WIND POWER FORECASTING: LEVERAGING DEEP LEARNING FOR INTRA-DAY PREDICTIONS WITH APPLICATIONS IN ENERGY TRADING

Boris Guillerey (s192624), Gísli Helgason (s203357), Gonzalo Mazzini (s202683)

DTU Wind Energy

ABSTRACT

In this report a wind power forecasting problem is addressed, namely predicting hourly power generation up to 6 hours ahead using state of the art deep learning methods based on historical measurements and additional wind forecast information from a Numerical Weather Prediction model (NWP). Our approach is to combine a recurrent neural network, namely a long short-term memory model (LSTM), using past power measurements for one wind farm, and a feed forward neural network (FFNN) mapping the weather forecast data to power. This model outperforms a simple but effective persistent baseline and an industry standard time series auto-regressive (AR) model in the range of 1 to 6 hours ahead.

Index Terms— LSTM, FFNN, Wind Power Forecasting, Intra-day energy trading

1. INTRODUCTION

Wind energy is one of the leading sources of renewable energy in the world [1]. Kinetic energy in the wind is converted into electrical energy by wind turbines (WT) according to a power curve [2]. The unstable nature of the wind and complexity of the power curve makes wind power forecasting a challenging but important task.

Intra-day (ID) forecasting is highly relevant to wind farm (WF) operators and energy traders for energy trading in dynamical markets such as Nord Pool's Electricity Balancing Adjustment System (Elbas), where the goal is to balance out production and consumption of electric power. Higher quality forecasts up to 6 hours ahead can be leveraged to improve market decisions and to avoid penalization due to over- or underproduction.

Short-term forecasts (hours to day ahead) of wind power production generally rely on wind speed forecasts from Numerical Weather Prediction (NWP) models (on global or regional scales) that are combined with a model of the wind turbine or farm power curve [2]. Power curves of individual WT are provided by the manufacturer and are characterized by the cubed dependency of aerodynamic power to wind speed, and operational conditions such as cut in, cut out and nominal wind speed. Furthermore, the power curve is affected by additional factors including air density, icing, and degradation of the turbine's blades [2] and electrical and mechanical losses. On a wind farm level the problem becomes dynamic as aerodynamic wake effects (upwind turbines extract energy from the flow) influence the total power output [3].

Time series models are normally used for forecasting based on previous values, when future patterns are expected to be similar to historical patterns. Wind and thus wind power is usually very correlated to its past values for very short lags (minutes to few hours), making time series models based on measured power values a common approach for very short-term wind power forecasting [2]. However the performance generally degrades rapidly further out than 1-3 hours on the forecast horizon, after which NWP models give the most reliable forecast information.

Deep learning models have been found to perform well in short-term and very short-term wind power forecasting, for individual wind turbines and whole wind farms [4][5]. In Deep Learning, the most natural architecture used for sequence problems (like time series) is called Recurrent Neural Network (RNN). The RNNs are basically Artificial Neural Networks (ANNs) with connections (loops) between nodes along a temporal sequence. This loop allows the information to be passed from one step of the network to the next, which is why RNNs deal well with short-term time dependency sequential data. However, the RNNs suffer from vanishing gradient problems, which makes it difficult to learn long-term temporal dependencies. Therefore, another version is needed to consider the information for longer time in advance. LSTMs (Long Short Term Memory) are a special type of recurrent network that are well known for dealing with this problem in Deep Learning.

The goal of this work is to combine both the power measurements of the wind farm and wind speed forecasts using fully data-driven Deep Learning models, seeking a model that can over-perform the two modelling approaches individually in the 1-6 hours ahead forecast. The performance is further compared to a simple persistent baseline[1] and an auto-regressive (AR) time series model. Finally, we show why accurate forecast are of great importance in intra-day energy trading, providing a fictitious but illustrative example

[1]Equals the prediction to the last observed value.

(assuming a simplified market framework) in which the forecast of the model can be used for reducing revenue loss. The project code is available on Github.

2. METHODOLOGY

2.1. Data

A Kaggle dataset [6] containing normalized wind farm power production of seven wind farms and wind predictions (wind vector components giving direction and magnitude) up to 48h ahead from a numerical weather model is considered in this study. The provided data is in hourly granularity. Roughly one and a half year of data is available as shown in Figure 1, with 80% used for training and 20% for testing (using the Hold-out method, see [7]). One year of non-continuous data is intended for testing, but as the dataset belongs to an inactive Kaggle competition this period is of no use in this study.

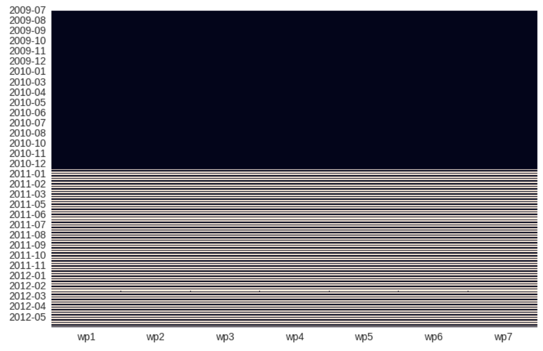

Fig. 1. Data availability (white lines denote missing data).

Power measurements and NWP data corresponding to wind farm one (WF1) is chosen for further analysis (and modelling). The distribution of wind speed forecasts (ID) at the site are shown in Figure 2 along with a Weibull fit (method of moments, see e.g. [8]). The fitted density represents the main power production range (4-12 m/s) quite well.

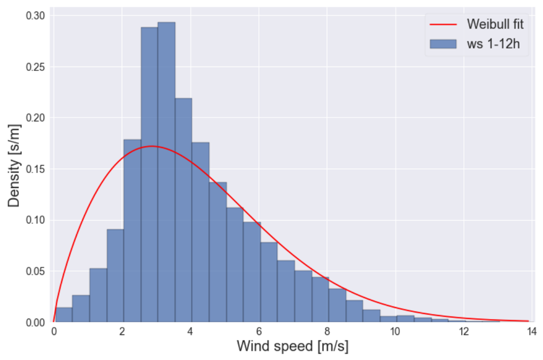

Fig. 2. Distribution of intra-day forecasted wind speed at WF1 along with a Weibull fit.

The mapping between the roughly Weibull distributed wind speed forecast (ID) and power is furthermore quite noisy and non-linear at the site (see Figure 3).

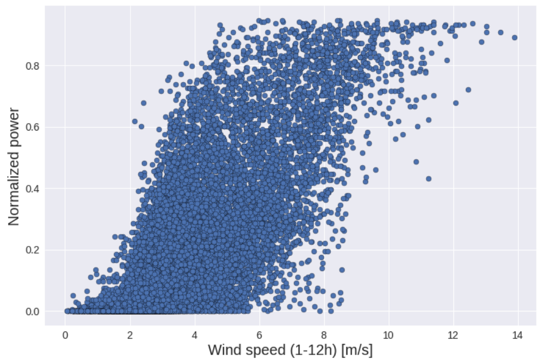

Fig. 3. Normalized wind power measurements against forecasted intra-day wind speed at WF1.

Finally, a window of all wind speed predictions and power measurements is shown for the chosen wind farm (see Figure 4). The measured power generally follows the trend in the wind. Some inertia in the power with respect to forecasted wind speed is however observed. Four predictions of wind speed and direction are available at any given time as the 48h forecast is updated every 12 hours. For our purpose we only use the most recent forecast (1-12h ahead). Therefore, the forecasts from 13 to 48 hours ahead are removed from the data. This leads to a continuous time series of wind speed and wind direction where the predictions are in the best case given 1 hour ahead and in the worst case 12 hours ahead.

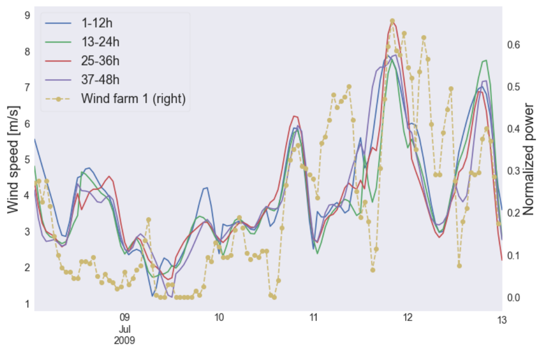

Fig. 4. Wind speed predictions and normalized power production for WF1 during a period of 5 days.

2.2. Time series analysis

The sample autocorrelation function (ρ) of the power measurements at WF1 is computed and illustrated in Figure 5. The autocorrelation decays exponentially, while the partial autocorrelation has three significant lags (see appendices).

This indicates [9] that an AR(3) model (Auto-regressive model of order 3) is a suitable time series forecasting model for wind power at the site, namely

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \phi_3 Y_{t-3} + \epsilon_t \quad (1)$$

where ϕ_i are the model coefficients to be determined and ϵ_t is Gaussian white noise. Statistical analysis (Ljung-Box test) of the model output indicates white noise residuals, but the residual distribution has heavy tails (compared to a Gaussian) likely because the wind power is bounded by 0 and the wind farms rated power.

The integral time scale is given by

$$T = \int_0^\infty \rho(\tau) d\tau. \quad (2)$$

By integrating ρ until the first zero crossing, $T \approx 24\,\text{h}$ is obtained. This scale can be taken as a characteristic time scale of variations in the weather (passing of weather systems) at the site.

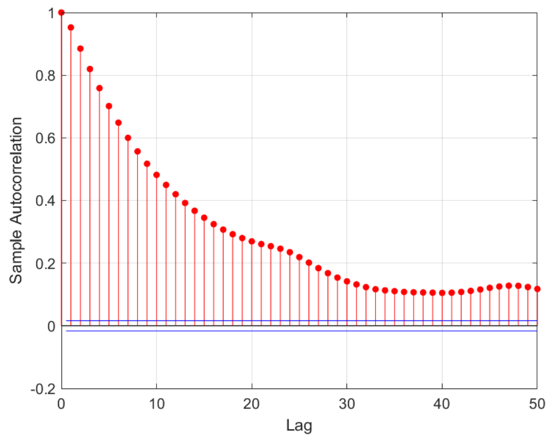

Fig. 5. Sample autocorrelation function of the power measurements at WF1.

2.3. Deep learning model Architecture

To take into account the weather forecast in the Deep Learning model, a classic Feed Forward Neural Network (FFNN) architecture is considered (Model 1, referred to as $M1$), as it is the most natural way of establishing a causal relationship between the variable being forecasted (normalized wind farm power) and other related variables (i.e wind speed and wind direction).

An LSTM model (Model 2; $M2$) is considered for a fully autoregressive deep learning model that only uses past power measurements to predict the power. There are different variants of the LSTM, the one used for this work can be seen in Figure 13, and is explained in subsection 6.2.

Finally, the combination of these two models results in the final model (Model 3; $M3$). For the implementation of $M3$, two more FFNN layers were added, in order to take the output of both models as input for predicting the normalized power. The described model architecture is illustrated in Figure 6

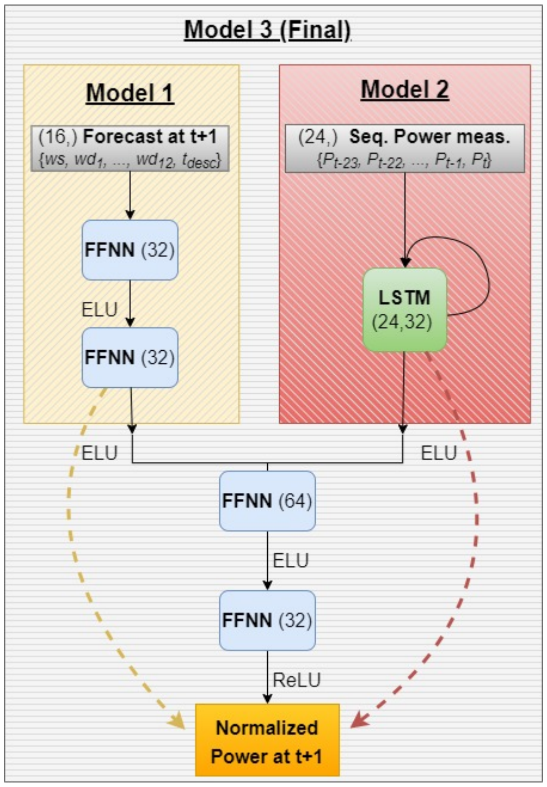

Fig. 6. Model architecture representation.

2.3.1. *FFNN model*

$M1$ is a classical two hidden layers FFNN, with ELU activation functions except for the output layer which uses a RELU activation function (for convenience as normalized targets are between 0 and 1). This FFNN maps from inputs at time t to power output, also at same time t. The prediction of the power is therefore only based on the forecasts, and does not take into account the past time series of power.

Three types of inputs are considered, namely the wind speed, the wind direction and temporal descriptors. The wind speed and wind direction are given by the NWP forecast, only considering the first 12 hours as described above. The wind direction given by the forecast is in degrees, with origin for northern winds and rotating in clockwise direction. This representation introduces a discontinuity at the point where winds come from the North. This problem is overcome by one-hot encoding the wind direction, assuming 12 different sectors separated by 30 degrees. Thus 11 binary variables are obtained instead of one continuous variable initially.

Four temporal descriptors were considered for two kind of variations, namely daily and yearly variations. This way, the model can account for diurnal and seasonal cycles which are predominant in determining the variations in the weather and atmospheric stability (affecting wind shear and wake effects). Cosine and sine descriptors are considered in order to capture not only the amplitude of the variations but also their phase. Hence the mathematical expression for the descriptors is: [10]:

$$\begin{aligned} t_1 = cos(\frac{2\pi h}{24}), \quad t_2 = sin(\frac{2\pi h}{24}) \\ t_3 = cos(\frac{2\pi d}{365}), \quad t_4 = sin(\frac{2\pi d}{365}) \end{aligned} \tag{3}$$

where h is hour of day (varies from 1 to 24) and d is day of year (from 1 to 365). The FFNN then maps from 16 inputs at time t to one output power also at time t.

2.4. Hyper-parameter tuning

One of the drawbacks of an LSTM network in comparison with a simpler time series model is that it requires the selection and optimization of many hyper-parameters, such as number of hidden units, learning rate and batch size, among others. Several methodologies can be used to approach the parameter tuning. Random search or grid search are some of the most used in machine learning parameter tuning. However, these mechanisms require a lot of resources in term of computational capabilities when the model complexity is high. Moreover, there is no guarantee that the parameters in consideration will affect the performance significantly. Hence including a parameter that has minor impact would imply a waste of resources.

The approach for selecting the parameters of $M3$ in this work consisted in performing trial runs by varying one specific parameter from an initial selection, while keeping others constant. It is worth noting that this technique is not optimal for tuning hyper-parameters. However, finding the best parameters is not a main goal for this work. Nevertheless, some parameters tuning is of course needed for the model to perform well enough in a reasonable computational time. Therefore, this section provides a summary of the model sensitivity and different trade-offs observed during the $M3$ training stage.

2.4.1. Batch size and number of Epochs

A batch is a subset of the training set that is used to evaluate the gradient of the loss function and update the weights. Using too large batch size can have a negative effect on the accuracy since it reduces the stochasticity of the gradient descent [11]. On the other side, using small batch size can slow down the training too much, as it takes more time per epoch (pass through the entire training dataset). As an example, training $M3$ with a batch size of 2 took 42 minutes ($\approx$ 100 epochs) to reach the same validation accuracy than using 16 as batch size ($\approx$ 500 epochs) in only 27 minute. Using this criteria for some runs, the final selected batch size was 128 for a training of 2.000 epochs.

2.4.2. Number of hidden units

A well known fact in Deep Learning is the relation between number of hidden units and the capability of over-fitting the training data (while low number of units might result in poor predictions).

For an LSTM layer, the number of hidden units will affect how much information between time-steps is remembered and the capability of learning more or less abstract patterns. The number of hidden units is 32 for both the FFNN and LSTM layers. The performance of the model was not found to be significantly improved by increasing the number of hidden units in the FFNN part (see appendix).

2.4.3. Sequence length

The sequence length for an LSTM network depends on the nature of the data, and is therefore problem specific. Based on the time series analysis, 24 hours of normalized power was selected to predict the power in the next hour (t_{25}). Additionally, one run using 36 hours made it clear that the computational time was significantly increased while model accuracy, evaluated as Root Mean Square Error (RMSE), increased just 0.3 %.

2.4.4. Learning rate and Optimizer

The learning rate and optimizer are placed together as the model performance was barely affected for reasonable ranges of learning rate when using the most common solver algorithms in Deep Learning (RMSProp, Adam, SGDM). Finally a learning rate of 5e-6 and the Adam optimizer were chosen. It is worth noting that higher learning rates were tested but led to undesired oscillations in the the validation loss.

2.5. Market framework

A brief description of the intra-day market based on the danish electricity market system is given here. The intra-day market is the next market to be cleared after the day ahead, where producers and demands submit offers/bids for the 24 hours of the next day (day D). Unlike the day ahead market, which has a specific gate-closure time at noon of the day before (D-1), the intra-day market is continuous, meaning it is cleared every hour by the market operator (*Nord Pool*) in Denmark. Another important difference is the market clearing mechanism: while the day-ahead clearing price is based on the intersection of supply/demand curves, the intra-day is simply based on financial contracts to be agreed bilaterally between each pair of buyers/sellers. For several reasons, the power buyers and sellers may wish to modify the day-ahead financial contracts.

For instance, market participants may get an updated forecast of future wind power, which might differ from the original in the day-ahead stage. The intra-day market provides the opportunity to offset such a deviation. In general, any mismatch between forecast and real production is undesired, though the associated cost depends on many other variables that are outside the scope of this report (i.e if the power imbalance requires upward or downward regulation).

An illustrative example can be provided assuming a simplified penalty framework with the following (realistic) values[2]:

- 3 months operation of a 100 MW wind farm.
- an energy price of 140 €/MWh;
- mismatches due to over-and underproduction are equally penalized (π_-=π_+=30 €/MWh).

The total revenue in € for a period can be then computed as

$$R_{\text{total}} = \sum_t R_t = \sum_t \left(E_t \cdot 140 - |\hat{E}_t - E_t| \cdot 30 \right) \qquad (4)$$

where R_t is the revenue for one hour (t), $\hat{E}_t$ is the forecast used for intra-day trading and E_t is the real produced power for hour t. This expression is used together with the model forecast to obtain illustrative results.

3. RESULTS AND DISCUSSIONS

3.1. Model interpretation

The FFNN mapping is as a model of the wind farm power curve. The obtained mapping is clearly not linear and has some directional dependencies, as seen in Figure 7. It is not entirely realistic as the forecasted power should not exceed the rated power of the wind farm (could be addressed with sigmoid activation function). More meteorological parameters from the NWP forecast model would likely be beneficial as the power curve is known to also depend on atmospheric stratification (wind shear and wake effects) and air density.

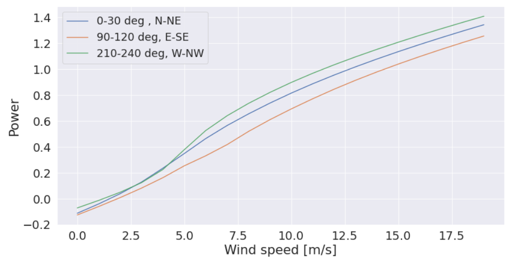

Fig. 7. Modelled (FFNN) normalized power curves for three chosen directional sectors at WF1.

[2]Capacity of 100 MW is chosen to represent a standard Danish wind farm; energy price corresponds to the Nord Pool market in November 2021 and the penalty values match the same fraction of the electricity price as the annual average presented in [12] for the Dutch electricity market in 2002.

3.2. Model Comparison

In this section, the model performances are compared considering the RMSE as metric. As shown in Figure 8, the RMSE is computed for three fully autoregressive models (Persistent, AR(3) and $M2$), one FFNN model ($M1$) and the final combined model ($M3$). First, we observe that the three fully autoregressive models perform almost equally along the six hours, with $M2$ performing slightly better than the Persistent and AR(3) performing slightly better than both models. For one hour ahead, $M3$ performs also similarly as the three autoregressive models. It can be argued that for one hour ahead the FFNN part of $M3$ based on meteorological forecast does not increase significantly the performance and that the power forecast is mostly based on past power information. However, as the horizon increases, $M3$ performs progressively better than the purely autoregressive models as the information provided by the meteorological forecast becomes more relevant than past measurements. Finally, it is worth evaluating more closely the role of the deterministic FFNN part of the model by inspecting the performance of $M1$ (purely FFNN). In this model no distinction is made on the horizon, it is trained on mapping the meteorological data to power for all forecasts 1 to 12 hours ahead. Therefore, this model is time independent[3] and yields the same performance at every horizon. For a short horizon (below 4 hours), the autoregressive models perform better than $M1$. In other words, within 4 hours, the past values of power are more descriptive of future values than the meteorological data. After 4 hours however, the pattern is reversed and meteorological data performs better. As $M3$ includes both type of data, it consistently yields better results than the other models. For higher horizons, as the value of past power data becomes less relevant, the performance of $M3$ converges towards the performance of a purely deterministic mapping (FFNN).

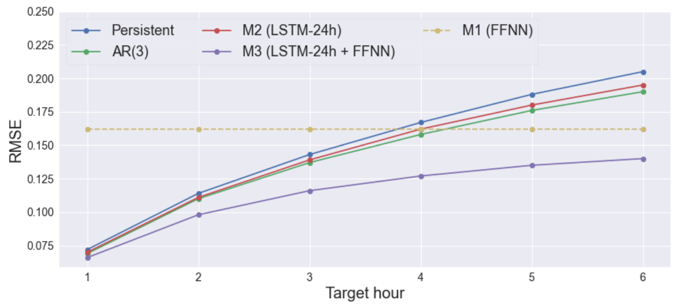

Fig. 8. Comparison of models based on RMSE for 1-6 hours ahead.

[3]NWP forecast 12h ahead is however more uncertain and less accurate than 1h ahead.

3.3. Best Model ($M3$) performance

In Figure 9 the $M3$ predictions versus targets (validation set), together with a linear fit (and corresponding R^2) is presented to show how the performance degrades from 1 to 6 hours ahead.

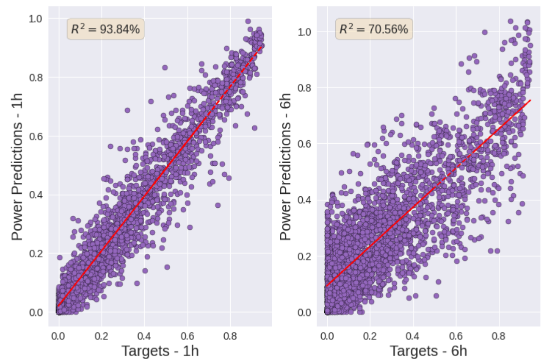

Fig. 9. $M3$ Power predictions 1 and 6 hours ahead vs. target.

Figure 10 shows the prediction and target power for a period of six hours (again, from test set), where confidence levels (Gaussian) are included. As expected, the confidence interval widens with time. The main purpose is to illustrate how the model is able to capture the trend for the 6-hour period, including the power reduction at the last hour.

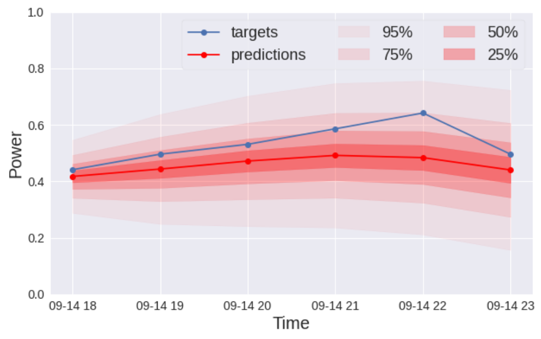

Fig. 10. M3 predictions on a horizon of 6 hours with prediction intervals.

3.4. Market analysis

In Figure 11 the revenue loss of the best model ($M3$) and AR(3) are compared, assuming that the 1, 3 and 6 hours ahead forecast are used when computing the revenue. The left axis corresponds to the loss of revenue (in Millons of €). The right axis is the ratio of optimal revenue (commonly used in literature [13]) which is the percentage of the loss with respect to the perfect forecast. At 1 hour ahead the ratio of loss to optimal revenue is 4.8% for the AR model and 3.4% for $M3$, while at at 6 hour ahead the ratio is roughly 12% and 8% respectively. For all 3 horizons, the loss is reduced by about one third by applying $M3$ instead of AR(3).

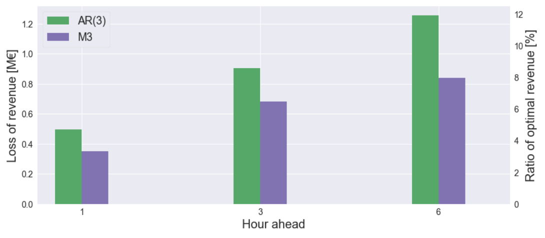

Fig. 11. Revenue loss comparison between AR(3) and final model (M3).

4. CONCLUSIONS AND FURTHER WORK

A deep learning model has been developed to produce more accurate forecasts up to 6 hours ahead (relevant time window for intra-day trading). As a result, **the combination of the Power history (LSTM) and NWP forecast (FFNN)** consistently over-performed other models (1-6 h ahead): the final model ($M3$), yielded a **RMSE reduction** of 4.2% at 1 hour ahead, 19.6% at 4 hours and 13.6% at 6 hours, in comparison with the next best model at each of those horizons.

Finding the optimal hyper-parameters for improving the model performance as much as possible is outside the scope of this work. Furthermore, AR(3) and Persistent that were used for benchmarking are simpler models in terms of interpretability and architecture compared to $M3$. Therefore, it was desired for fair comparison to show with few trial runs that adequate parameters values can be found.

Through this report it has been demonstrated the capabilities of Deep Learning for power forecasting. Deep Learning allows for an architecture which takes advantage of all the available data that a wind farm operator would normally have with just some relative simple feature engineering (i.e. one hot encoding wind direction and using the temporal descriptors).

Other topics were also considered in the making of this work, but were not addressed due to limited time. Namely Day-Ahead forecasting, Transfer learning and statistical performance testing. Firstly, using a similar structure as $M3$, i.e. combining the FFNN mapping and a LSTM based on time series of predicted wind speeds, can possibly yield good results. Secondly, using pre-trained models (such as $M3$ for WF1) via transfer learning is relevant to forecast providers as new wind farms have little to no memory (training data). The dataset used in this study includes six other wind farms (not considered here) that potentially could be used in transfer learning. For stronger results, these wind farms should also be considered in the model performance evaluation along with statistical performance testing.

5. REFERENCES

[1] GWEC, "Global wind report," 2021.

[2] Fotios Petropoulos et al., "Forecasting: theory and practice," 2021.

[3] Tuhfe Göçmen, Paul van der Laan, Pierre-Elouan Réthoré, Alfredo Peña Diaz, Gunner Chr. Larsen, and Søren Ott, "Wind turbine wake models developed at the technical university of denmark: A review," *Renewable and Sustainable Energy Reviews*, vol. 60, pp. 752–769, 2016.

[4] Erick López, Carlos Valle, Héctor Allende-Cid, and Héctor Allende, "Comparison of recurrent neural networks for wind power forecasting," in *Pattern Recognition*, Cham, 2020, pp. 25–34, Springer International Publishing.

[5] Q. Zhu, H. Li, Z. Wang, J. Chen, and B. Wang, "Short-term wind power forecasting based on lstm," *Dianwang Jishu/Power System Technology*, vol. 41, pp. 3797–3802, 12 2017.

[6] Kaggle, "Global Energy Forecasting Competition 2012 - Wind Forecasting," `https://www.kaggle.com/c/GEF2012-wind-forecasting/data`, 2014.

[7] T. Herlau, M. N. Schmidt, and M. Mørup, *Introduction to Machine Learning and Data Mining*, 2020.

[8] M. Kelly, J. Berg, J. Mann, and M. Nielsen, *Micrometeorology for Wind Energy*, 2020.

[9] Henrik Madsen, *Time Series Analysis*, 10 2008.

[10] F. Zanetta and D. Nerini, "Nowcasting of surface wind speed using probabilistic, explainable deep learning," *EMS Annual Meeting 2021, online*, 6–10 Sep 2021.

[11] Daniel Holden, "My Neural Network isn't working! What should I do?," `https://theorangeduck.com/page/neural-network-not-working`, 2017.

[12] Pierre Pinson, Christophe Chevallier, and George Kariniotakis, "Trading wind generation from short-term probabilistic forecasts of wind power," *Power Systems, IEEE Transactions on*, vol. 22, pp. 1148 – 1156, 09 2007.

[13] Pierre Pinson, Christophe Chevallier, and George Kariniotakis, "Trading wind generation from short-term probabilistic forecasts of wind power," *Power Systems, IEEE Transactions on*, vol. 22, pp. 1148 – 1156, 09 2007.

[14] Jennifer J. Gago, Valentina Vasco, Bartek Lukawski, U. Pattacini, Vadim Tikhanoff, Juan G. Victores, and Carlos Balaguer, "Sequence-to-sequence natural language to humanoid robot sign language," *ArXiv*, vol. abs/1907.04198, 2019.

6. APPENDIX

6.1. Time series analysis

The sample partial autocorrelation is shown in Figure 12. Three significant lags determine the order of the AR model considered in the study (subsection 2.2).

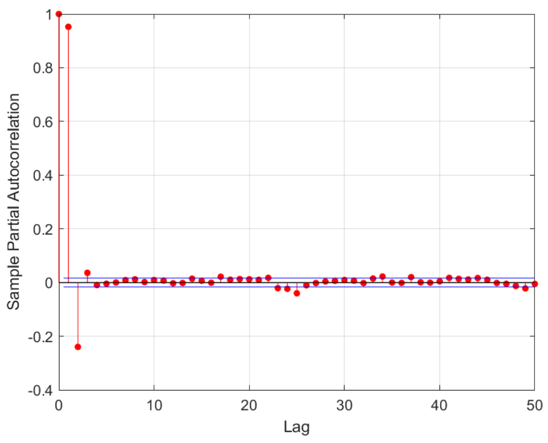

Fig. 12. Sample partial autocorrelation function of the power measurements at WF1.

6.2. LSTM implementation

From the different variants of the LSTM, the one implemented in *PyTorch* was used. The LSTM module can be explained by two main components: the cell state (top horizontal line) and the hidden state (bottom horizontal line). The cell state is responsible for the long term of the model. The information in the cell state is controlled by the so called gates. First, the **forget gate** (f) decides how much information will be thrown away from the cell state of the previous module (c_{t-1}). The next step is to decide which new information will be stored in the cell state and to update it, which is performed by both the **input gate** (i) and **cell candidate** (g). On the other hand, the hidden state is the output value of the module. The **output gate** (o) decides what is the output based on the updated cell state (c_t), the hidden state from the previous module (h_{t-1}) and the input channels.

The mathematical expressions used in *PyTorch* implementation are:

$$\begin{aligned}
&f = \sigma\left(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}\right)\\
&i = \sigma\left(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}\right)\\
&g = \tanh\left(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}\right)\\
&c_t = f * c_{t-1} + i * g\\
&o = \sigma\left(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}\right)\\
&h_t = o * \tanh\left(c_t\right)
\end{aligned} \tag{5}$$

where σ is sigmoid function and $*$ Hadamard product.

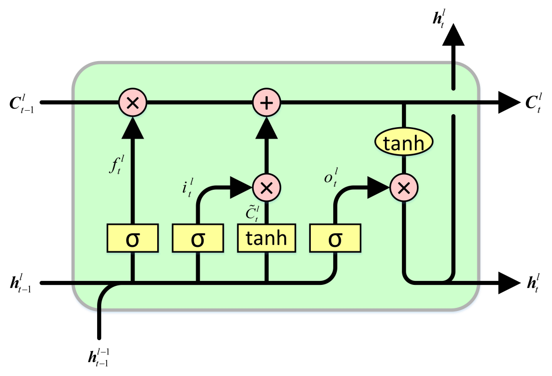

Fig. 13. LSTM cell [14].

6.3. Number of hidden units

The number of hidden units in the layers of the FFNN part of model 3 was determined by a series of trial runs starting with 32 hidden units in both layers and then increasing this number with different configurations. The results in term of RMSE can be observed in Table 1.

Layer 1 hidden units	Layer 2 hidden units	RMSE
32	32	0.06801
512	32	0.06805
256	256	0.06808
32	512	0.06887

Table 1. Comparison of RMSE obtained for different number of hidden units in the FFNN part of model 3.

The performance was not improved by increasing the number of hidden units. Therefore the initial configuration comprising 32 hidden units in both layers was retained for the final model.