NEURAL NETWOK MODEL FOR PREDICTING WIND POWER PRODUCTION

Kengjian Lin, Max Thrane Nielsen

(s202541),(s202785)

ABSTRACT

Wind power is a sustainable and renewable energy source that is being adopted at a fast phase as new technologies and growing demand of green energy emerges. With the inclusion of wind power in the electricity mix, grid power management relies on accurate wind power forecasting to keep a grid stable and to minimize the usage of fossil fuel-based resources. Wind power production depends on the local weather conditions, and due to its unpredictable nature, it is challenging to get accurate long term wind power predictions. Neural network based algorithms are applied in state-of-the-art wind prediction models in industry as of today. LSTM (Long-Short Term Memory) network is known as a time series forecasting model that can predict future values based on previous, sequential data. A public wind forecast dataset from a competition on Kaggle is available to be researched. Our work proposes a model that demonstrates the ample use of recurrent neural network architecture. Given a sequence of wind forecast information as input, the model is able to make a long-term univariate prediction. Finally we evaluated the performance of our model using RMSE (root mean square error) and benchmark it against a persistence forecast model. The generalisation score of the model we formulate is able to score 0.1790.

***Index Terms*—** Wind power prediction, Neural network, Recurrent neural network, Long short-term memory

1. INTRODUCTION

Wind power, as a viable renewable energy resource, plays an important role in the green energy industry. However, wind power production is sensitive to the intermittent weather conditions, especially the wind. The stability of a power grid and the price of electricity can also be volatile because of unpredictable wind power supply. The need of wind power prediction arise, and there are many practical uses if we are able to predict wind power production.

With the rapid development of machine learning and neural network technologies, they are becoming more and more popular in being widely used to guide decision making. Therefore we make use of the advances made in the field of deep learning and network based algorithms to design and build a model to predict wind power with given wind forecast data.

In this project, we focus on the wind data from a global energy forecasting competition on Kaggle[1]. The topic for the wind forecasting track is focused on mimicking the operation 48-hour ahead prediction of hourly power production at 7 wind farms, based on historical measurements and additional wind forecast information.

Our goal is to develop models using a neural network, which can forecast the power measurement in a variety of time periods, based on a sequence of wind forecast data. We will evaluate the performance of our models and compared them with a baseline model.

2. THEORETICAL FUNDAMENTALS

LSTM architecture is an extension of the Recurrent Neural Network (RNN) that is suitable for time series data. The LSTM design overcomes some of the vulnerabilities of RNN such as vanishing or exploding gradient by introducing a gated neuron structure called a memory cell. As with RNN it is capable of learning temporal dependencies but it avoids a morphing translation of past memory, making it better at to learning long term dependencies [2]. The gating system manages how the information flows through the layers in the network, and there are three gates: the input gate, the forget gate and the output gate. These gates can be thought as representing the operations associated with a computer cell such as being able to overwrite, delete, update or retain the state of the cell.

The forget gate decides which information from the past time step t_{-1} to erase from the cell state. The input x_t is concatenated with the hidden state h_{t-1} and multiplied with a weight matrix W_f before being processed by a sigmoid function that outputs a value in the range $[0, 1]$ for each value in the previous cell state C_{t-1}. The equation given for the update to the cell state is:

$$f = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$

The input gate decides which values to update whilst the input modulation gate computes candidate values:

$$i = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

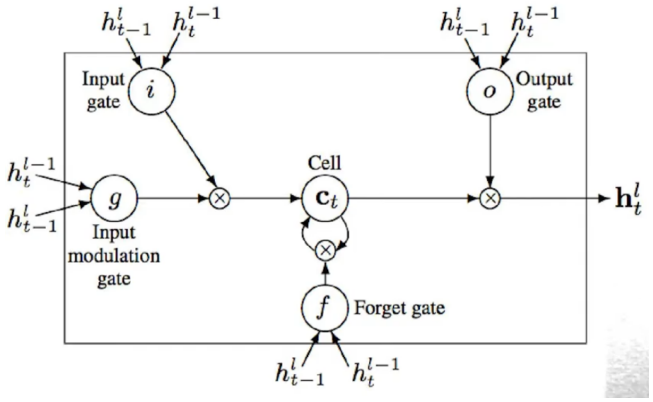

Fig. 1. Illustration of the structure of the memory cell. The illustration is a copy of an image that is shown in one of the video lectures [2].

$$g = \tanh(W_g \cdot [h_{t-1}, x_t] + b_g) \tag{3}$$

We then update the cell state by taking the Hardmard ($\otimes$) product between the old cell state C_{t-1} and f and add it with $i \otimes g$ to get a cell state C_t for the current time step t.

$$C_t = C_{t-1} \otimes f + i \otimes g \tag{4}$$

The output gate controls what information in the cell state we want to output as the hidden state h_t. First we compute the linear map given by the equation:

$$o = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$

o acts like a filter that suppresses or express values of C_t, and the final output that is carried to the next time step is the Hardmard product between 5 and $\tanh(C_t)$.

$$h_t = o \otimes \tanh(C_t) \tag{6}$$

LSTM can be used on different types of predictive tasks such as whole sequence from a sequence which is denoted many-to-many or a value at a time step ahead of the sequence which is denoted many-to-one.

Another variation of RNN is the Gated Recurrent Unit (GRU) which contains similar logic to the LSTM design, but contains fewer parameters and is missing an output gate and cell state. An advantage of using GRU instead of LSTM is that it is less computationally expensive [3].

3. DATA

Table 1 provides an overview of the properties of the attributes.

date	date timestamp, discrete, ordinal
hors	lead time of forecast, discrete, absolute
u	zonal wind component (ms^{-1}), continuous, relative
v	meridional wind component (ms^{-1}), continuous, relative
wd	wind direction (deg), continuous, relative
ws	wind speed (ms^{-1}), continuous, relative
power	hourly generated power (kW/h), continuous, relative

Table 1. Classification of the attribute types in the wind forecast and power measurement data sets.

3.1. Data Visualisation and Mining

The data can be separated into two major parts, the power measurements from 7 wind farms, and wind forecast of every hour for each farm. Each record in the power measurement data is sampled in every one hour, from the 1st of July 2009 at 00:00 to the 26th of June 2012 at 12:00. All the power measurement dataset has been normalized, ranging from 0 to 1.

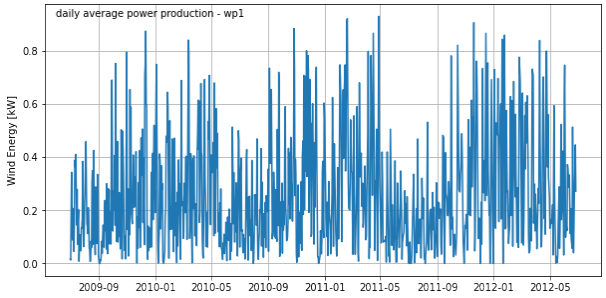

Fig. 2. Daily power measurement visualization

Figure 2 shows the daily average power production of one wind farm during the whole period. From inspecting the graph we discern that the power production from May to September is lowest during a year. We can also see that the power production around November and March is the peak period of the whole year and become volatile during this time. We assume that the power production might have some correlation between the month of the date.

3.2. Data cleansing

The data have been checked for outliers by using two state-of-the-art automatic outlier detection algorithms from the

	wp1	wp2	wp3
count	18757.0	18757.0	18757.0
mean	0.252	0.266	0.335
std	0.246	0.273	0.319
min	0.0	0.0	0.0
25%	0.050	0.042	0.069
50%	0.175	0.164	0.2170
75%	0.381	0.423	0.563
max	0.947	0.989	0.978

	wp4	wp5	wp6	wp7
count	18757.0	18757.0	18757.0	18757.0
mean	0.286	0.273	0.287	0.291
std	0.295	0.294	0.283	0.305
min	0.0	0.0	0.0	0.0
25%	0.022	0.0400	0.053	0.025
50%	0.182	0.157	0.184	0.177
75%	0.474	0.415	0.461	0.505
max	0.992	0.966	0.974	0.960

Table 2. Wind power production time-series description.

sklearn library [4] [5]. Both found 0 anomaly data points in any one of the wind forecast dataset.

3.3. Preprocessing

The wind forecast data is standardized to remove units and re-scale the data s.t. the features contribute approximately proportionally and to improve convergence speed in backpropagation [6].

One feature of the wind forecast data, the wind direction (*wd*), is represented in units of degree. Since angles do not make good model input, we decide to covert the wind direction from angle to radian.

Based on the observation of the seasonal variation in the power production, we introduce a categorical feature *month* in the range from $[1, 12]$ to the wind forecast data. Each input will now also convey information on the month the prediction is being made in. A problem arises from the cyclic nature of the month not being captured in the current representation as there is a discontinuity between 1 and 12. A common suggestion is to make a sinus and cosines transformation of the variable: This gives two new features:

$$\begin{aligned} mnth\_sin &= \sin\left(month - 1 \cdot \frac{2 \cdot \pi}{12}\right) \\ mnth\_cos &= \cos\left(month - 1 \cdot \frac{2 \cdot \pi}{12}\right) \end{aligned} \tag{7}$$

Given a sequence of wind forecast data start from t_i as input, the goal of our model is to make a single prediction n hours in the future, to predict the power measurement y_{i+n+1} at time t_{i+n+1}. In order to build time series dataset for the neural network, we need to segment the raw wind forecast dataset into segments and use a technique called sliding windows. The implementation of sliding windows technique is shown in Figure 3. We put all the wind forecast entries that share the same time of issuing into one segment. When we use sliding windows, we can set the window size to n and step size to 1, to get a sequence of time series data $x_i = [x_{i1}, ..., x_{i(n-1)}]$ and also the corresponding power measurement data y_n. The sliding window size will affect the performance of our model, and it is very important to have a proper sliding window size. The sliding window size should not be too small, otherwise it will undermine the accuracy of the model. A larger sliding window would have the advantage of noise resistant and makes the model more robust. We repeat this process and finally we will get two sequences $X_n = [x_1, ..., x_j]$ and $Y_n = [y_1, ..., y_j]$ as our dataset.

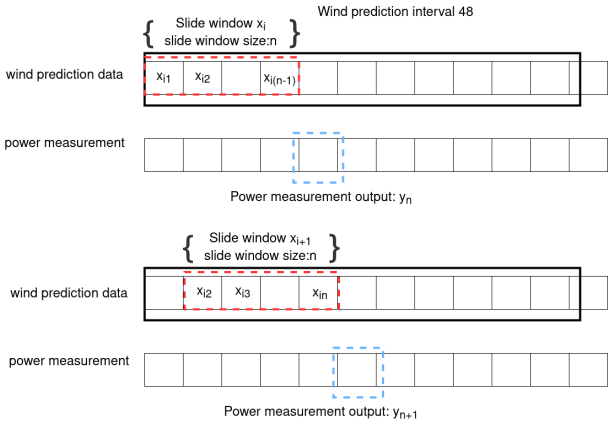

Fig. 3. Slide window

Finally, once we get the input and target dataset, we partition the dataset for cross validation. We select the period between 2009/07/01 and 2010/12/31 as model identification and training period. The remainder of the dataset, is used for evaluating the generalisation error of a model.

4. FRAMEWORK

Given the matrices containing the input sequences $s_{wi} = [x^n_{wj}, .., x^n_{wm}]$ of weather forecast predictions x^n_{wj} of length m where $x^n_{wj} \in \mathbb{R}^n$ with n the number of features, for each wind farm w. The goal is to predict y_{wm+1} at the $m + 1$ time stamp. The proposed model (fig 4) is comprised of a many-to-one LSTM layer that takes a sequence s_{wi} as input and outputs h_{wm}, the vector with the hidden state of the last element in s_{wi}. h_{wm} is propagated to a fully connected layer with a non linear ReLU activation function that outputs a vector with the same dimensions. Then a fully connected layer that maps the image of the ReLU function to the output y_{wm+1}. The procedure is repeated for all wind farms during training, allowing the model to generalise to varying

distributions of wind power productions.

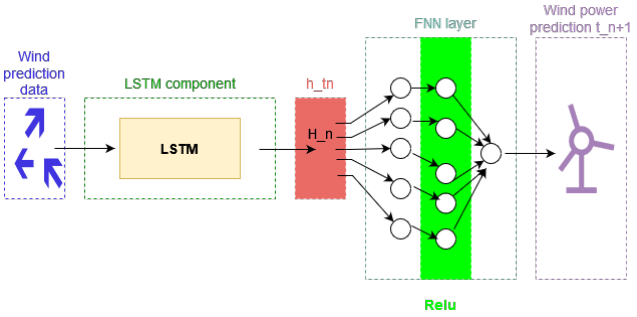

Fig. 4. Illustration of architecture of the model proposed by the authors of this paper for the wind power prediction task. Figure is created with the diagramming tool draw.io.

4.1. Objective Function

The RMSE loss function is the objective for gradient descent during training. The function penalizes large residuals and has been specified as the criterion in the kaggle competition.

$$RMSE = \sqrt{\frac{\sum_i^n (y_i - \hat{y})^2}{N}} \tag{8}$$

5. EXPERIMENTS

5.1. Experimental Setup

The environment for this experiment is conducted on Google Colab, which is a free cloud based Jupyter notebook platform with collaboration functionality.

Table 3 is an overview of the hyper parameters of the experiments:

Network weight initialization	Xavier Initialization
Optimizer	Adam
Learning rate	0.0001
Learning rate decay	0.033
Epoch	100
Batch size	128

Table 3. Hyperparameter

We use Xavier initialization scheme [7] for our neural network model, and the aim of weight initialization is to prevent layer activation outputs from exploding or vanishing during the course of a forward pass through a deep neural network. We choose Adam optimization algorithm to train and optimize our network. In LSTM, the presence of the forget gate will help to solve the problem of gradient vanishing, therefore we simply set the learning rate to 0.0001. We also set the learning rate decay to 0.033.

The goal of the original dataset from the competition is to predict some missing wind power measurements in 48 hour time spans in the period from 2011/1/1 to 2012/6/28. Although we cannot directly acquire the test data from the host of the competition to evaluate our model performance. We decided to extract all the data of the period from the original training data, and turn it into our test set for model evaluation.

Then for model training, we use the simple holdout method and set the training and validation ratio to 0.2. The data between 2009/07/01 and 2010/12/31 is randomly split into training set and evaluation set by percentage, 80% and 20%, respectively. Using a simple holdout method for cross validation allows us to evaluate our model quickly, but it might bring a high variance for our model.

5.2. Results

Table 4 shows the average *RMSE* score of all 7 wind farms by different model configurations. Each column corresponds to one model, and each model is trained for 30 and 100 epochs. The model with the best score is color coded in gray. The test results indicates that the model $M = \{nn = 128, hl = 1, ReLU = true, FCN = 2, CF = true, epochs = 30\}$ performs best with predicting the power production on average. The average score serves as a metric for assessing a models capability to generalise to wind farms situated at different geolocations.

nn	128	64	128	128	64
hl	1	1	1	1	2
ReLU	true	true	false	true	true
FCN	2	2	1	2	2
CF (*)	false	false	false	true	true
epochs					
30	0.1811	0.1812	0.1817	0.1790	0.2908
100	0.1811	0.1811		0.1793	0.2907
PMF	0.4029	0.4029	0.4029	0.4029	0.4029

Table 4. Test results shown for different models tweaking one model parameter at a time to compare the final test score between the models. All experiments are run in 30 and 100 epochs. (*) field indicates whether custom features are used (true) or not (false).

GRU	0.1819	0.1819	0.1855	0.1794	0.2907

Table 5. Test results for replacing RNN unit from LSTM to GRU . Hyperparameters and network configurations stay the same as the table above. The results are generated with 100 epochs training model.

During training we observed that the gradient converges rapidly in the first 5 epochs and then slows down with an order

of 3 magnitudes, indicating that it finds a local minimum quite fast in training (5). This occurs independent of the model parameters, all though the quality of the minimum changes as can be seen from 4. The training and evaluation score did not differ with more than one tenth when we trained for most but all models, thus there is no indication that the model over fits to the training data.

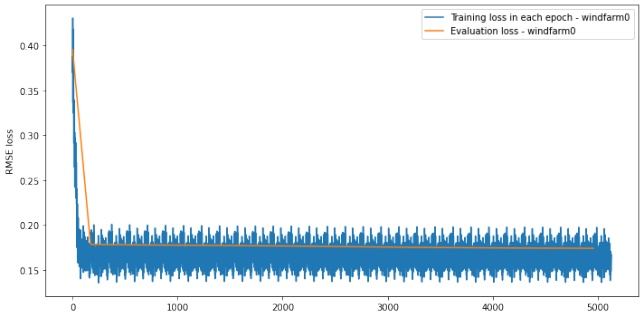

Fig. 5. Training visualisation for wind farm 1. Train error decreases at a rapid rate and then slows down. The same observation can be seen with the validation error, suggesting the model does not overfit to training data.

Weight decay regularization improved over fitting encountered in pretrials, and the range in the learning rate have also been experimented with in order to find the optimal value.

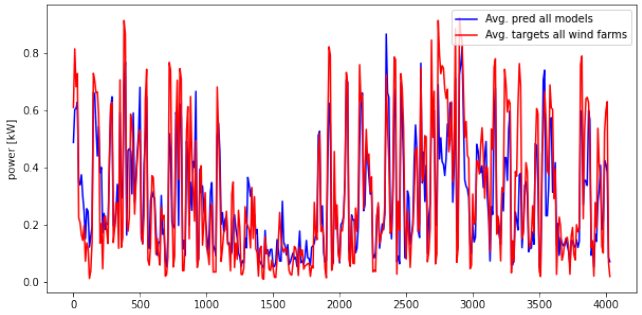

Fig. 6. Line plots showing the average power prediction of all 7 wind farms for all time steps in the test period and the averaged actual target values. The figure visualises the models accuracy.

We compared different RNN models. We replaced the RNN unit from LSTM to GRU, and kept the same hyperparameters and other neural network configurations. Comparing the results between Table 4 and Table 5, LSTM has a slight advantage than GRU, but the difference is almost negligible.

Figure 6 shows a double plot of the average power predictions (blue) and the average target values (red) for all test data. As can be discerned from the plot the model maps to the distribution of the targets fairly accurately.

6. DISCUSSION

The model with the lowest test score would enter the top 50 in the kaggle competition, with 133 team submissions into the competition. This accentuates the potential in deep learning since the model we propose is fairly immature compared to state-of-the-art ensemble models used for forecasting tasks in industry [8].

The preprocessing included feature engineering based on the long-term trend in the power production. We saw that the results improved with the extracted features, but we conjecture that the additionally addition of a time dependent feature *hour of the day*, would help the model capture the short-term daily variation in power production as it have been the choice of other participants [9].

Whilst we check for anomalies in the data as a standard preprocessing step, we didn't examine whether there are abnormal inputs and targets where *ws* is low and *power* high or opposite by defining a cutoff value. Abnormalities like this would confuse the model as the input to output relation is inconsistent.

Our lowest scoring model converges fast to a minimum, and it does not improve the model to apply regularisation techniques s.t. *batchnorm* and *dropout* in the training period. Compared to other studies [3] the chosen number of epochs for training is consistent with their discovery. Thus it can be assumed that in order to improve the test score we would have to apply more preprocessing steps or increase the model complexity to improve further on the test score.

7. CONCLUSION

Wind power is one of the most important renewable power resource for the development of our society. Prediction of the wind power generation has many practical uses and machine learning algorithms are applied to facilitate the need for forecasting energy production. In this research work, a model multilayered deep learning algorithm based on the LSTM architecture is proposed to predict the power measurement of 7 wind farms from time series wind prediction data. The best model of our research can generate data with high accuracy. The forecasting accuracy is evaluated with a Root Mean Square Error (RMSE) criterion, and the lowest generalisation score is 0.179. We have implemented feature engineering during the preprocessing step to help the model learn long-term trends in the power production. Moreover, we also try different hyperparameters and experiments with different neural network architectures and configurations in our model formulation process, in order to understand which model performance is best. Lastly, more optimization techniques are also possible to be applied in our future work, to further reduce prediction error and improve model accuracy.

8. REFERENCES

[1] Kaggle, "Kaggle: GEF2012," .

[2] Ole Winter, "02456 deep learning," 2021.

[3] A.N. Buturache and S. Stancu, "Usage of neural-based predictive modeling and iiot in wind," *Amfiteatru Economic*, vol. 23, pp. 412–428, 2021.

[4] sklearn, "sklearn.neighbors.localoutlierfactor," .

[5] sklearn, "sklearn.ensemble.isolationforest," .

[6] Wikipedia, "Feature scaling," .

[7] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, 2010.

[8] superkailang, "Wpp2012," .

[9] Tao Hong, Pierre Pinson, and Shu Fanc, "Global energy forecasting competition 2012," https://www.sciencedirect.com/science/article/pii/S0169207013000745, June 2014, vol. Volume 30, Issue 2.

9. APPENDIX

https://colab.research.google.com/drive/1J5lqCukAt63Hy9_s5iHtVwxz3zcxxOTu?usp=sharing