Çukurova Üniversitesi Mühendislik Fakültesi Dergisi, 39(2), ss. 287-300, Haziran 2024
Cukurova University Journal of the Faculty of Engineering, 39(2), pp. 287-300, June 2024

Wind Speed Prediction Using Deep Recurrent Neural Networks and Farm Platform Features for One-Hour-Ahead Forecast

Emre ÖZBİLGE[1] ORCID *0000-0002-2295-752X*
Yönal KIRSAL[*2] ORCID *0000-0001-7031-1339*

[1]Cyprus International University, Faculty of Engineering, Software Engineering Department, TRNC
[2]European University of Lefke, Faculty of Engineering, Electrical and Electronics Engineering Department, TRNC

Geliş tarihi: 11.01.2024 Kabul tarihi: 27.06.2024

Atıf şekli/ How to cite: ÖZBİLGE, E., KIRSAL, Y., (2024). Wind Speed Prediction Using Deep Recurrent Neural Networks and Farm Platform Features for One-Hour-Ahead Forecast. Cukurova University, Journal of the Faculty of Engineering, 39(2), 287-300.

Abstract

This paper proposes a deep recurrent neural network (DRNN) approach to model the one-hour-ahead wind speed forecasting by using various meteorological sensory data from the North Wyke farm platform (NWFP). To refine model input, mutual information analysis is applied to eliminate irrelevant sensory data. The DRNN architecture employs three recurrent layers Long-Short Term Memory (LSTM), Gated Recurrent Unit (GRU), and simple Recurrent Neural Network (RNN) to capture temporal relationships. The proposed networks are tested using real-life, one-year data from the NWFP. The results showed a strong correlation between the actual and predicted wind speed for LSTM, GRU, and RNN layers-based DRNN, however, simple RNN slightly outperformed the other two recurrent layers. The distribution of the network errors over the year is also analyzed. Although the observed meteorological data between the years was from different distributions, the proposed network generalized well even though these data were altered due to global warming.

Keywords: Wind speed prediction, Deep recurrent neural network, Intelligent systems, Time-series prediction, Feature selection

Derin Tekrarlayan Sinir Ağları ve Çiftlik Platformu Özellikleri Kullanılarak Bir Saat Önceden Rüzgâr Hızı Tahmini

Öz

Bu makale, Kuzey Wyke çiftliği platformundan (NWFP) çeşitli meteorolojik veriler kullanarak bir saat öncesine yönelik rüzgâr hızı tahmini modellemek için derin tekrarlı sinir ağı (DRNN) yaklaşımını önermektedir. Model girişini iyileştirmek için karşılıklı bilgi analizi kullanılarak ilgisi olmayan veriler elenmiştir. DRNN mimarisi, zamansal ilişkileri yakalamak üzere üç tekrarlı katmanı içerir: Uzun Kısa Vadeli Bellek (LSTM), Kapılı Tekrarlı Birim (GRU) ve basit Tekrarlı Sinir Ağı (RNN). Önerilen ağlar,

[*]Sorumlu yazar (Corresponding Author): Yönal KIRSAL, *ykirsal@eul.edu.tr*

NWFP'den gerçek zamanlı, bir yıllık veri kullanılarak test edilmiştir. Sonuçlar, LSTM, GRU ve basit RNN katmanları temelli DRNN için gerçek ve tahmin edilen rüzgâr hızı arasında güçlü bir korelasyon olduğunu göstermiştir; ancak basit RNN, diğer iki tekrarlı katmandan biraz daha iyi performans sergilemiştir. Ayrıca, ağ hatalarının yıl boyunca dağılımı analiz edilmiştir. Gözlemlenen meteorolojik verilerin yıllar arasında farklı dağılımlardan olmasına rağmen, önerilen ağ, bu veriler küresel ısınma nedeniyle değişmiş olsa bile iyi genelleme yapmıştır.

Anahtar Kelimeler: Rüzgâr hızı tahmini, Derin tekrarlayan sinir ağı, Akıllı sistemler, Zaman serisi tahmini, Özellik seçimi

1. INTRODUCTION

Analysis of meteorological variables and weather forecasts is important for clean and renewable energy sources. Wind energy is a significant part of renewable energy and the most promising one among renewable energy sources globally [1]. The prediction of wind speed is an essential feature in terms of wind power generation, agricultural areas, ship route planning, etc. In addition, predicting wind speed is a very important parameter for estimating the energy expected to be produced from wind turbines in the short and long terms. Based on these estimation values, the profitability of power generation plants can also be calculated. Moreover, the agriculture farm platform can be organized according to whether it is profitable to invest in wind energy in a specific region. The accuracy of short-and long-term wind power generation is also of great importance in balancing electricity generation using different resources [2, 3].

Continuous and strong wind speeds are required for the uninterrupted and high-quality electricity generation of wind turbines. However, the chaotic nature and dynamic uncertainty of the wind pose a major obstacle to wind speed prediction. Despite this chaotic structure and uncertainty, there are many methods developed for making predictions in the literature. Wind speed prediction tools are widely categorized into the physics-based model, the statistical-based model, and the hybrid prediction model [4]. The physical-based models require a firm theoretical background, many equations, and a high computational cost. Hence, the physical-based models are widely used in practice [5]. On the other hand, the statistical-based models perform better than the physical-based models in wind speed forecasting [6]. However, researchers develop statistical-based models for evaluating time series to minimize the error in estimation methods, and they also work on artificial intelligence (AI) methods for prediction. The hybrid prediction model is a combination of both adaptive structure and AI methods.

Hybrid methods achieve better accuracy and better wind prediction results compared to other individual models [7]. AI approaches are the most commonly used methods to predict wind speed based on various data in the literature [8, 9]. The main reason is that these methods can adapt themselves rapidly and accurately to changing trends within the datasets. In addition, AI approaches obtain high precision and better performance output measurements. They also produce algorithms based on input data rather than using a generalized model. Thus, to achieve better accuracy and good wind prediction results, AI models should be considered for such analysis. The artificial neural network (ANN) [10], support vector machines (SVM) [11], fuzzy logic [12], the classification algorithms of random forest (RF) [13], extreme learning machine (ELM) [14], and deep learning architectures of long-short term memory networks (LSTM) [15] are commonly used AI approaches in the literature. Recently, due to the uncertainty and complex structure of the wind speed, deep learning has gained more interest in wind speed prediction due to its imposing features, such as handling big data, solving complex systems, learning the feature hierarchy on its own, avoiding data overfitting problems, and obtaining successful results from unstructured data, etc. On the other hand, it is most suitable for real-world applications. The convolutional neural network (CNN), deep recurrent neural network (DRNN), deep belief network (DBF), gated recurrent unit neural

networks (GRUNNs), ELM, and LSTM are widely used deep learning algorithms rather than the traditional AI approaches [16].

This paper proposes the wind speed prediction model based on a DRNN approach for the first time regarding meteorological input variables from dedicated meteorological sensors from NWFP. The proposed network has been constituted with a five-layered deep architecture that contains convolutional, recurrent, and fully connected dense layers. This type of network is capable of learning the temporal and sequential relationship between the inputs and outputs of the system. Thus, it is capable of predicting future expected wind features in advance for the corresponding regions. Global warming has changed the weather over the past years; hence, all other meteorological features are also believed to have changed, such as precipitation, temperature, humidity, and so on. Therefore, the proposed network does not only receive the most relevant features, that is, the wind features. All other meteorological features were also presented to the network because there was always a causal relationship between these features. The main contributions of this study lie in the following aspects:

1. Relevancy analysis of the sensor readings and the derived features are analyzed and their contributions to the wind speed are also investigated.
2. A DRNN approach is proposed, considering convolutional, recurrent, and fully connected dense layers for the wind speed prediction of the collected data from NWFP.
3. Different recurrent layers such as LSTM, GRU, and Simple RNN are used on the DRNN's architecture, and their performances are compared.

The remainder of this paper is organized as follows: Section 2 presents the other studies about wind speed predictions; Section 3 provides the proposed prediction model and experimental methods; Section 4 presents experimental results and discussions; and finally, some conclusions are drawn in Section 5.

2. RELATED STUDIES

A wind power short-term prediction based on LSTM and discrete wavelet transform (WT) was proposed in [17]. The results showed that the prediction accuracy had been improved by the proposed method. In [18] a novel wind speed multistep prediction model was proposed by combining the variational mode decomposition (VMD), singular spectrum analysis (SSA), LSTM, and ELM. The results showed that the proposed model has the best multi-step prediction performance. In addition, it was also more effective and robust in extracting trend information. A GRUNN-based wind speed error correction model for short-term wind power forecasting was presented in [19]. The feature of wind speed was analyzed, and the standard deviation of wind speed error was also extracted as weights for the numerical weather prediction (NWP) wind speed time series.

The proposed prediction model was compared with existing models such as SVM and ANN. The results show that the proposed model gives better performance results than the existing models. In addition, the GRUNNs-based data-driven approach was also proposed in [20] for wind power forecasting. The proposed model was compared and contrasted with the LSTM algorithm. The results show that the GRUNNs outperformed the LSTM in terms of a faster training process and less sensitivity to noise. In [21] a novel hybrid forecasting system was proposed that is formed by effective data decomposition techniques, DRNN, and error decomposition correction methods. Four different wind farm datasets in China were performed and verified by the proposed model. The results showed that the proposed model obtained a highly accurate wind speed prediction compared to simple and traditional models.

In [22] a new hybrid deep learning model was also proposed for short-term wind speed forecasting. An improved complementary ensemble empirical mode decomposition with adaptive noise (ICEEMDAN) and autoregressive integrated moving average (ARIMA) are combined with the

ELM technique. The experiments focused on pre-processing and post-processing time series data. In [23] a hybrid model based on the crow search algorithm (CSA), WT, feature selection (FS) based on entropy, mutual information (MI), and deep learning time series prediction based on LSTM is proposed for short-term wind speed forecasting. The results showed that the proposed method can outperform the most basic existing wind speed forecasting methods. A deep learning-based approach was proposed in [16] to characterize the probability density function (PDF) of the wind for short-term wind speed forecasting. The proposed model considers CNN and GRU to learn features of wind speed time series. Two actual data sets are used from England and Iran in the analysis. More accurate results have been obtained compared to other deep mixture approaches.

In [24] ELM and LSTM methods have been used to obtain VMD and SSA to complete the prediction. On the other hand, two hybrid models were proposed in [25] where one was formed by the LSTM and DNN and the other was combined with GRU networks and DNN. Moreover, three hybrid models were also proposed in [26] to improve the forecasting accuracy for wind speed. The WT is first mapped into the original wind speed history into several subseries. Then, for the low-frequency sub-series, the RNNs were used to extract the deeper features and involved in suitable machine learning methods for predicting, while others were still predicted by the normal methods. The results show that deep learning models outperform traditional approaches.

3. METHODOLOGY

3.1. Datasets

The NWFP is a globally unique but national UK real-world farming platform established in 2010 [27]. The NWFP is located at North Wyke in the southwest of England to understand grassland management at the systems level [28]. The NWFP real-time data, as well as the experimental work, are available to the public. In other words, the data provided by the NWFP is open-access and free to download. More information about the NWFP can be found in [27,28]. The experiment data set was used from August 12, 2017, to August 24, 2020. There are 106,944 data points from 1,114 days of data obtained on the farm platform in the given specified period. The dedicated meteorological equipment and sensors were installed to record six different meteorological features as precipitation (mm), air temperature (◦C), relative humidity (%), wind speed (m/s), wind direction (◦) and solar radiation (W/m2) at 15-min intervals. *p*, *at*, *rh*, *ws*, *wd* and *sr* are the abbreviations used for the precipitation, air temperature, relative humidity, wind speed, wind direction, and solar radiation, respectively, in this paper.

3.2. Feature Engineering

To produce more distinguishability between the sensory data which are received at different dates and times, several input features are generated. First of all, the wind direction received from the sensor is in the range of [0◦,360◦]. Wind direction is cyclical data so this can cause a problem while training the network. Any two values close to *0*◦ and 360◦ must be close to each other as well. Otherwise, the large difference makes a large network's update by gradient descent algorithm during training even if these two degrees are close. Due to this reason, wind direction and speed are transformed to the wind vector (*wx*, *wy*) as given in Equations (1) and (2). Here, *wd* data has been logged in the unit of degree so that by multiplying this value with π/180 that is simply converted to radian.

$$wx = ws * cos\left(\frac{wd*\pi}{180}\right) \quad (1)$$

$$wy = ws * sin\left(\frac{wd*\pi}{180}\right) \quad (2)$$

Each sensory data is received at a specific date and time, therefore, date and time information can also be added to the network training alongside the meteorological sensory data. In this way, the wind

data could be associated with the time of the wind data that occurred. Thus, this association provides additional information to make the wind data more distinguishable along the year by the neural network. However, timestamped data (*e.g.* minutes, hours, seconds, and so on) are all cyclical and repeat in a certain period. In cyclical data, the quantitative between each consecutive data is small depending on the step size, however, this is not true between the first and last values of the cyclical data. To overcome this type of data, time-specific information is transformed into two-dimensional using cosine and sine functions. To do this, the date and time data must be transformed to the total second first. Therefore, each timestamp is represented as the number no seconds that have been passed since the 1st of January, 1970, i.e. the beginning of Unix time. Afterwards, two-dimensional (2D) features are generated by using the date and time information of each perceived sensory reading. Two different 2D features, these are time of day (*ds*, *dc*) and time of year (*ys*, *yc*), are generated as given in Equations (3) to (6),

$$ds = sin\left(\frac{t*2*\pi}{d}\right) \tag{3}$$

$$dc = cos\left(\frac{t*2*\pi}{d}\right) \tag{4}$$

$$ys = sin\left(\frac{t*2*\pi}{y}\right) \tag{5}$$

$$yc = cos\left(\frac{t*2*\pi}{y}\right) \tag{6}$$

where $d = 24 \times 60 \times 60$ and $y = 365.2425 \times d$, these are total seconds in a day and a year respectively and t indicates Unix time.

3.3. Feature Selection

After having obtained newly generated features (these are *wx*, *wy*, *ds*, *dc*, *ys* and *yc*) alongside the available sensory data, it is important to analyze all features whether or not they carry relevant information that contributes to the model of wind.

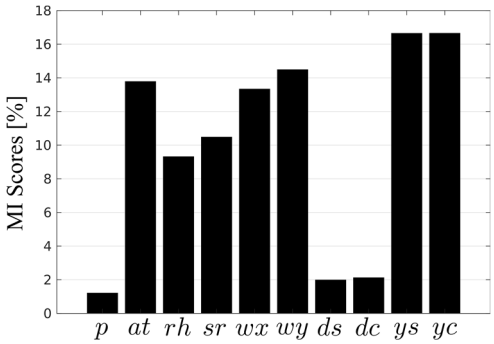

Figure 1. Mutual information analysis between the input features and the target outputs

Therefore, MI analysis is performed between all input features and the target wind speed (*ws*). Figure 1 shows the MI outcome of each input feature against each target output. The highest MI value implies the most relevant input feature to model the target wind speed. As seen in the figures, the highest MI values are obtained for the time of year features *ys* and *yc*. These results are expected because more or less meteorological data are similarly repeated every year with some deviation. Surely, this deviation increases with global warming which changes the climate. On the other hand, time of day and precipitation features have less or about 2% relevancy therefore these features are removed from the inputs of the network which do not contribute to modelling the future wind data. Figure 2 shows the final input-output configuration of the DRNN model, where the network receives the current input features and predicts a one-hour-ahead wind speed.

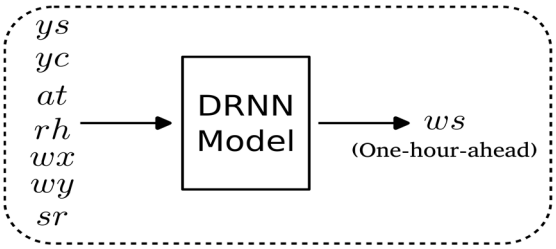

Figure 2. Inputs and outputs of the DRNN model after MI analysis.

Furthermore, the input data of the DRNN model were normalized by subtracting the mean and dividing by the standard deviation of each feature

before presenting them to the model. This method is also known as data standardization and is given in Equation (7). The target output data (ws) were kept raw during the model training.

$$\acute{x_i} = \frac{x_i - mean(x_i)}{stdev(x_i)} \quad (7)$$

where $\acute{x_i}$ and x_i indicate the scaled and raw feature i, respectively.

3.4. Deep Recurrent Neural Network

A deep recurrent neural network consists of multiple different types of layers which are convolutional, recurrent, and fully connected layers. To capture the temporal and sequential information from the data presented to the network, it is important to use recurrent connections-based layers such as LSTM, GRU, or a simple RNN layer on the network. Such a network does not only work on an input space but also on an internal state space which enables one to learn the representation of temporally or sequentially long-term dependencies over unspecified intervals [29]. Otherwise, it is not possible to learn a temporal model to predict the future values of the desired wind speed.

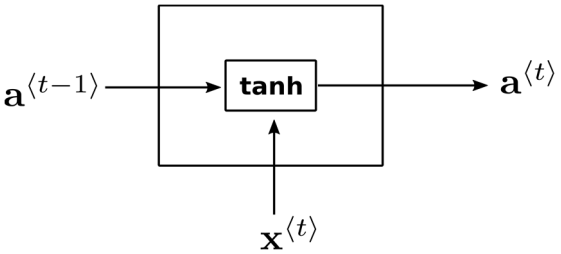

Figure 3. The diagram of a simple RNN cell at time t.

1) Simple RNN Cell: Simple RNN (Figure 3) was introduced by Elman [30] where the activation of hidden layer $a^{\langle t\rangle}$ is computed by feeding the hidden layer with the current inputs $x^{\langle t\rangle}$ and with the previously hidden activation values $a^{\langle t-1\rangle}$as given in Equation (8), this is also known as recurrent connections that create a short-term memory to remember the sequence of data presented to the network. To learn the long-term dependency, the simple RNN cells are connected sequentially to each other (see Figure 5) and the inputs of the corresponding cell become the output of the previous RNN cell unit, this is also known as unfolding RNN. However, a large number of unfolded simple RNN units can cause vanishing gradient problems [31].

$$a^{\langle t\rangle} = \sigma_h\left(\Theta_a\left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_a\right) \quad (8)$$

where σ_h indicates the hyperbolic tangent function, Θ is the connection weight matrix, and b is the bias vector.

2) LSTM Cell: The LSTM contains a number of the connected LSTM cell which has feedback connections to present a memory behaviour that remembers the values over arbitrary time intervals. Another advantage of using LSTM cells is to overcome the vanishing gradient problem when adding many layers to the deep network [32].

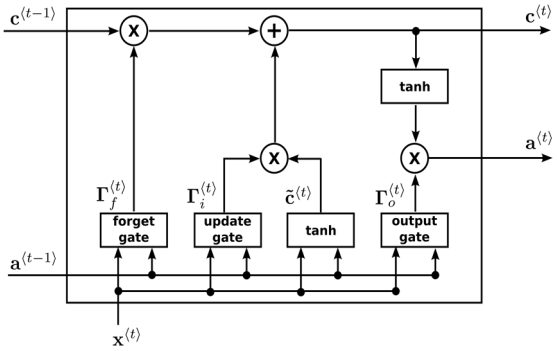

Figure 4. The diagram of an LSTM cell at time *t*.

LSTM cell is composed of a forget gate, update gate, cell state, output gates, and the hidden state as shown in Figure 4. Forget gate $\Gamma_f^{\langle t\rangle}$ enables the LSTM cell to reset the memory of the previous cell state $c^{\langle t-1\rangle}$. The gate receives inputs from the previous hidden state $a^{\langle t-1\rangle}$ and the current timestep's input $x^{\langle t\rangle}$ and a sigmoid function are used to keep the output of the gate in the range of [0,1], so that whenever the gate's value closes to 0, the LSTM cell forgets the majority of the previously stored cell state, otherwise the stored cell state is remembered for the next timestep when the gate's output closes to one.

Furthermore, the LSTM cell computes the candidate value $\tilde{c}^{\langle t \rangle}$ which contains information from the current timestep that may be stored in the current cell state $c^{\langle t \rangle}$. Then, the update gate $\Gamma_f^{\langle t \rangle}$ decides which part of the candidate could be passed to the cell state. The sigmoid function is used in the update gate that clamps the gate's output in the range of [0,1]. When the output of the update gate closes to one, the majority of the candidate's value is passed to the cell state, otherwise, the value of the candidate is not passed to the cell state when the output closes to zero. Therefore, the new value of cell state $c^{\langle t \rangle}$ becomes the combination of the previous cell state and the candidate value. Finally, the current hidden state $a^{\langle t \rangle}$ is computed by using the output of output gate $\Gamma_0^{\langle t \rangle}$. The equations of the LSTM cell are given in Equations (9) to (14) [32].

$$\Gamma_f^{\langle t \rangle} = \sigma_s\left(\Theta_f\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_f\right) \tag{9}$$

$$\Gamma_i^{\langle t \rangle} = \sigma_s\left(\Theta_i\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_i\right) \tag{10}$$

$$\tilde{c}^{\langle t \rangle} = \sigma_h\left(\Theta_c\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right) \tag{11}$$

$$\Gamma_0^{\langle t \rangle} = \sigma_s\left(\Theta_0\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_0\right) \tag{12}$$

$$c^{\langle t \rangle} = \Gamma_f^{\langle t \rangle} \circ c^{\langle t-1 \rangle} + \Gamma_f^{\langle t \rangle} \circ \tilde{c}^{\langle t \rangle} \tag{13}$$

$$a^{\langle t \rangle} = \Gamma_f^{\langle t \rangle} \circ \sigma_h\left(c^{\langle t \rangle}\right) \tag{14}$$

where σ_s and σ_h are sigmoid and hyperbolic tangent functions respectively, Θ indicates the weight vector of corresponding gates or candidate value, b is the vector of bias term, superscript $\langle t \rangle$ implies the timestep, $[\cdot]$ is the concatenate operation between two vectors and $\circ$ is the Hadamard product.

3) GRU Cell: Similar to the LSTM, the GRU (Figure 5) was also designed to overcome the vanishing gradient problem when the number of timesteps is increased (i.e. unfolding) on the recurrent layer [33]. The only difference between the GRU and from LSTM unit, the GRU does not have a separate memory cell state like shown in the LSTM cell diagram in Figure 4 with $c^{\langle t \rangle}$ notation when computing the activation output, instead the activation of the GRU is a linear interpolation between the previous activation $a^{\langle t-1 \rangle}$ and the candidate activation $\tilde{a}^{\langle t-1 \rangle}$ values as given Equation (18). Here, the candidate activation is computed similarly to the traditional recurrent unit given in Equation (8), unlikely the value of the reset gate Γ_r is also integrated by multiplying the gate's value with the previous state activation value. The reset gate, Equation (17), yields a value in the range of [0,1] because of the sigmoid function. As a result, when the gate's value closes to zero, the majority of the previously computed hidden state is forgotten as given in Equation (15). Finally, the update gate Γ_u, Equation (16), decides how much information from the previous state needs to be passed along to the current state $a^{\langle t \rangle}$, therefore when the gate's value closes to one, the majority of the current state is forgotten and the current state will mostly depend on the candidate activation. The GRU cell equations are listed as follows [33]:

$$\tilde{a}^{\langle t \rangle} = \sigma_h\left(\Theta_a\left[\Gamma_u^{\langle t \rangle} \circ a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_a\right) \tag{15}$$

$$\Gamma_u^{\langle t \rangle} = \sigma_s\left(\Theta_u\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_u\right) \tag{16}$$

$$\Gamma_r^{\langle t \rangle} = \sigma_s\left(\Theta_r\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_r\right) \tag{17}$$

$$a^{\langle t \rangle} = \Gamma_u^{\langle t \rangle} \circ \tilde{a}^{\langle t \rangle} + (1 - \Gamma_u^{\langle t \rangle}) \circ a^{\langle t-1 \rangle} \tag{18}$$

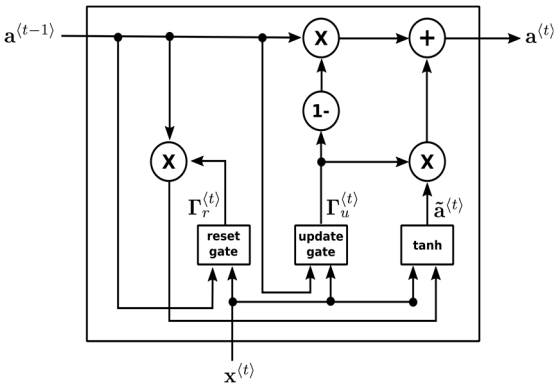

Figure 5. The diagram of a GRU cell at time t.

4) Deep Network Architecture: The detailed architecture of the deep recurrent neural network used in the following experiment is given in Figure 6. The network receives an input vector that is constituted with raw meteorological sensory data and the generated input data $x = \{ys, yc, st, rh, wx, wy, sr\}$. To incorporate temporal input data into the network training, the input data vector is arranged and 2,688 input data x back in time are used for the observation of the past input readings, therefore network looks back 40,320 min (2,688 data points × 15 min), here 15 min is the sampling rate of the meteorological sensors. By dividing 40,320 min with 1,440 (60 min × 24 hours) results 28 days is the total days back in time where the network uses.

However, instead of using every 15-minute input data, every 2 data points (i.e. every 30 min) are presented to the network, in this way dimensionality of the network inputs is reduced. There is no significant difference between 15- and 30-minute sampled data according to autocorrelation analysis [34]. As a result, 1,344 time-lag presents 28-day data observation in past (see Figure 5, where the 1,344 time-lagged input vector $x^{\langle t \rangle}$are shown at the bottom of the figure).

Afterwards, time-lagged inputs are presented to the 1D convolutional layer where 256 filters are used to extract sub-sequences from the input sequences of the network. The convolution operation leads to recognizing the input sequences at different timesteps, therefore the sequences become translation invariant. Then, the outputs from the convolutional layer become the inputs to the first recurrent layer. Here, two recurrent layers are stacked on the network architecture. By stacking multiple recurrent layers on top of each other, all intermediate recurrent layers must return their full-time sequence outputs rather than the output at the last timestep [35]. This is shown in Figure 5 where the outputs from recurrent cells at the first recurrent layer (indicating with superscript [2] as the layer number) are all connected to the associated next recurrent cell on the following recurrent layer (i.e. layer number [3]).

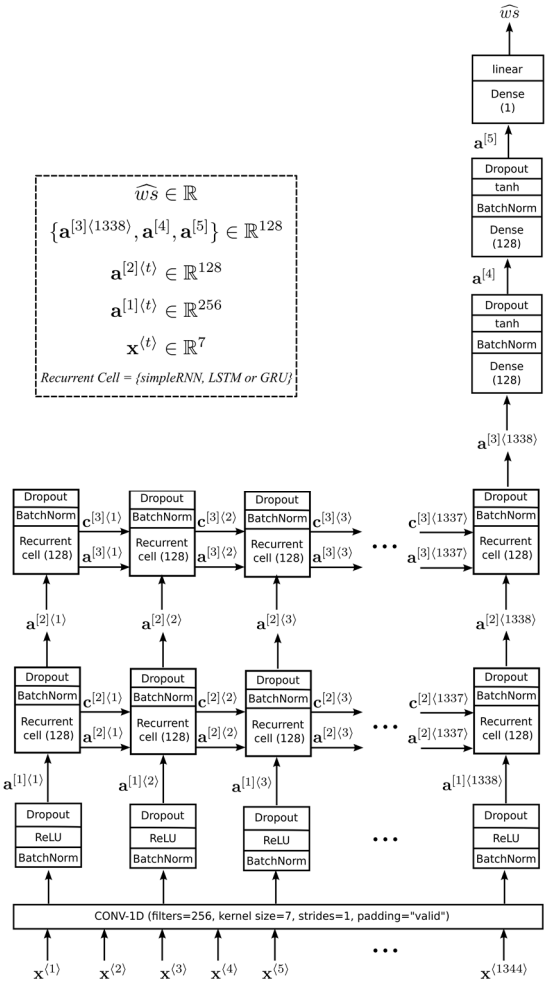

Figure 6. Deep recurrent network architecture for wind prediction. Superscripts $[n]$ and $\langle t \rangle$ indicate the n^{th} layer and t^{th} timestep, respectively. The dimensions of layers and the recurrent cell types for each model are given in the dashed box

After each layer of the DRNN, a batch normalization (BatchNorm) layer was added. The BatchNorm layer is simply normalized to the outputs from the previous layer which is connected to the BatchNorm layer using the mean and standard deviation of the current batch of input samples during the training. The main advantage of using batch normalization is that it overcomes the accumulation of large error gradients that can cause

the exploiting-gradient problem. In addition, by obtaining normalized outputs, the backpropagation algorithm produces a similar range of gradient values that can help move directly and smoothly to the local minimum of the loss optimisation. The two parameters (γ, λ) are learnable in the BathNorm process which is continuously learned during the training of the model, as given in Equation (18).

$$y_i = BatchNorm(x_i) = \gamma \hat{x}_i + \lambda \quad (19)$$

where x_i indicates the previous activation outputs, $\hat{x}_i$ is the standardized value of x_i using the batch mean and standard deviation value of the i[th] connection.

Furthermore, the more layers are added to the network, the network is more prone to become overfitting. To overcome this issue, each node of the network is connected with its corresponding next node by using the dropout rate. Therefore, during the training of the network, randomly selected nodes and their connection weights are disabled from the network's architecture so that the weight balance is spread to all the connection weights in order not to rely on one feature. The last timestamp's output from the last recurrent layer is then connected to a fully connected dense layer, here the dense layer is an ordinary feedforward neural network layer where a hyperbolic tangent activation function is used. Then, another fully connected layer is used at the top of the first dense layer, and finally, the activation outputs from the final dense layer are connected to the output dense layer with linear (identity) activation function to predict one-hour ahead wind speed $\widehat{ws}$. The network learns to predict t+4 timesteps forward of the output data during the training because the sampling rate of the sensory data is 15 min so t+4 represents one hour ahead of the information. It is also important to note that only the recurrent layers (layers 2 and 3) in Figure 6 were replaced with one of the recurrent cell approaches (simple RNN, GRU, or LSTM); thus, three different models were generated: DRNN- with a simple RNN, LSTM, and GRU. These models were trained separately, and their validity was compared for this study.

5) Network Training: Before the network is trained, the obtained dataset is split into training, validation, and testing sets with the ratio of 70%, 20%, and 10% respectively. The training set is used to learn the model of the relationship between inputs and target outputs presented to the network. A validation set is used to validate the network whether the network's training becomes overfitting, under-fitting, or optimal model therefore depending on the validation results in each run, the selected hyper-parameters can be changed to obtain the optimal model at the end of the training process. The initial learning rate α_0 is selected based on some trial training runs and is found to be the best choice to set 0.001. Generally, it is important not to select a too-large learning rate, it may diverge the learning, or too small may slow the convergence. However, when the network learning is near the local minimum on the loss landscape of the network, the learning algorithm needs to make smaller steps to reach the corresponding minima, so this reason the learning rate is reduced exponentially in each epoch of the training. The current learning rate during the training can be computed as follows [35]:

$$\alpha = \alpha_0 e^{-\tau t} \quad (20)$$

In addition, momentum (β) and discount factor (ρ) parameters for the root mean square propagation (RMSprop) optimizer are also set apart from the learning rate to remove the oscillation on the computed gradients to speed up the network learning. RMSprop optimizer is a more sophisticated version than the ordinary backpropagation algorithm. This algorithm also computes the gradients of the loss function (i.e. mean of squares of errors as given in Equation (21) for the network's connection weights by simply applying the chain rule to find the derivative from the loss of the output layer backwards to the corresponding layer's connection weights.

$$\xi(ws, \widehat{ws}) = \frac{1}{m}\sum_{i=1}^{m}(ws_i - \widehat{ws}_i)^2 \quad (21)$$

where m indicates the number of the data in the current batch.

Table 1. Training parameters for proposed DRNN model

Parameter	Description/Value
Optimization algorithm	RMSprop optimizer
Initial Learning Rate (α_0)	0.001
Momentum (β)	0.99
Discount factor (ρ)	0.9
Decay rate (τ)	0.05
Weight initialization	Xavier initialiser
Dropout rate	0.8
Number of epoch	200
Batch size	64

Once the gradients of the corresponding layer's connection weights are computed, then the momentum is applied to the acquired gradients of the connection weight matrix (Θ) and bias vector (b) as given in Equations (22) and (23), and finally, RMSprop is computed by taking the squared gradients of the corresponding connection weights in Equations (24) and (25). In this way, the moving average of the gradients for each connection weight, i.e. smoothed gradients, is used to update the current connection weights of the network. After applying momentum and RMSprop operations, Θ and b of the corresponding layer $[l]$ can be updated as given in Equations (26) and (27) [36].

$$\boldsymbol{V}_{\partial\Theta}^{[l]} = \beta \boldsymbol{V}_{\partial\Theta}^{[l]} + (1-\beta)\frac{\partial\xi}{\partial\Theta^{[l]}} \tag{22}$$

$$\boldsymbol{V}_{\partial b}^{[l]} = \beta \boldsymbol{V}_{\partial b}^{[l]} + (1-\beta)\frac{\partial\xi}{\partial b^{[l]}} \tag{23}$$

$$\boldsymbol{S}_{\partial\Theta}^{[l]} = \rho \boldsymbol{S}_{\partial\Theta}^{[l]} + (1-\rho)\left(\frac{\partial\xi}{\partial\Theta^{[l]}}\right)^2 \tag{24}$$

$$\Theta^{[l]} = \Theta^{[l]} - \alpha\frac{\boldsymbol{V}_{\partial\Theta}^{[l]}}{\sqrt{\boldsymbol{S}_{\partial\Theta}^{[l]}+\epsilon}} \tag{25}$$

$$b^{[l]} = b^{[l]} - \alpha\frac{\boldsymbol{V}_{\partial b}^{[l]}}{\sqrt{\boldsymbol{S}_{\partial b}^{[l]}+\epsilon}} \tag{26}$$

where ϵ is a very small constant number to prevent the denominator from becoming zero, i.e. $\epsilon = 10^{-7}$. Finally, the summary of all training parameters for DRNN is listed in Table 1.

4. EXPERIMENTAL RESULTS AND ANALYSIS

After three RNN layers-based DRNNs are trained, they are tested by using unseen test data from September 2019 to August 2020.

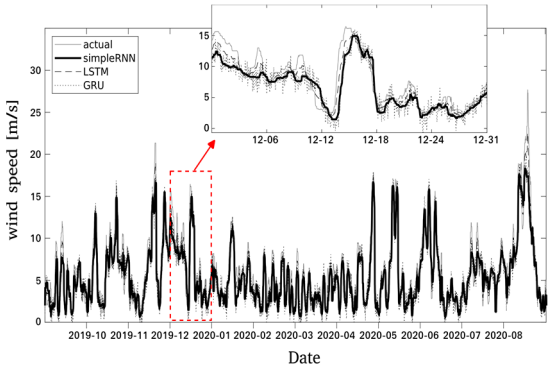

Figure 7. Actual versus predicted values of all DRNN models for one-hour ahead wind speed prediction

As a result, the one-hour-ahead wind speed predictions of the test dataset for the simple RNN, LSTM and GRU-based DRNN are demonstrated in Figure 7. As can be seen, there is some noise in the predicted signals in comparison with true signals. The data from training and testing must come from the same distribution, otherwise, the trained networks' model can struggle to predict true values and could have some fluctuations. These results are expected because of global warming where the seasons are getting changed. The neural network is capable of learning the repeated patterns where there is a correlation between inputs and outputs, however, when the input data changes in the case of global warming and the target output data does not change in the way the inputs changed, it is certainly impossible that the neural network generalizes the altered data well enough because these data are not seen during the network training so that the knowledge of naturally altered data cannot be embedded within the network's weights during the training.

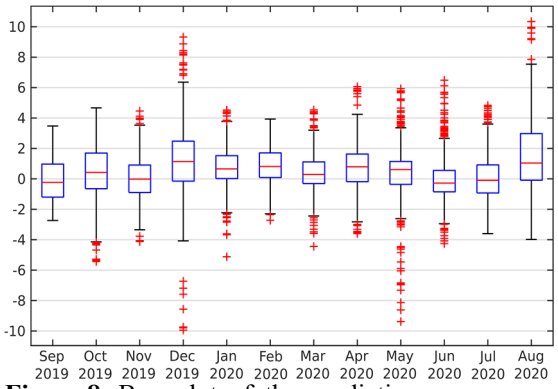

Figure 8. Box plot of the prediction errors over months.

Further analyses are also carried out to examine the network's prediction errors along the months of the year. Figure 8 shows the box plot of the network's prediction error in each month. It can be seen that the highest number of errors were in December 2019 and August 2020. This is shown with the median error values (middle horizontal line in the box) which have the furthest distance to the zero error for the corresponding months.

Furthermore, the difference between the lower and the higher end of the boxes indicates how the variability of the error data from the median error, in other words, this is also called mathematically interquartile range (IQR). The IQR values of these two months are higher than the other months so 50% of the error data (the box indicates the middle 50% of data) are spread out more because the network has high uncertainty on the predictions. In addition, the whiskers of the boxes for the corresponding months indicate that they have the highest positive and negative errors among the other months. There are also lots of outlier data points of the prediction error which are indicated with the (+) symbol for December 2019, May 2020, and June 2020. The network makes these types of outlier predictions whenever the corresponding input data are significantly different from the training data. The data used in the network training are obtained from earlier years than the test data used for the network evaluation. This difference between the years is possible because global warming causes weather shifts over the years.

To clarify whether the occurrence of the high errors in December 2019 and August 2020 whether or not is dependent on the distribution of training and testing datasets, the *Kolmogorov Smirnov* (KS) test is carried out to compare the raw meteorological sensory data of these months in each year from 2017 to 2020. Suppose the relationship between the sensory data of the specific month at the different years is not changed due to global warming. In that case, the acquired model of DRNN can generalise the prediction well, even if the magnitude of those data is changed in the following year. First of all, the training data which belongs to August month from 2017 to 2019 are separately compared with the test data belonging to August 2020.

The results show that only precipitation data in 2019 and 2020 of the August month are from the same distribution at the 5% significance level and the null hypothesis is rejected for the other meteorological sensory data (*i.e.* temperature, humidity, solar radiation, wind speed, and direction). Similar results are also obtained when comparing the test data of December 2019 against the training data of December 2017 and 2018. This is an expected outcome and shows how well the acquired network's model generalizes the meteorological data with an acceptable error rate even if they come from a different distribution.

4.1. Comparison between LSTM, GRU and Simple RNN

To assess the performance of the DRNN using different recurrent layers, various statistical analyses are carried out. Firstly, the correlation coefficient values between the DRNN's predicted wind speed and the actual wind speed are computed by using three different correlation analyses, these are Spearman, Pearson, and Kendall correlation coefficients. The value of the correlation coefficient indicates how correlated predicted outputs are and actual outputs, the value 1 implies perfect correlation, 0 is a random guess and -1 means negative correlation. Furthermore, prediction errors of the network on the test data are also interpreted therefore mean absolute error (MAE), root mean square error (RMSE), and mean absolute percentage error (MAPE) are calculated.

Table 2 shows the statistical results of all recurrent layers using the same test data. As can be seen, there is a strong correlation between the network-predicted and actual wind speed for all RNN layer-based DRNN where their values are over 0.8 for both Spearman and Pearson correlations, however, both GRU and simple RNN-based recurrent layers on the DRNN architecture presents better than LSTM-based layer. In addition, the simple RNN-based DRNN demonstrates slightly better performance than the GRU-based DRNN according to the Spearman and Kendall coefficients, whereas the GRU-based DRNN shows better performance in terms of the RMSE metric.

Table 2. Correlation coefficient values and overall prediction errors between each RNN layer for 1-hour ahead prediction

	Spearman	**Pearson**	**Kendall**	**MAE [m/s]**	**RMSE [m/s]**	**MAPE [%]**
LSTM	0.8312	0.8943	0.6477	1.4819	1.9360	34.3462
GRU	0.8815	0.9237	0.7079	1.3402	1.7863	27.9873
Simple RNN	0.8829	0.9166	0.7089	1.3088	1.8010	26.7737

Similar results are also obtained when prediction errors of the networks are analyzed. Both GRU and simple RNN-based DRNN have close overall prediction error according to MAE and RMSE, however, the accuracy of the Simple RNN has slightly better accuracy than GRU layers according to the MAPE values which are 72.01% and 73.23% (100 - MAPE) for GRU and simple RNN-based layers, respectively. Further analyses are also carried out between GRU and simple RNN-based layers to investigate how widely prediction errors deviate from the mean error, therefore standard errors for both networks are computed as $\overline{\sigma}_{GRU}$=0.0002056 m/s and $\overline{\sigma}_{simpleRNN}$=0.0002138 m/s. This shows that the prediction of using a simple RNN layer on the DRNN is slightly spread out more than the GRU-based layer. On the other hand, the confidence intervals of the prediction errors with a 95% significant level for GRU and simple RNN layers are also reported as [0.6931, 0.6939] m/s and [0.5581, 0.5590] m/s respectively. Although the simple RNN-based DRNN has a higher standard error, its prediction errors lie in the smaller range in comparison with GRU-based DRNN. Consequently, the lower error range yields better network performance.

5. CONCLUSIONS

Due to global warming, weather conditions are changing, and previously obtained forecasting models may fail to predict wind speed accurately. Previous models that relied only on previous wind speed information are no longer sufficient, as global warming affects multiple meteorological factors such as precipitation, temperature, humidity, and solar radiation. Therefore, a deep recurrent neural network is proposed to model the relationship between multiple meteorological sensory data to produce more robust wind speed predictions. This approach can adapt to changes in weather conditions and is capable of producing accurate wind speed predictions even as wind speeds change each year. The proposed model is designed for one-hour-ahead wind speed predictions in an agricultural area.

To predict future wind speed, raw meteorological sensory readings are transformed into non-cyclical features such as wind vectors and timestamps. The mutual information of these features is analyzed to determine their contribution to the prediction. Three types of recurrent layers (LSTM, GRU, and simple RNN) are tested in a DRNN architecture to model the relationship between sensory data and wind speed. The models are validated using unseen sensory data from September 2019 to August 2020, and statistical analyses are performed to compare their performance. The results show that there is a strong correlation between the predicted and actual wind speeds for all models, which used different types of recurrent layers. Furthermore, the distribution of the mean prediction error along the months of the year is also analyzed using box plot

analyzes to highlight the month in which the highest average error occurred. It is reported that the networks performed worse in December 2019 and August 2020 than in other months. To clarify this, the *Kolmogorov-Smirnov* test is carried out, and it is found that the training and the test sensory data in these two months are not from the same continuous distribution. These results are expected because global warming affects the weather in different ways between consecutive years. Thus, the sensory readings between the years become significantly different. In conclusion, a deep recurrent network is capable of learning very high-degree nonlinear data, and by combining multiple recurrent layers on the network's architecture, it becomes more powerful than an ordinary one-layer recurrent neural network to model the noisy sensory data.

6. REFERENCES

1. 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.
2. Chen, Y., Dong, Z., Wang, Y., Su, J., Han, Z., Zhou, D., Zhang, K., Zhao, Y., Bao, Y., 2021. Short-term Wind Speed Predicting Framework Based on the Eemd-ga-lstm Method under Large-scale Wind History. Energy Conversion and Management, 227, 113559.
3. Hayes, L., Stocks, M., Blakers, A., 2021. Accurate Longterm Power Generation Model for Offshore Wind Farms in Europe using Era5 Reanalysis. Energy, 229, 120603.
4. Deng, X., Shao, H., Hu, C., Jiang, D., Jiang, Y., 2020. Wind Power Forecasting Methods Based on Deep Learning: A Survey. Computer Modeling in Engineering & Sciences, 122(1), 273-301.
5. Mi, X., Liu, H., Li, Y., 2019. Wind Speed Prediction Model Using Singular Spectrum Analysis, Empirical Mode Decomposition and Convolutional Support Vector Machine. Energy Conversion and Management, 180, 196-205.
6. Lei, M., Shiyan, L., Chuanwen, J., Hongling, L., Yan, Z., 2009. A Review on the Forecasting of Wind Speed and Generated Power. Renewable and Sustainable Energy Reviews, 13(4), 915-920.
7. Azimi, R., Ghofrani, M., Ghayekhloo, M., 2016. A Hybrid Wind Power Forecasting Model Based on Data Mining and Wavelets Analysis. Energy Conversion and Management, 127, 208-225.
8. Santhosh, M., Venkaiah, C., Vinod K.D.M., 2020. Current Advances and Approaches in Wind Speed and Wind Power Forecasting for Improved Renewable Energy Integration: A Review. Engineering Reports, 2(6), e12178.
9. Lipu, M.S.H., Miah, M.S., Hannan, M.A., Hussain, A., Sarker, M.R., Ayob, A., Saad, M. H.M., Mahmud, M.S., 2021. Artificial Intelligence Based Hybrid Forecasting Approaches for Wind Power Generation: Progress, Challenges and Prospects. IEEE Access, 9, 102460-102489.
10. Puri V., Kumar, N., 2021. Wind Energy Forecasting Using Artificial Neural Network in Himalayan Region. Modeling Earth Systems and Environment, 1-10.
11. Li, L.L., Chang, Y.B., Tseng, M.L., Liu J.Q., Lim, M.K., 2020. Wind Power Prediction Using a Novel Model on Wavelet Decomposition-Support Vector Machines-Improved Atomic Search Algorithm. Journal of Cleaner Production, 270, 121817.
12. Sfetsos, A. 2000. A Comparison of Various Forecasting Techniques Applied to Mean Hourly Wind Speed Time Series. Renewable Energy, 21(1), 23-35.
13. Lin, W., Wu, Z., Lin, L., Wen, A., Li, J., 2017. An Ensemble Random Forest Algorithm for Insurance Big Data Analysis. IEEE Access, 5, 16568–16575.
14. Tian, Z., Li, S., Wang, Y., 2020. A Prediction Approach Using Ensemble Empirical Mode Decomposition-Permutation Entropy and Regularized Extreme Learning Machine for Short-term Wind Speed. Wind Energy, 23(2), 177-206.
15. Huang, G.B., Zhu, Q.Y., Siew, C.K, 2006. Extreme Learning Machine: Theory and Applications. Neurocomputing, 70(1), 489-501.
16. Afrasiabi, M., Mohammadi, M., Rastegar, M., Afrasiabi, S., 2021. Advanced Deep Learning Approach for Probabilistic Wind Speed Forecasting. IEEE Transactions on Industrial Informatics, 17(1), 720-727.

17. Liu, Y., Guan, L., Hou, C., Han, H., Liu, Z., Sun, Y., Zheng, M., 2019. Wind Power Short-term Prediction Based on lstm and Discrete Wavelet Transform. Applied Sciences, 9(6).
18. Liu, H., Mi, X., Li, Y., 2018. Smart Multi-step Deep Learning Model for Wind Speed Forecasting Based on Variational Mode Decomposition, Singular Spectrum Analysis, lstm Network and Elm. Energy Conversion and Management, 59, 54-64.
19. Ding, M., Zhou, H., Xie, H., Wu, M., Nakanishi, Y., Yokoyama, R., 2019. A Gated Recurrent Unit Neural Networks Based Wind Speed Error Correction Model for Short-term Wind Power Forecasting. Neurocomputing, 365, 54-61.
20. Kisvari, A., Lin, Z., Liu, X., 2021. Wind Power Forecasting a Data-driven Method Along with Gated Recurrent Neural Network. Renewable Energy, 163, 1895-1909.
21. Duan, J., Zuo, H., Bai, Y., Duan, J., Chang, M., Chen, B., 2021. Short-term Wind Speed Forecasting Using Recurrent Neural Networks with Error Correction. Energy, 217, 119397.
22. Wang, L., Li, X., Bai, Y., 2018. Short-term Wind Speed Prediction Using an Extreme Learning Machine Model with Error Correction. Energy Conversion and Management, 162, 239-250.
23. Memarzadeh, G., Keynia, F., 2020. A New Short-term Wind Speed Forecasting Method Based on Fine-tuned lstm Neural Network and Optimal Input Sets. Energy Conversion and Management, 213, 112824.
24. Liu, H., Mi, X., Li, Y., Duan, Z., Xu, Y., 2019. Smart Wind Speed Deep Learning-based Multi-Step Forecasting Model Using Singular Spectrum Analysis, Convolutional Gated Recurrent Unit Network and Support Vector Regression. Renewable Energy, 43, 842-854.
25. Yu, C., Li, Y., Zhang, M., 2017. An Improved Wavelet Transform Using Singular Spectrum Analysis for Wind Speed Forecasting Based on Elman Neural Network. Energy Conversion and Management, 148, 895-904.
26. Yu, C., Li, Y., Bao, Y., Tang, H., Zhai, G., 2018. A Novel Framework for Wind Speed Prediction Based on Recurrent Neural Networks and Support Vector Machine. Energy Conversion and Management, 178, 137-145.
27. Orr, R.J., Griffith, B.A., Rose, S., Hatch, D., Hawkins, J., Murray, P.J., 2011. Designing and Creating the North Wyke Farm Platform. Catchment Science.
28. Hawkin, J., 2015. Design, Establishment and Development, http://resources.rothamsted.ac.uk/sites/default/files/groups/NorthWykeFarmPlatform/FPUG.Doc.001EstabDevelopver1.5.pdf, Access date: 11/02/2023.
29. Boden, M., 2002. A Guide to Recurrent Neural Networks and Backpropagation. The Dallas Project.
30. Elman, J.L., 1990. Finding Structure in Time. Cognitive Science, 14(2), 179-211.
31. Goodfellow, I., Bengio, Y., Courville, 2016. Deep Learning. MIT Press.
32. Hochreiter, S., Schmidhuber, J., 1997. Long Short-term Memory. Neural Computation, 9(8), 1735-1780.
33. Chung, J., Gulcehre, C., Cho, K., Bengio, Y., 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modelling. arXiv Preprint arXiv:1412.3555.
34. Kaplan, D., Glass, L., 1997. Understanding Nonlinear Dynamics. Springer Science & Business Media.
35. Chollet, F., 2017. Deep Learning with Python. Manning.
36. Werbos, P.J., 1990. Backpropagation Through Time: What it Does and How to do It. Proceedings of the IEEE, 78(10), 1550-1560.