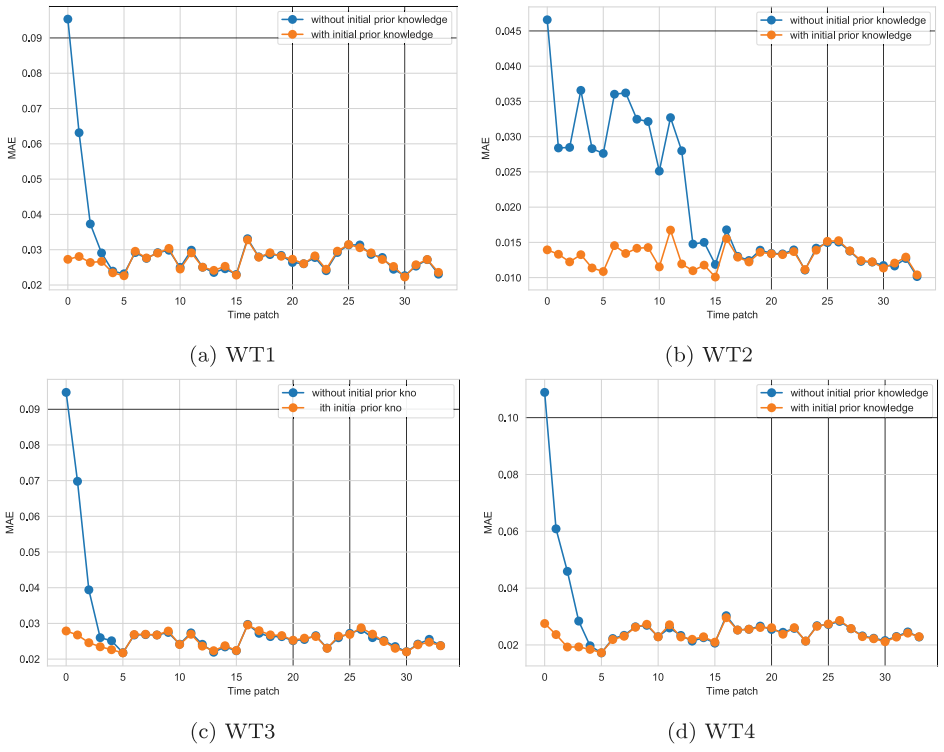


Fig. 6. MAE values obtained by the LSTCN-based model on the four windmill datasets with and without using initial prior knowledge. It can be noticed that the model needs to process more time patches to reduce the error when the model is initialized with a random weight matrix. If this knowledge is not available, the network will still produce good results provided it performs enough iterations.

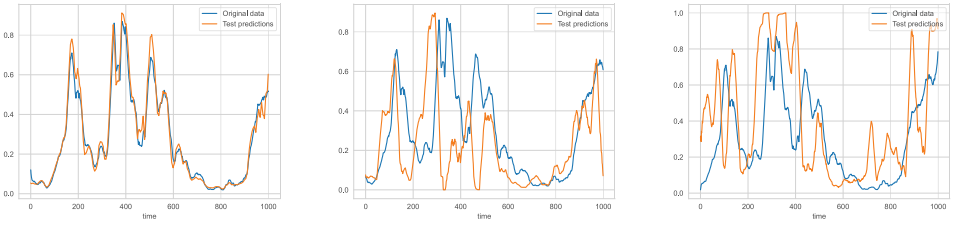


Fig. 7. Moving average power predictions ($w = 24$) for the first windmill with (a) $L = 6$, (b) $L = 48$ and (c) $L = 72$.

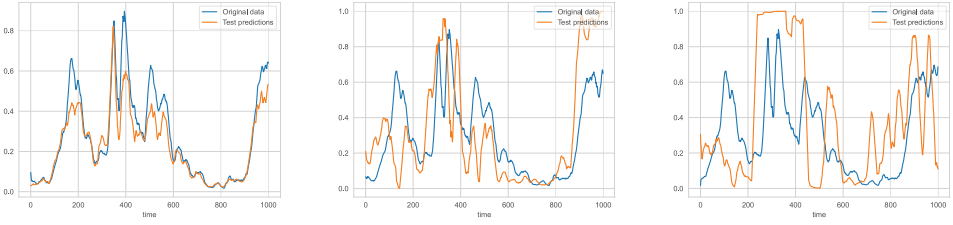
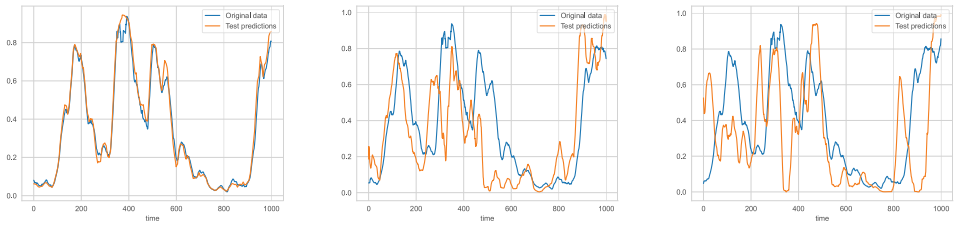


Fig. 8. Moving average power predictions ($w = 24$) for the second windmill with (a) $L = 6$, (b) $L = 48$ and (c) $L = 72$.

5. Concluding remarks

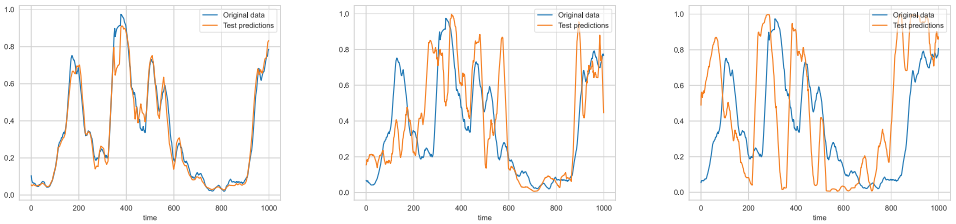
In this paper, we investigated the performance of Long Short-term Cognitive Networks to forecast windmill time series in online setting scenarios. This brand-new recurrent model system consists of a sequence of Short-term Cognitive Network blocks. Each of these blocks is trained with the available data at that moment in time such that the learned knowledge is propagated to the next blocks. Therefore, the network is able to adjust its knowledge to new information, which makes this model suitable for online settings since we retain the knowledge learned from previous learning processes.

The experiments conducted using four windmill datasets reported that our approach outperforms other state-of-the-art recurrent neural networks in terms of MAE. In addition, the proposed LSTCN-based model is significantly faster than these recurrent models when it comes



(a) $L = 6$ (b) $L = 48$ (c) $L = 72$

Fig. 9. Moving average power predictions ($w = 24$) for the third windmill with (a) $L = 6$, (b) $L = 48$ and (c) $L = 72$.



(a) $L = 6$ (b) $L = 48$ (c) $L = 72$

Fig. 10. Moving average power predictions ($w = 24$) for the fourth windmill with (a) $L = 6$, (b) $L = 48$ and (c) $L = 72$.

to both training and test times. Such a feature is of paramount relevance when designing forecasting models operating in online learning modes. Regrettably, the overall performance of all forecasting models deteriorated when increasing the number of steps ahead to be predicted. While this result is not surprising, further efforts are needed to build forecasting models with better scalability properties as defined by the prediction horizon.

Before concluding our paper, it is worth mentioning that the proposed architecture for online time series forecasting is not restricted to windmill data. Instead, the architecture can be applied to any univariate or multivariate time series provided that the proper pre-processing steps are conducted.

CRediT authorship contribution statement

Alejandro Morales-Hernández: Methodology, Software, Validation, Writing – original draft, Visualization. **Gonzalo Nápoles:** Conceptualization, Methodology, Software, Validation, Writing – original draft, Visualization, Supervision. **Agnieszka Jastrzebska:** Validation, Writing – original draft. **Yamisleydi Salgueiro:** Validation, Writing – review & editing. **Koen Vanhoof:** Supervision, Writing – review & editing.

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.

Acknowledgments

Alejandro Morales and Koen Vanhoof from Hasselt University would like to thank the support received by the Flanders AI Research Program, as well as other partners involved in this project. Agnieszka Jastrzebska's contribution was founded by the National Science Centre, grant No. 2019/35/D/HS4/01594, decision no. DEC-2019/35/D/HS4/01594. Y. Salgueiro would like to acknowledge the support provided by the National Center for Artificial Intelligence CENIA FB210017, Basal ANID and the super-computing infrastructure of the NLHPC (ECM-02). The authors would like to thank Isel Grau from the Eindhoven University of Technology for revising the paper.

References

Ahmed, S. D., Al-Ismail, F. S., Shafiullah, M., Al-Sulaiman, F. A., & El-Amin, I. M. (2020). Grid integration challenges of wind energy: A review. *IEEE Access*, *8*, 10857–10878. http://dx.doi.org/10.1109/ACCESS.2020.2964896.

Ahmed, A., & Khalid, M. (2019). A review on the selected applications of forecasting models in renewable power systems. *Renewable and Sustainable Energy Reviews*, *100*, 9–21. http://dx.doi.org/10.1016/j.rser.2018.09.046.

Bhaumik, D., Crommelin, D., Kapodistria, S., & Zwart, B. (2019). Hidden markov models for wind farm power output. *IEEE Transactions on Sustainable Energy*, *10*, 533–539. http://dx.doi.org/10.1109/TSTE.2018.2834475.

Cao, L., Qian, Z., Zareipour, H., Huang, Z., & Zhang, F. (2019). Fault diagnosis of wind turbine gearbox based on deep bi-directional long short-term memory under time-varying non-stationary operating conditions. *IEEE Access*, *7*, 155219–155228. http://dx.doi.org/10.1109/ISIE.2019.8781108.

Cao, L., Zhang, J., Wang, J., & Qian, Z. (2019). Intelligent fault diagnosis of wind turbine gearbox based on long short-term memory networks. In *2019 IEEE 28th international symposium on industrial electronics* (pp. 890–895). http://dx.doi.org/10.1109/ACCESS.2019.2947501.

Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, *8*(6085), http://dx.doi.org/10.1038/s41598-018-24271-9.

Chen, H., Liu, H., Chu, X., Liu, Q., & Xue, D. (2021). Anomaly detection and critical SCADA parameters identification for wind turbines based on LSTM-AE neural network. *Renewable Energy*, *172*, 829–840. http://dx.doi.org/10.1016/j.renene.2021.03.078.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing* (pp. 1724–1734). http://dx.doi.org/10.3115/v1/D14-1179.

Cui, Y., Bangalore, P., & Bertling Tjernberg, L. (2021). A fault detection framework using recurrent neural networks for condition monitoring of wind turbines. *Wind Energy*, *24*(11), 1249–1262. http://dx.doi.org/10.1002/we.2628.

Delgado, I., & Fahim, M. (2021). Wind turbine data analysis and LSTM-based prediction in SCADA system. *Energies*, *14*, http://dx.doi.org/10.3390/en14010125.

Du, M., Yi, J., Mazidi, P., Cheng, L., & Guo, J. (2017). A parameter selection method for wind turbine health management through SCADA data. *Energies*, *10*, http://dx.doi.org/10.3390/en10020253.

Feng, B., Zhang, D., Si, Y., Tian, X., & Qian, P. (2019). A condition monitoring method of wind turbines based on long short-term memory neural network. In *2019 25th international conference on automation and computing* (pp. 1–4). http://dx.doi.org/10.23919/IConAC.2019.8895037.

Foley, A. M., Leahy, P. G., Marvuglia, A., & McKeogh, E. J. (2012). Current methods and advances in forecasting of wind power generation. *Renewable Energy*, *37*, 1–8. http://dx.doi.org/10.1016/j.renene.2011.05.033.

Gers, F. A., Schraudolph, N. N., & Schmidhuber, J. (2002). Learning precise timing with lstm recurrent networks. *Journal of Machine Learning Research*, *3*, 115–143.

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.

Janssens, O., Noppe, N., Devriendt, C., de Walle, R. V., & Hoecke, S. V. (2016). Data-driven multivariate power curve modeling of offshore wind turbines. *Engineering Applications of Artificial Intelligence*, *55*, 331–338. http://dx.doi.org/10.1016/j.engappai.2016.08.003.

Jorgensen, K. L., & Shaker, H. R. (2020). Wind power forecasting using machine learning: State of the art trends and challenges. In *2020 8th international conference on smart energy grid engineering* (pp. 44–50). http://dx.doi.org/10.1109/SEGE49949.2020.9181870.

Kong, W., Dong, Z. Y., Jia, Y., Hill, D. J., Xu, Y., & Zhang, Y. (2019). Short-term residential load forecasting based on LSTM recurrent neural network. *IEEE Transactions on Smart Grid*, *10*, 841–851. http://dx.doi.org/10.1109/TSG.2017.2753802.

Kong, Z., Tang, B., Deng, L., Liu, W., & Han, Y. (2020). Condition monitoring of wind turbines based on spatio-temporal fusion of SCADA data by convolutional neural networks and gated recurrent units. *Renewable Energy*, *146*, 760–768. http://dx.doi.org/10.1016/j.renene.2019.07.033.

Kramti, S. E., Ben Ali, J., Saidi, L., Sayadi, M., & Bechhoefer, E. (2018). Direct wind turbine drivetrain prognosis approach using elman neural network. In *2018 5th international conference on control, decision and information technologies* (pp. 859–864). http://dx.doi.org/10.1109/CoDIT.2018.8394926.

Lei, J., Liu, C., & Jiang, D. (2019). Fault diagnosis of wind turbine based on long short-term memory networks. *Renewable Energy*, *133*, 422–432. http://dx.doi.org/10.1016/j.renene.2018.10.031.

Li, T., Tang, J., Jiang, F., Xu, X., Li, C., Bai, J., & Ding, T. (2019). Fill missing data for wind farms using long short-term memory based recurrent neural network. In *2019 IEEE 3rd international electrical and energy conference* (pp. 705–709). http://dx.doi.org/10.1109/CIEEC47146.2019.CIEEC-2019284.

Lin, C.-H. (2013). Recurrent modified elman neural network control of PM synchronous generator system using wind turbine emulator of pm synchronous servo motor drive. *International of Electrical Power & Energy Systems*, *52*, 143–160. http://dx.doi.org/10.1016/j.ijepes.2013.03.021.

Lin, C.-H. (2016). Wind turbine driving a PM synchronous generator using novel recurrent Chebyshev neural network control with the ideal learning rate. *Energies*, *9*, http://dx.doi.org/10.3390/en9060441.

Liu, B., Zhao, S., Yu, X., Zhang, L., & Wang, Q. (2020). A novel deep learning approach for wind power forecasting based on WD-LSTM model. *Energies*, *13*, http://dx.doi.org/10.3390/en13184964.

López, E., Valle, C., Allende-Cid, H., & Allende, H. (2020). Comparison of recurrent neural networks for wind power forecasting. In *Pattern recognition* (pp. 25–34). http://dx.doi.org/10.1007/978-3-030-49076-8_3.

Losing, V., Hammer, B., & Wersing, H. (2018). Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, *275*, 1261–1274. http://dx.doi.org/10.1016/j.neucom.2017.06.084.

Manero, J., Bejar, J., & Cortes, U. (2018). Wind energy forecasting with neural networks: A literature review. *Computing and Systems*, *22*, 1085–1098. http://dx.doi.org/10.13053/cys-22-4-3081.

Mishra, S., Bordin, C., Taharaguchi, K., & Palu, I. (2020). Comparison of deep learning models for multivariate prediction of time series wind power generation and temperature. *Energy Reports*, *6*, 273–286. http://dx.doi.org/10.1016/j.egyr.2019.11.009.

Nápoles, G., Grau, I., Jastrzebska, A., & Salgueiro, Y. (2021). Long short-term cognitive networks. arXiv preprint arXiv:2106.16233.

Nápoles, G., Vanhoenshoven, F., & Vanhoof, K. (2019). Short-term cognitive networks, flexible reasoning and nonsynaptic learning. *Neural Networks*, *115*, 72–81. http://dx.doi.org/10.1016/j.neunet.2019.03.012.

Niu, Z., Yu, Z., Tang, W., Wu, Q., & Reformat, M. (2020). Wind power forecasting using attention-based gated recurrent unit network. *Energy*, *196*, Article 117081. http://dx.doi.org/10.1016/j.energy.2020.117081.

Qian, P., Tian, X., Kanfoud, J., Lee, J. L. Y., & Gan, T.-H. (2019). A novel condition monitoring method of wind turbines based on long short-term memory neural network. *Energies*, *12*, http://dx.doi.org/10.3390/en12183411.

Qu, K., Si, G., Sun, X., Lian, W., Huang, Y., & Li, P. (2021). Time series simulation for multiple wind farms based on hmms and regular vine copulas. *Journal of Renewable and Sustainable Energy*, *13*, Article 023311. http://dx.doi.org/10.1063/5.0033313.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*, 257–286. http://dx.doi.org/10.1109/5.18626.

Riganti-Fulginei, F., Sun, Z., & Sun, H. (2018). Health status assessment for wind turbine with recurrent neural networks. *Mathematical Problems in Engineering*, *2018*, Article 6972481. http://dx.doi.org/10.1155/2018/6972481.

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, *404*, Article 132306. http://dx.doi.org/10.1016/j.physd.2019.132306.

Sinsel, S. R., Riemke, R. L., & Hoffmann, V. H. (2020). Challenges and solution technologies for the integration of variable renewable energy sources—a review. *Renewable Energy*, *145*, 2271–2285. http://dx.doi.org/10.1016/j.renene.2019.06.147.

Strobelt, H., Gehrmann, S., Pfister, H., & Rush, A. M. (2018). Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, *24*, 667–676. http://dx.doi.org/10.1109/TVCG.2017.2744158.

Wang, H., Lei, Z., Zhang, X., Zhou, B., & Peng, J. (2019). A review of deep learning for renewable energy forecasting. *Energy Conversion and Management*, *198*, Article 111799. http://dx.doi.org/10.1016/j.enconman.2019.111799.

Wang, Y., Xie, D., Wang, X., & Zhang, Y. (2018). Prediction of wind turbine-grid interaction based on a principal component analysis-long short term memory model. *Energies*, *11*, http://dx.doi.org/10.3390/en11113221.

Weerakody, P. B., Wong, K. W., Wang, G., & Ela, W. (2021). A review of irregular time series data handling with gated recurrent neural networks. *Neurocomputing*, *441*, 161–178. http://dx.doi.org/10.1016/j.neucom.2021.02.046.

Xiang, L., Wang, P., Yang, X., Hu, A., & Su, H. (2021). Fault detection of wind turbine based on SCADA data analysis using CNN and LSTM with attention mechanism. *Measurement*, *175*, Article 109094. http://dx.doi.org/10.1016/j.measurement.2021.109094.

Xue, X., Xie, Y., Zhao, J., Qiang, B., Mi, L., Tang, C., & Li, L. (2021). Attention mechanism-based CNN-LSTM model for wind turbine fault prediction using SSN ontology annotation. *Wireless Communications and Mobile Computing*, *2021*, Article 6627588. http://dx.doi.org/10.1155/2021/6627588.

Yin, A., Yan, Y., Zhang, Z., Li, C., & Sanchez, R.-V. (2020). Fault diagnosis of wind turbine gearbox based on the optimized lstm neural network with cosine loss. *Sensors*, *20*, http://dx.doi.org/10.3390/s20082339.

Zhang, X.-M., Han, Q.-L., Ge, X., & Ding, D. (2018). An overview of recent developments in lyapunov-krasovskii functionals and stability criteria for recurrent neural networks with time-varying delays. *Neurocomputing*, *313*, 392–401. http://dx.doi.org/10.1016/j.neucom.2018.06.038.

Zhang, F., Wen, Z., Liu, D., Jiao, J., Wan, H., & Zeng, B. (2020). Calculation and analysis of wind turbine health monitoring indicators based on the relationships with scada data. *Applied Sciences*, *10*, http://dx.doi.org/10.3390/app10010410.

Zhen, H., Niu, D., Yu, M., Wang, K., Liang, Y., & Xu, X. (2020). A hybrid deep learning model and comparison for wind power forecasting considering temporal-spatial feature extraction. *Sustainability*, *12*, http://dx.doi.org/10.3390/su12229490.