Comparative performance study of DBN, LSTM, CNN and SAE models for wind speed and direction forecasting

1st Régis Donald HONTINFINDE*
LSIMA^{+}
University of Abomey (UNSTIM)
Abomey, Republic of Benin
donald.hontinfinde@yahoo.com
*Corresponding author

2nd K. B. Audace DIDAVI
DEE^{++}*, EPAC*
University Of Abomey Calavi (UAC)
Cotonou, Bénin
didavia@gmail.com

3rd Mahugnon Géraud AZEHOUN PAZOU
LSIMA^{+}
University of Abomey (UNSTIM)
Abomey, Republic of Benin
geraud.pazou@gmail.com

4th Faith Y. P. J. HOUESSOU
LSIMA^{+}
Universiry of Abomey (UNSTIM)
Abomey, Republic of Benin
faith03houessou@gmail.com

5th Christian Djidjoho Akowanou
LSIMA^{+}
Universiry of Abomey (UNSTIM)
Abomey, Republic of Benin
djidjohoako@yahoo.fr

6th Macaire B. AGBOMAHENA
DEE^{++}*, EPAC*
University Of Abomey Calavi (UAC)
Cotonou, Bénin
agbomac@yahoo.fr

***Abstract*—More evenly distributed renewable energy sources are emerging as an alternative to fossil fuels, especially for power generation in rural areas. This study compares four of the most widely used models, the Convolutional Neural Network (CNN), the Long Term Memory (LSTM), the Stacked Autoencoder (SAE) and the Deep Belief Network (DBN), for forecasting wind speed and direction in order to control the output of a wind turbine generator. The meteorological data used were downloaded from the NASA database for the period January 1, 2012 to December 31, 2021 on the Benin coast. The forecasting model developed combines one of the LSTM, CNN, DBN, SAE models and the Fast Fourier Transform, which was used to extract the most important frequencies in the data. The results of the comparison showed that the best-performing model was the LSTM, with respective root mean square errors (RMSE) and coefficients of determination of 0.36 m/s and 92% for wind speed, and 8.1 ° and 71% for wind direction.**

Keywords : Machine learning algorithms - Forecast - Wind speed and direction - LSTM - DBN - CNN - SAE

I. Introduction

In order to expand their networks and reach often isolated populations in rural areas, Telecom Network Companies often need to install their equipment (including Base Transceiver Stations (BTS)) in remote areas with no access to conventional power grids. Renewable energy sources, given their more uniform distribution and the fact that they are in line with policies to reduce greenhouse gas emissions, are a viable alternative to the fossil fuels often used in generating sets. With the current context, the purchase price as well as the transport cost of diesel have increased on the market. As alternatives to diesel, telecom operators are exploring other renewable energy sources to power mobile base stations. Renewable energies include solar, wind, hydraulic, biomass and geothermal. The favorable climatic and geographical conditions of Africa and in particular of Benin give it considerable potential in terms of renewable energies [1]. Benin benefits from abundant sunshine in many regions. Photovoltaic solar panels can be widely deployed to generate solar-generated electricity. In addition to these factors, Benin, like many regions of Africa, has considerable wind potential, particularly along the coasts and in desert areas. Wind energy can be harnessed using wind turbines. All this theoretical renewable potential should allow telecommunications operators to do without easy sources to power base stations by proposing new hybrid power architectures for telecommunications relay sites. Indeed, beyond investment issues, the integration of renewable energies into existing energy systems in general and in the telecommunications sector in particular can present a certain number of challenges, among which: they are generally temporal and spatial . -dependent, have low electricity production capacity and low energy efficiency. As a result, their integration (mainly photovoltaic solar and wind) complicates grid balancing, degrades grid reliability and increases the cost of electricity due to technical constraints [2]– [4]. Faced with this random variation in the power generated by solar photovoltaic and wind sources, several solutions can be considered, in particular the prediction of the variability of the output power [5]– [6]. Several forecasting models have therefore been developed: the persistence method [7], physical methods [8]– [9] and generally more precise statistical methods including artificial intelligence techniques

LSIMA^{+}:Laboratory of Science Engineering and Mathematics
DEE^{++}:Department of Electrical Engineering

such as Convolutional Neural Network (CNN). , Long-term memory (LSTM).), Stacked Autoencoder (SAE) or Deep Belief Network (DBN) [10]– [12]. In this article, we propose a comparative study of Deep Learning models to predict wind speed variability, with the aim of effectively controlling wind production for their use in the telecommunications sector to power base stations. In the second section, the methodology adopted is described step by step, as well as the equipment used. The third section presents the results obtained and concluded.

II. METHOD

A. Literature review on Deep Learning models used for predicting wind power

The number of publications reporting on deep learning-based methods has grown fast in recent years. This is illustrated in Figure 2-a, which shows the number of articles on deep learning-based wind energy research found in the SCOPUS database for the period 2014-2023 (the last ten years). The figure shows that most of the articles in SCOPUS are published in the last three years, with 4292 articles out of a total of 7215 published from January 2014 to February 2023.This exponential growth shows the importance of predictions and the current interest in them in the scientific community. Recently, interest in hybrid models has increased. Almost half of the articles published in SCOPUS in 2022 concern hybrid forecasting models. This trend is explained by the superiority of these models over simple deep learning models, as shown by all comparative experiments. These hybrid models are either a combination of several Deep Learning models, or a combination of a Deep Learning model and a data processing technique. Of these hybrid models encountered, the majority include CNNs, LSTMs, DBNs, SAEs or GRUs. RNNs and all their newer versions, such as LSTM and GRU, are the most popular models after hybrid models, which makes sense because renewable energy data are time series, and RNN models are known for their ability to extract temporal features. This is why researchers use them alone or in combination with other methods. Figure 2-b shows the percentage use of each deep learning architecture in papers published in SCOPUS in 2022 (for wind power forecasting). In this study, we have therefore selected the CNN, LSTM, DBN and SAE models for a comparative study, combined with data processing methods.

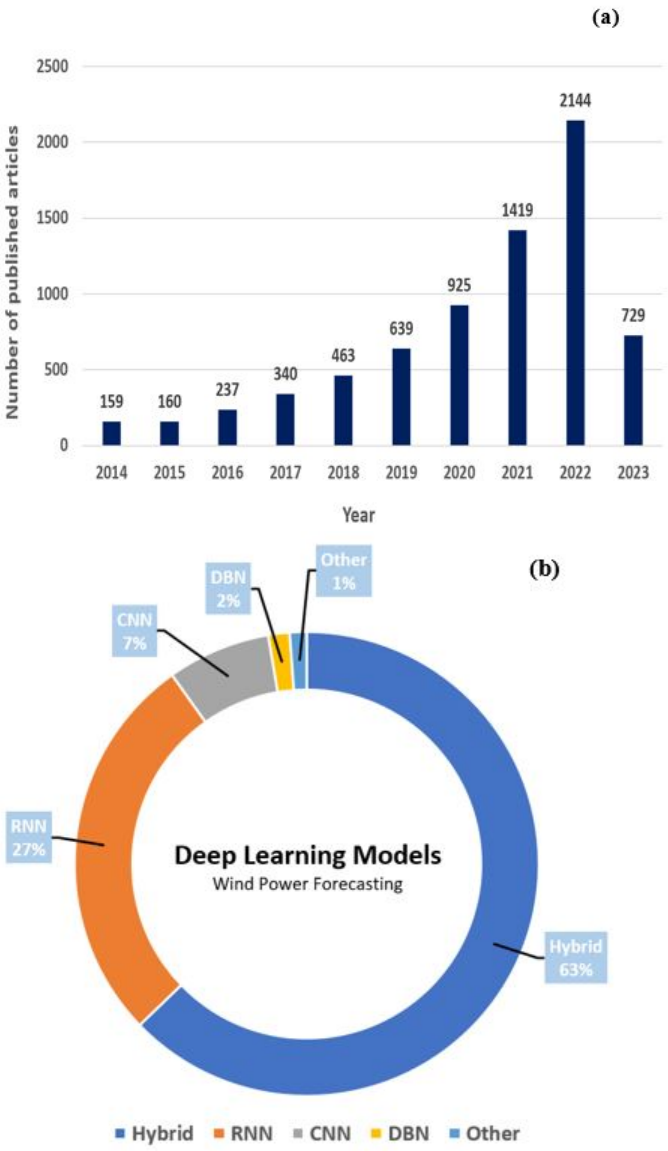

Fig. 1. (a) Number of publications reporting on deep learning-based methods in SCOPUS, (b) Deep learning models for wind energy forecasting used in 2022 publications in SCOPUS.

B. Data acquisition and pre-processing

Once the meteorological parameters to be studied (wind speed and direction) were known, a database was identified that could provide the related data for all the samples. For our work, NASA's POWER — Data Access Viewer database was identified and exploited [13] . Information on the data downloaded from it is presented below:

- **Period**: January 1, 2012 to December 31, 2021;
- **Satellite**: MERRA-2;
- **Resolution**: (1/2)° × (5/8)° latitude/longitude ;
- **Longitude** : 2.3855;
- **Latitude** : 6.3471;
- **Data step**: hourly;
- **Parameters**:
  - W_s: Wind speed at 50 meters from ground in (m/s);
  - W_d: Wind direction at 50 meters from ground in (°);

To obtain high-performance forecasting models, input data usually needs to be pre-processed. This is the most important step, and usually the largest part of a forecasting task, as a forecasting model only draws information from the data it is presented with. Data pre-processing is the process of preparing raw data and making it suitable for a machine learning model. It is the crucial first step in creating a machine learning model. In our proposed data processing technique, we extract the

most important frequencies in the data to integrate the concept of periodicity into our dataset in order to increase model accuracy. Since we're dealing here with meteorological time series, they often have clear daily or annual periodicity, for example. To facilitate forecasting, it is important to make models understand these notions of frequency. To this end, Python 3's Fast Fourier Transform (FFT) is used to determine the important frequencies in these series.

It is seldom possible to exploit parameters and their data as downloaded or acquired in a forecasting task. Typically, these parameters are either transformed, combined or supplemented to create a new dataset in order to give the models a precise understanding of the target information. The first variables added to the dataset are those created from the significant frequencies identified earlier. For each significant frequency, two variables described by equations (1) and (2) are added:

$$x_{\sin} = \sin\left(t * \left(\frac{2\pi}{T}\right)\right) \tag{1}$$

$$x_{\cos} = \sin\left(t * \left(\frac{2\pi}{T}\right)\right) \tag{2}$$

where x_{sin} and x_{cos} are the newly added variables, t is the identified important period. T is the identified important period.

In our case, it's also important to make the models understand the concept of wind direction. Wind directions are angles, and models need to understand that 0° and 360° are identical. What's more, wind direction isn't very useful if the wind is weak. The two variables (wind speed and direction) are therefore combined to create two new variables, as shown in equations (3) and (4).

$$W_x = W_s \sin(W_d) \tag{3}$$

$$W_y = W_s \cos(W_d) \tag{4}$$

W_x and W_y are the two new variables created, W_s wind speeds in m/s and W_d wind directions in radian. Once data cleansing has been carried out and new variables added, quality data must be consolidated in other forms by modifying the value, structure or format of the data, using data transformation strategies such as normalization or standardization. Normalization, often simply called Min-Max scaling, reduces the extent of the data so that it is set between 0 and 1 (or between -1 and 1 if there are negative values). It is most effective in cases where standardization doesn't work as well. If the distribution is not Gaussian, or if the standard deviation is very small, the min-max scale works better. Normalization is generally performed using the following equation:

$$x_{norm} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{5}$$

Finally, the dataset is split. The aim of splitting the data into time series is similar to that of random splitting, namely to validate the predictability of the model irrespective of how the training-test datasets are split. However, time-series splitting ensures that the test datasets are more recent or older than the training datasets, which is more realistic since we won't be able to train on "future" data. We have opted for a split into three parts: Training, Validation and Test, with 70%, 20% and 10% respectively.

C. Modeling and hyper parameter optimization

1) Modeling: As indicated in section 2.2, the models selected for this study are LSTM (Long short-term memory), CNN (Convolutional Neural Network), DBN (Deep Belief Network) and SAE (Stacked Autoencoder).

Long Short-Term Memory

The structure of an LSTM model is composed of three main gates:

- **Forget Gate**
- **Input Gate**
- **Output Gate**

Mathematically, the operations in an LSTM are defined as follows, x_t is the input at time t, h_t is the hidden state at time t, c_t is the cell state at time t, σ is the sigmoid function, and $\tanh$ is the hyperbolic tangent function.

- Calculation of the forgetting gate (f_t):
  $f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$
- Calculation of the input gate (i_t):
  $i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$
  $c'_t = \tanh(W_c * [h_{t-1}, x_t] + b_c)$
- Update cell status (c_t):
  $c_t = f_t * c_{t-1} + i_t * c'_t$
- Output gate calculation (0_t):
  $o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$
- Calculation of the forgetting gate (h_t):
  $h_t = o_t * \tanh(c_t)$

In these equations,W_f,W_i,W_c and W_o are the weight matrices for the different gates, and b_f,b_i,b_c and b_o are the corresponding bias vectors.

Convolutional Neural Network

A CNN model is structured as follows:

- **Convolution layer**: The first layer of a CNN is generally a convolution layer. It applies a number of filters (kernels) to the input data to extract relevant features. Each filter traverses the data and produces an "activation map" by applying a convolution operation. Let I be the input data, K the convolution filter, and B the associated bias. The convolution operation for an output feature O is given by:

$$O(i,j) = \sum_m \sum_n I(i+m, j+n) * K(m,n) + B \tag{6}$$

- **Activation function**: After each convolution operation, an activation function (such as ReLU - Rectified Linear Unit) is applied element by element to the resulting activation map. This introduces non-linearity into the model, enabling it to learn complex patterns.

$$ReLU(x) = \max(0, x) \tag{7}$$

- **Pooling layer**: After certain convolution layers, pooling operations (such as max-pooling) are performed to reduce the spatial dimension of the activation map and the number of parameters in the model. Pooling retains the essential features. For a given region of the activation map, max-pooling retains the maximum value:

$$Max - Pooling(X) = \max(region) \tag{8}$$

- **Fully Connected (Dense) layers**: One or more fully connected layers can follow the convolution and pooling layers. These layers perform linear operations on the extracted features to perform final classification or other tasks. Let X be the output of the previous layer, W the weight matrix and b the associated bias. The fully connected operation is given by:

$$Y = XW + b \tag{9}$$

- **Output layer**: The last layer of the CNN produces the output results, usually using an appropriate activation function (such as softmax for classification) to obtain probabilities or class scores.

Deep Belief Network

A DBN consists of two main layers:

- Visible layers: The first layer of the network, called the visible layer, represents the model inputs. These inputs can be binary or real variables
- Hidden layers: The intermediate layers are called hidden layers. These layers are organized into several levels, with each level connected to the visible and other hidden layers. The neurons in these layers are generally binary, representing features extracted from the input data.

Mathematically, a DBN can be represented as follows:

- Boltzmann Restricted Network (BRN): Each pair of adjacent layers in a DBN is modeled as a BRN. An BRM is a probabilistic model with a visible layer v and a hidden layer h. Activations of the visible layer are denoted $v = (v_1, v_2, ..., v_n)$ and those of the hidden layer are denoted $h = (h_1, h_2, ..., h_m)$.
  The energy of a particular state (v, h) in an BRM is given by :

$$E(v,h) = -\sum_{i=1}^{n}\sum_{j=1}^{m} W_{ij} v_i h_j - \sum_{i=1}^{n} b_i\, v_i - \sum_{j=1}^{m} c_j\, h_j$$

  where W_{ij} are the weights between the visible v_i and hidden h_j units, b_j and c_j are the biases of the visible and hidden units respectively.
- Top-down Propagation (Fine-Tuning): Once the BRMs have been pre-trained layer by layer, the network is usually fine-tuned using a technique such as gradient back-propagation to adjust the weights and biases of the entire network, thus optimizing model performance for the specific task.

Stacked Autoencoder

An SAE typically consists of:

- **Encoder**: The encoder transforms the input data into an internal representation (encoding) using a layer of neurons called the encoding layer. This layer is generally narrower than the input layer, resulting in information compression.
- **Decoder**: The decoder reconstructs the input data from the internal representation obtained from the encoder. The aim of the decoder is to minimize the reconstruction error between the input data and the reconstructed data.

2) Hyper parameter optimization: As presented previously, LSTM, CNN, DBN and SAE networks are made up of several layers whose configurations can influence their learning performance. Thus, for a given forecasting task, the optimal parameterization to achieve good forecasting performance with reduced computation time must be sought. In general, building an efficient machine learning model is a complex and tedious process, involving determining the appropriate algorithm and obtaining an optimal model architecture by tuning its hyperparameters (HP). To build an optimal model, we need to explore a range of possibilities. In Table 1 below, models hyperparameters and variation ranges are presented.

TABLE I
MODELS HYPERPARAMETERS AND VARIATION RANGES

Hyper Parameters	Hyper Parameters
Input data size	2 to 24 hours in 1-hour steps
Output data size	1 to 23 hours in 1-hour steps
Learning_rates	Between 10^{-2}, 10^{-3} and 10^{-4}
CNN	
Number of Filters	10 to 300 in steps of 10
Number of units	10 to 1000 in steps of 10
LSTM	
Number of units	10 to 1000 in steps of 10
DBN	
Hidden layers	1 to 6 in steps of 1
Hidden layer dimension	32 to 260 in steps of 32
SAE	
Number of units	10 to 1000 in steps of 10

Hyper-parameter optimization is performed using the Keras-tuner library in Python3 [14] .

D. Models training

Once the optimal parameters of the models have been determined, they are then trained on the dataset. The models are developed and trained using Tensorflow under python3. TensorFlow is a leading open source platform for machine learning. It features a comprehensive and flexible ecosystem of tools, libraries and community resources that enable researchers to advance machine learning and developers to easily create and deploy machine learning-based applications [15] .

E. Model performance evaluation

To evaluate the models, we use the Mean Square Error (RMSE) and the coefficient of determination (R^2) as metrics. As a boundary condition of the forecast horizon when optimizing hyperparameters, we aim to obtain models for

which the RMSE is less than 10% of the nominal value of the study parameter. The RMSE and R^2 equations are given in (1) and (2) respectively:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2} \tag{10}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2}{\sum_{i=1}^{n}\left(Y_i - \hat{Y}\right)^2} \tag{11}$$

III. RESULTS

A. Optimized hyperparameters

Table II shows the hyperparameter optimization results. We can see that the maximum horizon is 8h, in accordance with the condition stated in 2.6, with an input data size of 12h. For the LSTM model, the optimal number of units is 560. For CNN, the optimal number of filters and units are 180 and 320 respectively. For the DBN model, the number of hidden layers and layer size are 2 and 128 respectively. The optimal number of SAE units is 370. The optimal learning rates for all models are 0.001.

TABLE II
OPTIMIZED MODELS HYPERPARAMETERS

HYPER PARAMETERS	Value
Input data size	12h
Output data size	8h
Learning_rates	10^{-3}
CNN	
Number of Filters	180
Number of units	320
LSTM	
Number of units	560
DBN	
Hidden layers	2
Hidden layer dimension	128
SAE	
Number of units	370

B. Forecasting results

Once the optimized hyperparameters had been found, they were used with the models on the Test set to make forecasts. Tables III, IV 4 below show the forecasting performance of the CNN, LSTM, DBN and SAE models over the entire Test set of parameters W_x, W_y respectively.

TABLE III
W_x PREVISION PERFORMANCES

	LSTM	CNN	DBN	SAE
RMSE (m/s)	0.47	0.61	0.52	1.02
R^2	0.89	0.76	0.81	0.65

It can be noted that, overall, the best-performing model is the LSTM model for Wx and Wy parameter predictions, followed in order by the DBN, CNN and SAE models. From these two predicted parameters W_x and W_y, velocity and direction can then be deduced using expressions (11) and (12) below.

TABLE IV
W_y PREVISION PERFORMANCES

	LSTM	CNN	DBN	SAE
RMSE (m/s)	0.32	0.41	0.31	0.53
R^2	0.85	0.78	0.84	0.74

$$W_d = \tan^{-1}\left(\frac{W_y}{W_x}\right) \tag{12}$$

$$W_s = \frac{W_y}{\sin\left(\tan^{-1}\frac{W_y}{W_x}\right)} \tag{13}$$

Tables V and VI below show the forecasting performance of the CNN, LSTM, DBN and SAE models over the entire Test set of parameters W_s and W_d respectively.

TABLE V
W_s PREVISION PERFORMANCES

	LSTM	CNN	DBN	SAE
RMSE (m/s)	0.36	0.48	0.42	0.78
R^2	0.92	0.85	0.89	0.78

TABLE VI
W_d PREVISION PERFORMANCES

	LSTM	CNN	DBN	SAE
RMSE (m/s)	8.1	11.2	8.6	15
R^2	0.71	0.62	0.68	0.56

Figures 2 and 3 below show the predicted and actual W_x and W_y respectively for a randomly selected portion of the dataset. We can see that the performances presented in Tables 3 and 4 are confirmed: the values predicted by the LSTM model are those closest to the true values, followed by DBN, CNN and SAE. It is worth remembering that the indexes represent the order numbers of the forecast times

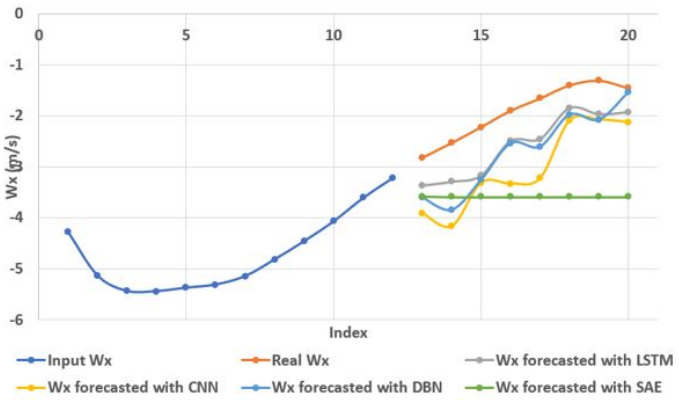

Fig. 2. W_x Real and Forecasted

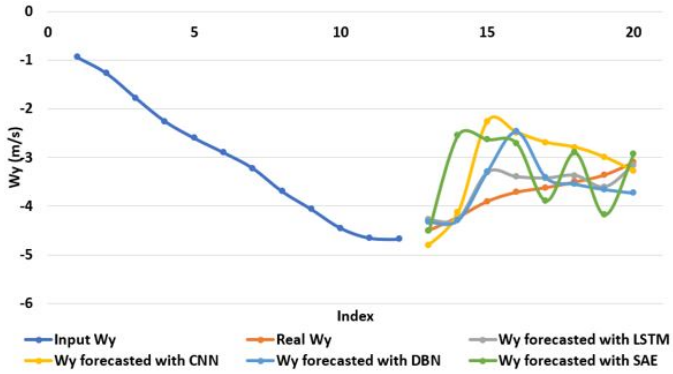

Fig. 3. W_y Real and Forecasted

Figures 4 and 5 show predicted and true speeds and directions. The conclusion is the same: the best-performing model is the LSTM. Overall, we can see that the best-performing model for forecasts remains the LSTM model, followed by the DBN, CNN and SAE models in that order.

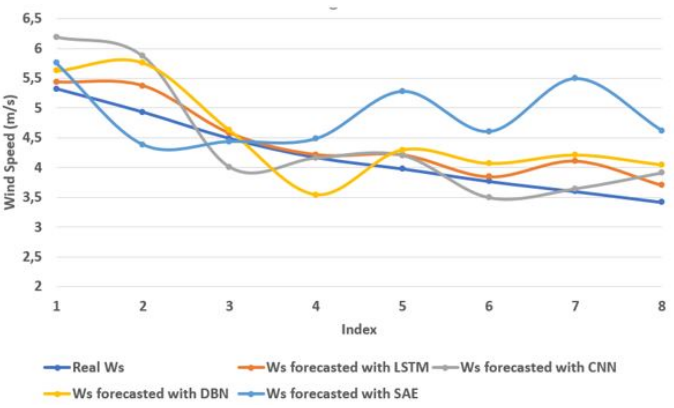

Fig. 4. Wind Speed Real and Forecasted

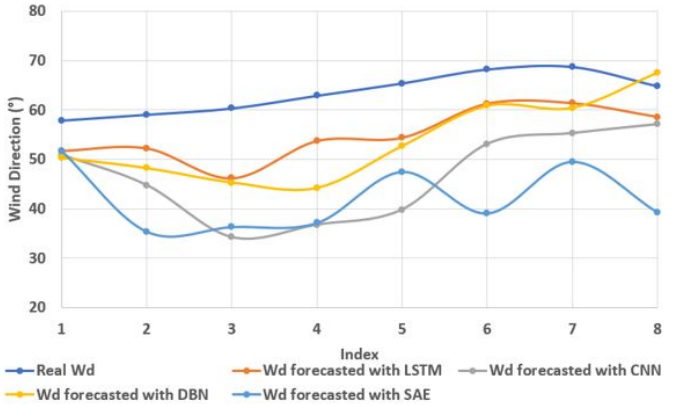

Fig. 5. Wind Direction Real and Forecasted

IV. Conclusion

Advances in mathematical modeling, physical representations, statistical analysis and computing power have made forecasting a viable option today. In our case, the prediction of wind speeds and directions is essential for the smooth and efficient integration and operation of wind turbines. In this work, a comparative study between the LSTM, CNN, DBN and SAE models was carried out, with the LSTM model proving to be the best performer. Nevertheless, the performance of the DBN and CNN models is appreciable, as they are often not far from the predictions of the LSTM model. These results could be improved by combining time series decomposition techniques, given the strong random variation in wind speed.

References

[1] IRENA, "The Renewable Energy Transition in Africa." Accessed: Jan. 07, 2023. [Online]. Available: https://www.irena.org/Publications/2021/March/The-Renewable-Energy-Transition-in-Africa

[2] A. S. Brouwer, M. Van Den Broek, A. Seebregts, and A. Faaij, "Impacts of large-scale Intermittent Renewable Energy Sources on electricity systems, and how these can be modeled," Renew. Sustain. Energy Rev., vol. 33, pp. 443–466, 2014.

[3] G. Gowrisankaran, S. S. Reynolds, and M. Samano, "Intermittency and the value of renewable energy," J. Polit. Econ., vol. 124, no. 4, pp. 1187–1234, 2016.

[4] S. D. Ahmed, F. S. Al-Ismail, M. Shafiullah, F. A. Al-Sulaiman, and I. M. El-Amin, "Grid integration challenges of wind energy: A review," IEEE Access, vol. 8, pp. 10857–10878, 2020.

[5] Ö. Kiymaz and T. Yavuz, "Wind power electrical systems integration and technical and economic analysis of hybrid wind power plants," in 2016 IEEE International Conference on Renewable Energy Research and Applications (ICRERA), Nov. 2016, pp. 158–163. doi: 10.1109/ICRERA.2016.7884529.

[6] P. Menanteau and C. Clastres, "L'intégration des énergies renouvelables: Le débat est aussi économique," 2009.

[7] S. Dutta et al., "Load and Renewable Energy Forecasting for a Microgrid using Persistence Technique," Energy Procedia, vol. 143, pp. 617–622, Dec. 2017, doi: 10.1016/j.egypro.2017.12.736.

[8] J. Zhang, J. Yan, D. Infield, Y. Liu, and F. Lien, "Short-term forecasting and uncertainty analysis of wind turbine power based on long short-term memory network and Gaussian mixture model," Appl. Energy, vol. 241, pp. 229–244, May 2019, doi: 10.1016/j.apenergy.2019.03.044.

[9] J. Jung and R. P. Broadwater, "Current status and future advances for wind speed and power forecasting," Renew. Sustain. Energy Rev., vol. 31, pp. 762–777, Mar. 2014, doi: 10.1016/j.rser.2013.12.054.

[10] Z. Pang, F. Niu, and Z. O'Neill, "Solar radiation prediction using recurrent neural network and artificial neural network: A case study with comparisons," Renew. Energy, vol. 156, pp. 279–289, Aug. 2020, doi: 10.1016/j.renene.2020.04.042.

[11] Z. Yang and J. Wang, "A hybrid forecasting approach applied in wind speed forecasting based on a data processing strategy and an optimized artificial intelligence algorithm," Energy, vol. 160, pp. 87–100, Oct. 2018, doi: 10.1016/j.energy.2018.07.005.

[12] M. S. Hossain and H. Mahmood, "Short-Term Photovoltaic Power Forecasting Using an LSTM Neural Network and Synthetic Weather Forecast," IEEE Access, vol. 8, pp. 172524–172533, 2020, doi: 10.1109/ACCESS.2020.3024901

[13] "POWER — Data Access Viewer." Accessed: Mar. 21, 2023. [Online]. Available: https://power.larc.nasa.gov/data-access-viewer/

[14] T. O'Malley et al., "KerasTuner." 2019. [Online]. Available: https://github.com/keras-team/keras-tuner

[15] Martín Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." 2015. [Online]. Available: https://www.tensorflow.org/