IT UNIVERSITY OF COPENHAGEN

Evaluating The Trade-Off Between Performance And Carbon Emission In Wind Power Forecasting

Frederik Bechmann Faarup

Research Project
Computer Science
Frederik Bechmann Faarup

November 20, 2024
Supervisor: Maria Sinziiana Astefanoaei

CONTENTS

1 Introduction 1
2 Background 2
2.1 Forecasting hierarchical time series . . . 2
2.2 ARIMA . . . 3
2.3 BERT . . . 3
2.4 GNN . . . 3
2.5 Carbontracker . . . 4
2.5.1 Alternatives to Carbontracker . . . 4
3 Data and Material 5
3.1 SDWPF: KDD Cup 2022 Dataset . . . 5
3.2 Official Train And Test Set . . . 5
3.3 Data Analysis . . . 6
3.4 Data Processing . . . 7
4 Method 13
4.1 Model Architectures and Training . . . 13
4.1.1 Baseline ARIMA . . . 13
4.1.2 BERT model (KDD Cup 3rd place) . . . 13
4.1.3 GNN model (KDD Cup 11th place) . . . 14
4.2 Evaluation . . . 15
5 Results & Analysis 16
6 Conclusion 17

1 INTRODUCTION

Wind power takes the storm. It is a part of the solution choices when it comes to converting our energy usage to sustainable sources. In Denmark, more than 40% of the total energy production comes from wind power[1], and the number is growing. However, wind power relies heavily on the weather conditions, resulting in a high variation in the production and reliability of the energy source. Accurate wind power forecasting can secure a smoother operation on the grid system, help plan unit commitment and maintenance and further reduce the need for additional balancing energy sources.

This problem coupled with the advances in deep neural networks has given rise to new methods of wind power forecasting, such as graph neural networks (e.g. Wu et al. (2020)), that utilise the spatial data of wind turbines. While such methods may be effective - and they are, according to the wind power forecasting competitions[2] - they also require more computational power than previous statistical models, like linear regression or ARIMA. A bigger computational need of the models will inevitably lead to a higher carbon emission.

Strubell et al. (2019) analysed the carbon emissions of state of the art deep learning models in the field of natural language processing and found that training the transformer-based BERT model with 110 million parameters for 96 hours resulted in 652.3 kg carbon dioxide emitted. Given this, we must ask ourselves if similar unwanted effects occur when forecasting wind power. With that, this project seeks to answer the following questions: **What is the trade-off between model complexity and prediction accuracy of wind power forecasting models? And what are the effects in carbon emission?**

To answer the question[3], the tool Carbontracker (Anthony et al. (2020)) was used on the training process of three wind power forecasting methods with different magnitudes of parameter space. The methods includes a baseline ARIMA model and two deep neural networks from the KDD CUP 2022 (Zhou et al. (2022).

[1] https://ens.dk/ansvarsomraader/vindmoeller-paa-hav/fakta-om-vindenergi-paa-hav
[2] https://aistudio.baidu.com/competition/detail/152/0/introduction
[3] Github repository associated with project: https://github.com/frfa1/research_project

2 BACKGROUND

This section describes the general theory of the wind power forecasting models evaluated, including a baseline ARIMA, a BERT architecture and a GNN arctitecture. It further outlines details about the methods used to track the carbon emission of the models. Section 4.1 in methods goes into more detail about the exact architecture, training setup and hyperparameters of the models.

2.1 FORECASTING HIERARCHICAL TIME SERIES

Wind power forecasting can be viewed as a hierarchical structure with the total power generated at the top, individual wind turbines at the bottom and wind farms or other types of groups in-between. The structure is arbitrary and can be made to fit different examples.

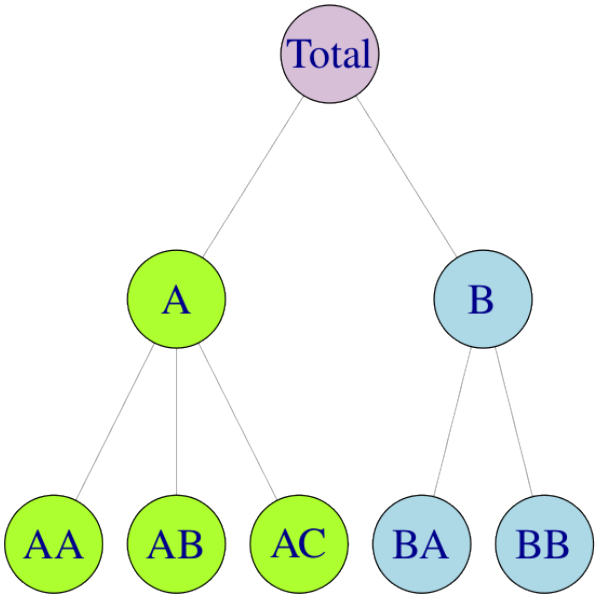

Figure 1: Example of hierarchical structure in time-series forecasting. Image is taken from Hyndman and Athanasopoulos (2018)

For any timestep t, the observations at the top will be the sum of the observations in the level beneath it. For example:

$$y_{Total} = y_A + y_B$$

$$y_A = y_{AA} + y_{AB} + y_{AC}$$

$$y_B = y_{BA} + y_{BB}$$

There are different methods of forecast reconciliation when dealing with multiple models. Examples of simple methods are:

- **Top-down**: Forecasts at bottom level and sum up to reach a higher level in the tree
- **Bottom-up**: Firstly generates a forecast for the total and then disaggregates. Different methods can find the proportions to split into each disaggregate

- **Middle-out**: A middle-level is chosen and forecasts are made. For levels above bottom-up is used and for levels below top-down is used

English and Abolghasemi (2023) exemplify the use of hierarchical structure in wind power forecasting with reconciliation. They apply traditional methods of linear regression and gradient boosting in a wind power forecasting scenario to compare different methods of reconciliation. They test different methods of *cross-temporal forecasting*: An idea that combines cross-sectional hierarchies with cross-temporal hierarchies. The former means logical or geographical hierarchies, like wind turbines in wind power farms. The latter means aggregating different frequencies of data, like 10-minutely, hourly and daily.

As we will see shortly, the dataset in this project considers a single wind farm with 134 wind turbines, which can be considered a hierarchical structure. The following introduces the three models.

2.2 ARIMA

We begin with a baseline model. An ARIMA (Autoregressive integrated moving average) model was used as to represent a traditional method with less parameters in contrast to modern deep learning architectures.

ARIMA(p,d,q) contains three parameters (or hyperparameters). Firstly p, which is the number of autoregressive terms. It represents the number of lag values used in the model. The parameter d controls the number of differences for the series to become stationary. It is used to make time-series with trends or seasonality stationary such that short-term predictions can be made. Lastly, the parameter q is the window size of the moving average. In total, the hyperparameters of ARIMA models make up $m = p + q + 1$ parameters that are estimated on the training data, usually through maximum likelihood estimation.

$$y'_t = c + \phi_1 y'_{t-1} + ... + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q} + \epsilon_t$$

2.3 BERT

The first model we consider from the KDD CUP 2022 is a BERT (Devlin et al. (2019)) model. We consider this model because it achieved an impressive third place in the competition, and because it is a type of deep learning model that generally has a big number of parameters.

BERT is one of the state of the art models for language models, and it is based on transformers and the attention-mechanism. Not much literature covers its use in wind power forecasting, however, Tan and Yue (2022) use it in their solution to the KDD CUP 2022. They argue that it can handle long dependencies for long sequence predictions, which is a typical property of the attention-mechanism (Vaswani et al. (2017)) in the transformer architecture. More details on their architecture and training can be found in section 4.1.2.

2.4 GNN

The next model we consider is a Graph Neural Networks (GNNs). Serveral GNNs achieved great results at KDD CUP 2022, however, the first with reproducible code was found at 11th place.

GNNs are a generalization of Convolutional Neural Networks (CNNs) where the assumptions of a grid-like structure are dropped. CNNs contain filter opera-

tions of a fixed size that combine an element with its neighbours by taking their dot product. Similarly, GNNs use a reference operator that has the sparsity pattern as the adjacency matrix A, for example the Laplacian: $L = D - A$ or the normalized adjacency matrix: $A_n = D^{-1/2}AD^{-1/2}$. Reference operators have the property that when multiplied by a graph signal will compute a weighted sum of each node's neighbourhood. This allows GNNs to understand relationships between neighbouring nodes - which is useful in wind power forecasting as neighbouring wind turbines can be represented as a graph. Jiang et al. (2022) use GNNs in their solution to the KDD CUP 2022, and more about their architecture and training can be found in 4.1.3.

2.5 CARBONTRACKER

A key part of this project is measuring the carbon emission of the three models. This is done with the Carbontracker toolkit, which was introduced by Anthony et al. (2020). It measures the total power of components such as GPU, CPU and DRAM during training of deep learning models. They argue that total power consumption is a fair measurement as the user should be accountable for the static power used in the time that the resource is reserved. It is also more pragmatic to measure compared to dynamic power consumption.

To account for the power consumption of the supporting infrastructure, Carbontracker multiplies the measured power by Power Usage Effectiveness (PUE) – a measurement of the energy efficiency of the data center that handles the computation.

$$\text{PUE} = \frac{\text{Total Facility Energy}}{\text{IT Equipment Energy}}$$

Carbontracker uses the 2022 global average PUE of 1.55.

Beside measuring the the power consumption of the components during training time, Carbontracker also fetches carbon intensities every 900 seconds, which together with the power consumption make up the carbon footprint of training a deep learning model:

$$\text{Carbon Footprint} = \text{Power Consumption} \cdot \text{Carbon Intensity}$$

The carbon intensities are based on the IP Address of the compute, which is found by the Python library `geocoder`. The carbon intensities are limited to Denmark and Great Britain due to a lack of a globally accurate, free, publicly available real-time carbon intensity database. For Denmark, the API used is Energi Data Service[1], which is relevant for this research paper. For the remaining countries, the average carbon intensity of EU-28 countries is used.

2.5.1 Alternatives to Carbontracker

Code Carbon and Eco2AI are other Python libraries that similarly to Carbontracker measure the power consumptions of the hardware during execution of code and then translates it carbon emissions. However, they do not call APIs to get real-time data, and therefore Carbontracker was preferred.

[1] https://www.energidataservice.dk/

3 DATA AND MATERIAL

3.1 SDWPF: KDD CUP 2022 DATASET

This research paper focuses on the SDWPF dataset made by Zhou et al. (2022) for the Spatial Dynamic Wind Power Forecasting Challenge at KDD Cup 2022.

The dataset contains real-world data from Longyuan Power Group Corp. Ltd. (the largest wind power producer in China and Asia). It has data of 134 wind turbines from one wind turbine farm during half a year. It includes relative positions and internal statuses of wind turbines. Table 1 shows and overview of the dataset, and table 2 shows the columns (features) of the dataset.

Days	Interval	# of columns	# of turbines	# of records
245	10 minutes	13	134	4,727,520

Table 1: SDWPF dataset overview.

Column Name	Specification
TurbID	Wind turbine ID
Day	Day of the record
Tmstamp	Created time of the record
Wspd (m/s)	Wind speed (recorded by anemometer)
Wdir (°)	The angle between the wind direction and the position of turbine nacelle
Etmp (°C)	Temperature of the surrounding environment
Itmp (°C)	Temperature inside the turbine nacelle
Ndir (°)	Nacelle direction (the yaw angle of the nacelle)
Pab1 (°)	Pitch angle of blade 1
Pab2 (°)	Pitch angle of blade 2
Pab3 (°)	Pitch angle of blade 3
Prtv (kW)	Reactive power
Patv (kW)	Active power (target variable)

Table 2: Columns of the SDWPF dataset and their meanings.

The dataset contains a big proportion of missing or abnormal values that happen due to a variaty of reasons like errors in the tracking technology at the wind turbine. Table 3 shows all the known abnormal conditions in the dataset.

3.2 OFFICIAL TRAIN AND TEST SET

The KDD CUP 2022 dataset is split into an official train and test set, where the latter was not given to the participants before after the competition was held. This means that the participants have made their own train-validation splits to validate their model scores before submitting their best models. The official train set consists of the 245 days as seen in table 1, while the test set consists of 142 different input files

Condition
Power generated < 0
Wind speed < 1 and power generated > 10
Wind speed < 2 and power generated > 100
Wind speed < 3 and power generated > 200
Wind speed < 2.5 and power generated == 0
Wind speed == 0 and wind direction == 0 and external temperature == 0
External temperature < -21
Internal temperature < -21
External temperature > 60
Internal temperature > 70
Wind direction > 180 or wind direction < -180
Nacelle direction > 720 or nacelle direction < -720
Pitch angle of any blade > 89

Table 3: List of conditions that make data instances abnormal.

of 14 days (the predictors) and 142 output files of two days (the response). Both files contain all the variables seen in table 2, and the goal of the models in the KDD CUP was to take in up to 14 days of 10-minutely input and predict the power generated of the preceeding two days of data (288 10-minutely timesteps).

3.3 DATA ANALYSIS

The following section analyses the official training set of the SDWPF dataset. An overview of the distribution of each variable in the dataset can be found in table 2. The distribution of wind speeds resembles a log-normal distribution with most values between 0 and 5 m/s and a long tail of rarer extreme winds. The wind direction (relative to the nacelle position) seems normally distributed around 0 degrees. Interestingly, the power generated (Patv) resembles a power curve with most values at 0 or just below 1600 kW, indicating the limits of the wind turbines. This is more clear in figure 3 where the wind speed and the power generated of two wind turbines are scattered. The scatter plot shows how the power generated of the turbines increases as the wind speed increase, and moreover, that this increase follows a power curve with a hard maximum reflecting the physical limits of the wind turbines. The figure further illustrates the presence of outliers in the data set, which here are colored if they match any of the conditions in 3.1.

Figure 4 shows the mean power generated of the ten least and the ten most productive wind turbines. The top ten turbines each produce on average more than 400 kW per time step while the bottom ten each produce less than 300 kWh per time step, indicating that there is a difference in productivity of the wind turbines. Figure 5 further shows the difference in mean power produced for all the wind turbine, including their spatial distribution. The wind turbines are placed in rows with around 1 kilometer apart, and the turbines in the forth row from left seemingly produce the most power. Whether this difference is caused by the placement of the wind turbines (relative to the wind), the technology in each wind turbine, or something else, is unclear.

Figure 6 shows the daily average power and wind speed of turbine 1. The figure gives and indication that there is no seasonality in the data as the days vary equally.

Lastly, an Autocorrelation Function (ACF) plot is made in figure 7, and a Partial ACF is made in figure 8. The ACF plot shows how the target variable, power generation, is correlated with itself in previous time steps. It is commonly used to detect stationarity in time series - i.e. a time series with no long-term, predictable patterns. As the plot decreases slowly in values outside of the 95% confidence interval, we can conclude that the variable has a predictable pattern.

3.4 DATA PROCESSING

The two models from the KDD CUP 2022 evaluated in this report have each their respective preprocessing and are not commented here. This section focuses on the preprocessing used in the baseline ARIMA model.

The data points containing abnormal conditions seen in figure 3 and table 3 are all replaced by `NaN`-values. After that, a K-nearest neighbour algorithm was used to find the 10 spatially closest wind turbine of each wind turbine. The number 10 was chosen to ensure that each neighborhood contains more than one "row" of the wind turbine grid seen in figure 5. For each timestep and for each wind turbine, the `NaN` of that particular wind turbine is replaced with the mean of its nearest neighbours in the same timestep. For the rest of the `NaN`-values, the most preceding or succeeding value for the same wind turbine was used.

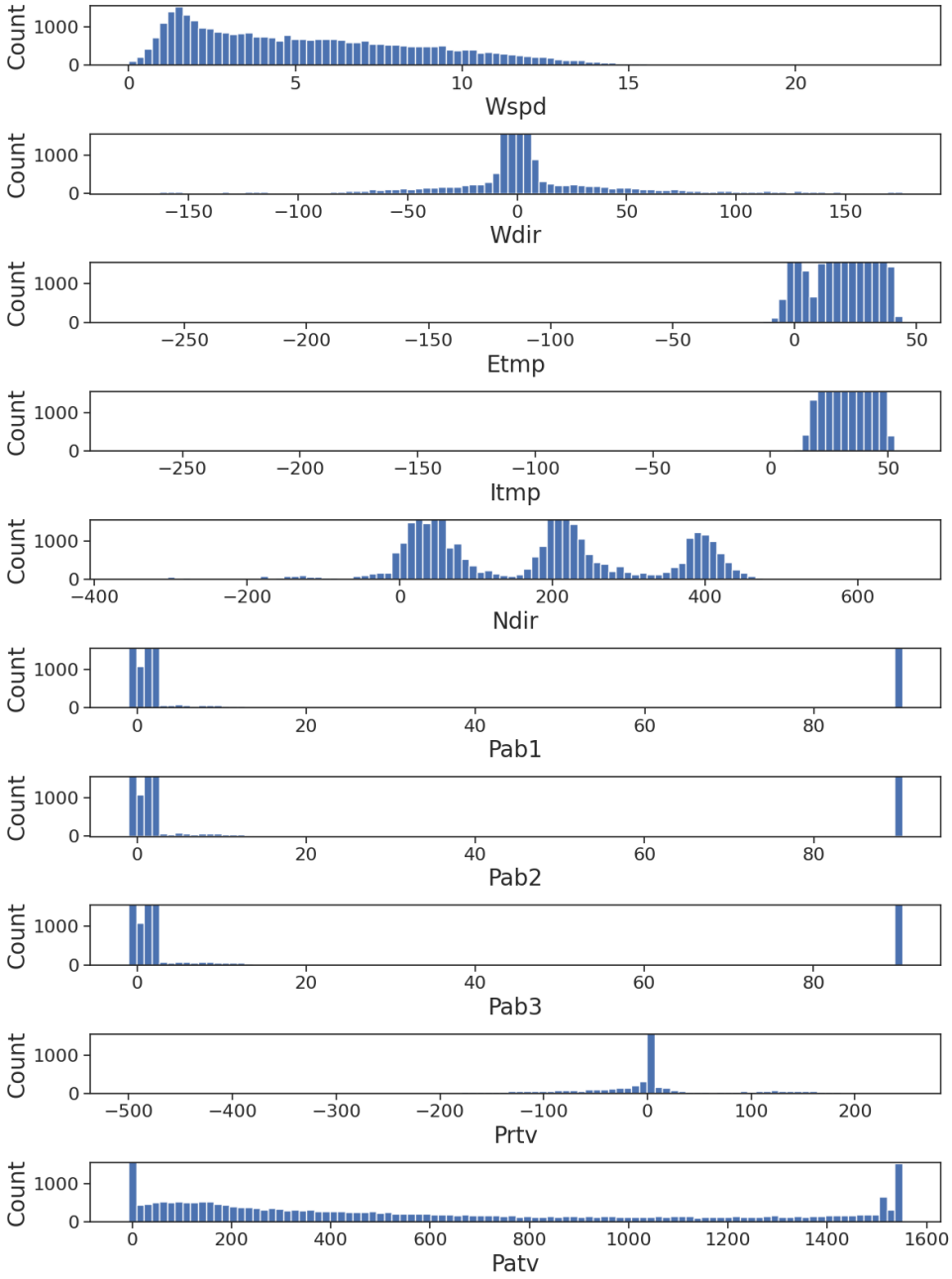

Figure 2: Histogram distributions of each variable in the data set. The metric of each variable is the same as in table 2, for example, Wind Speed is in meter per second

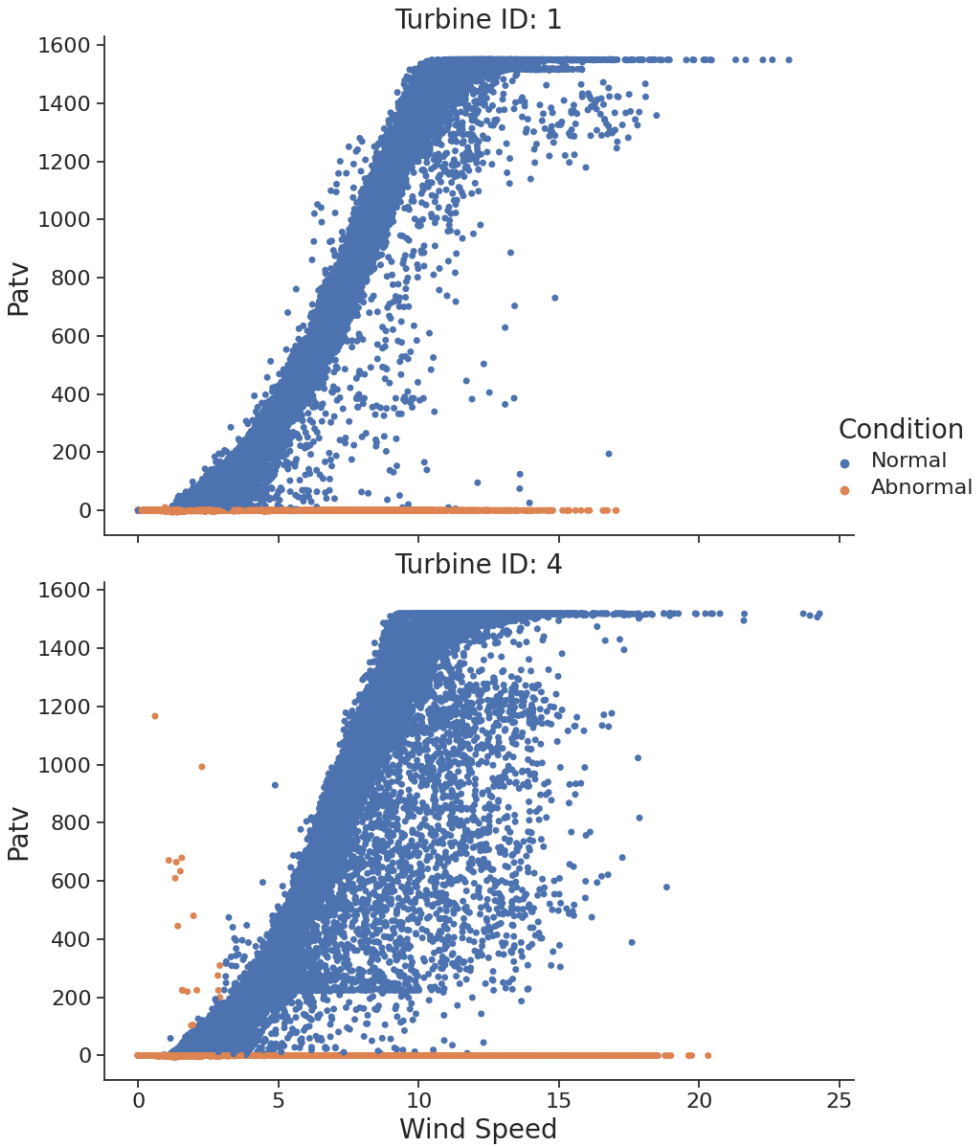

Figure 3: Scatter plots of wind speed (x-axis) and power generated (y-axis) for 2 arbitrary wind turbines. Points are labeled abnormal if they fit one of the conditions described in section 3.1

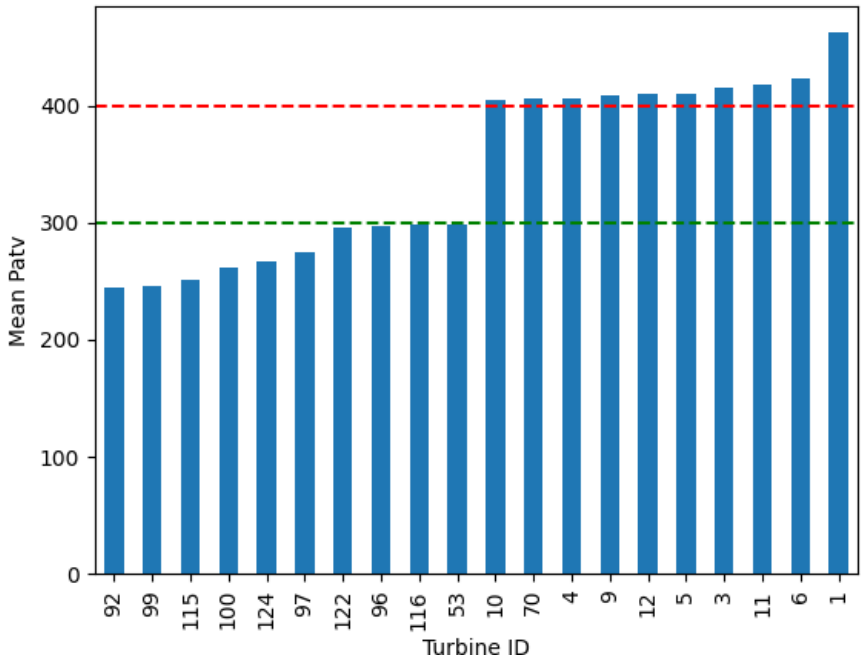

Figure 4: Mean power generated (Patv) of bottom and top 10 wind turbines in the whole period. The top 10 wind turbines have a mean Patv of above 400, and the bottom 10 have a mean below 300.

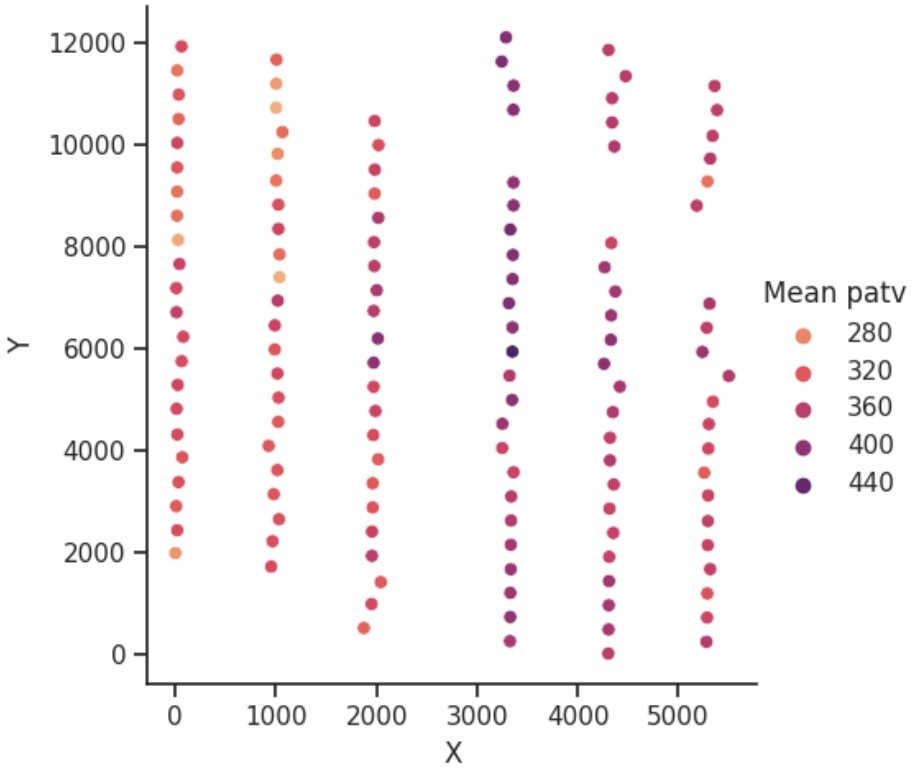

Figure 5: Spatial distribution of all 134 wind turbines in the farm. X and Y axes are both in meters. The wind turbines are colored by their mean power generated in the whole period.

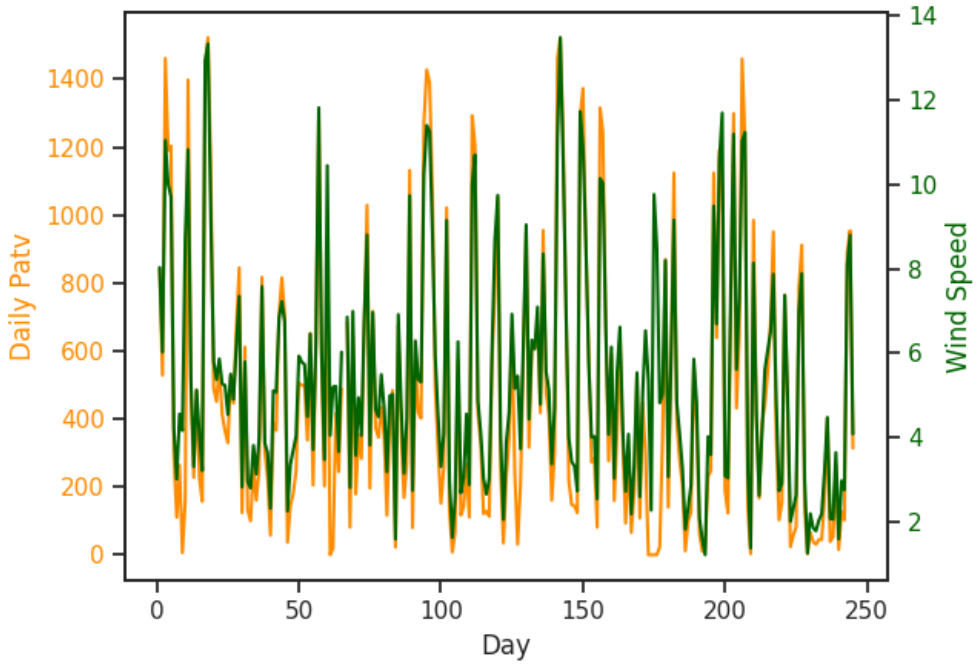

Figure 6: Plot of the daily aggregated power generated (kWh) and wind power (m/s) of turbine 1 during the whole period.

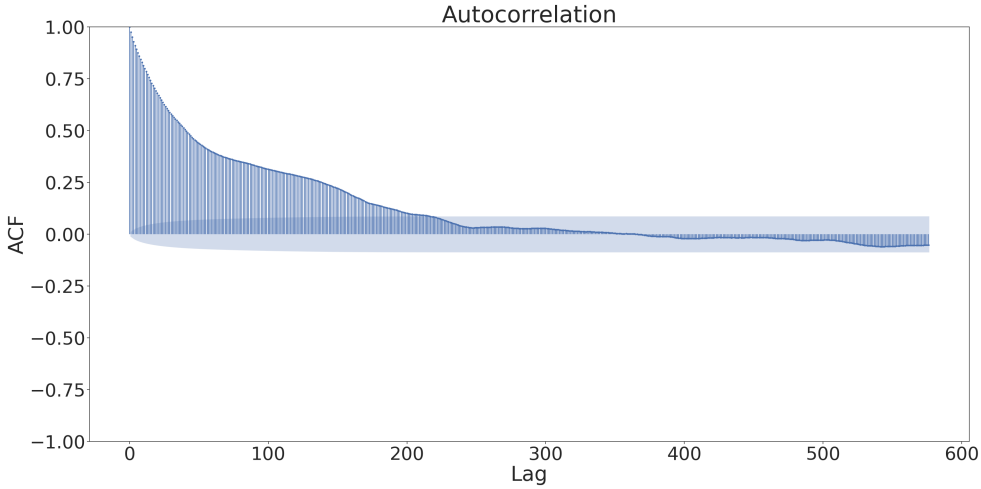

Figure 7: ACF plot of power generated (Patv) of turbine 1 (a single of the 134 turbines). The plot has a lag of 576 time steps of 10-minutely data, corresponding to 96 hours, or 4 days. The light-blue overlay is a 95% confidence interval

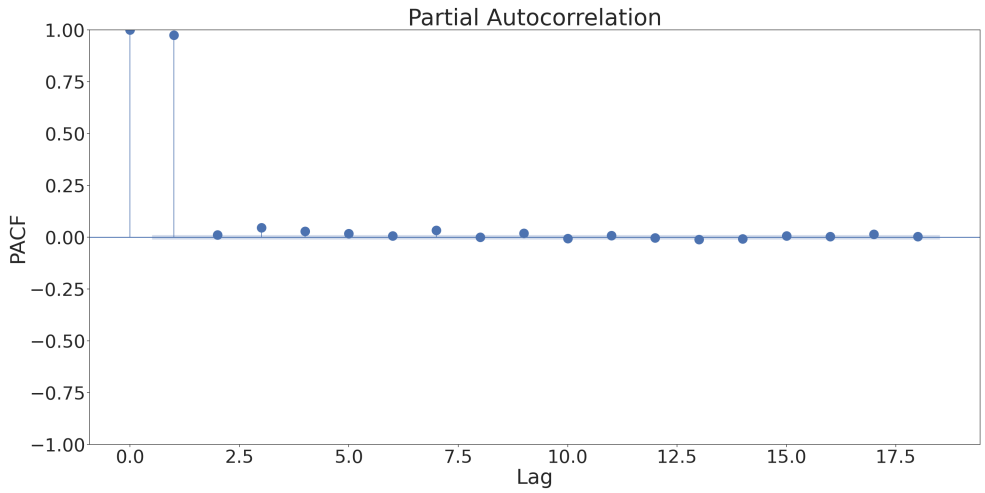

Figure 8: PACF plot of power generated (Patv) of turbine 1 (a single of the 134 turbines). The plot has a lag of 18 time steps of 10-minutely data, corresponding to 3 hours.

4 METHOD

4.1 MODEL ARCHITECTURES AND TRAINING

As mentioned, the models are selected such that there are different layers of complexity in the hopes that it translates to different levels of carbon emission, including the simple, baseline ARIMA model. The following describes the architectures of each of the models.

4.1.1 Baseline ARIMA

The previous figures 4 and 5 indicate a difference in the distributions of each respective wind turbine, and therefore an individual ARIMA model was created for each 134 wind turbines. The parameters of each model was estimated on the whole training set subset by each respective turbine IDs. This is a case of a bottom-up approach as described in section 2.1. However, since this project only evaluates models at the bottom, wind turbine level, no further reconciliation was needed.

As previously noted, ARIMA has three hyperparameters. They can be chosen with different methods, such as time-series cross-validation, information criteria[1] or simply by inspecting the data. The latter is chosen here for the simplicity of the baseline model. Visual inspection of figure 6 and the ACF plot in figure 7 indicates that the power generation data of a single turbine is stationary (at least at a 10-minutely frequency - aggregations at lower frequencies may have bigger trends). I assume that the remaining turbines show a similar pattern, and d is therefore set to 0 for the models, which means it is a case of an ARMA model - or ARIMA(p,0,q). The ACF plot at figure 7 shows significant autocorrelation up until around a lag of 200. Here, a lag of 10 (p) is chosen to only use the lags with the highest correlation and to avoid having too many parameters in the model. The partial ACF plot at figure 8 shows that the data has a high and significant correlation with just a single lag after removing the effects of the previous lags. As such, the last parameter q is chosen to be 1 for the models. So we have ARIMA(10,0,1) for all the 134 wind turbine models.

Since the official test set does not directly preceed the training, the `statsmodels`-method `apply()` is used to set each test input as the context of the ARIMA models during evaluation.

4.1.2 BERT model (KDD Cup 3rd place)

The model from Tan and Yue (2022) evaluated in this project consists of a single BERT model with an embedding layer that transforms the raw inputs into the attention space. This is followed by a standard BERT encoder with self-attention, a feed-forward network, layer normalisation and residual connections. Three dense layers with dropout comes afterwards. The last dense layer has a hidden size of 288, which is equal to the number of timesteps the model should predict. The model is implemented in `Tensorflow`.

Table 4 gives an overview of the hyperparameters used in the BERT model. The training was performed with a batch size of 1024, 3 training epochs, a learning rate of 0.005 and Adam as optimizer.

[1] https://otexts.com/fpp2/arima-estimation.html

Hyperparamters	Value
Train sequence lengths	288
Predict sequence lengths	288
Number of encoder layers	1
Attention hidden sizes	32
Number of attention heads	1
Attention drop out rate	0
Feed forward layer hidden sizes	32
Feed forward layer dropout rate	0
Dense1 hidden sizes	512
Dense1 dropout rate	0.25
Dense2 hidden sizes	1024
Dense2 dropout rate	0.25
Dense3 hidden sizes	288

Table 4: KDD CUP 3rd place BERT model hyperparameters.

4.1.3 GNN model (KDD Cup 11th place)

The model from Jiang et al. (2022) is a variant of GNN called MTGNN (short for Multivariate Time Series Forecasting Graph Neural Networks). It consists of multiple sequentially connected gated temporal convolutional modules and spatial graph modules with extra residual connections. The output module consists of two 1x1 convolutions that project the hidden features to the predicted timestep of 288. The model is implemented in `Pytorch`.

The hyperparamters of the GNN model can be seen in table 5. For the training, a sequence of length 144 was used to predict a sequence of length 288. Adam is used as optimizer with a learning rate of 0.001, a batch size of 32 and 30 training epochs.

Hyperparamters	Value
Train sequence lengths	144
Predict sequence lengths	288
Gated temporal convolutional modules	3
Dilation exponential factor in the Gated Temporal Convolutional Module	2
Spatial graph convolutional modules	3
Features retained in the Spatial Graph Convolution Module, β	0.05
Hidden dimension in both modules	32
Skip connection dimension in both modules	64
Information propagation step in both modules	2

Table 5: KDD CUP 11th place GNN model hyperparameters.

4.2 EVALUATION

Due to the abnormalities in the data, the KDD CUP 2022 officially evaluated its participant models on a wind turbine level (as opposed to the level of the whole wind farm), where abnormal data points were ignored. This project evaluates a similar approach, where abnormal data points in the test set are set to be equal to the model prediction, if they match any condition in 3 such that the error for any abnormal observation equals to 0.

For a given timestep t_0, the RMSE is computed for each wind turbine i and the preceeding 288 timesteps of that wind turbine, such that we have:

$$s^i_{t_0} = \sqrt{\frac{\sum_{j=1}^{288}(\text{Patv}^i_{t_0+j} - \bar{\text{Patv}}^i_{t_0+j})^2}{288}}$$

, where $\text{Patv}^i_{t_0+j}$ is the actual power of wind turbine i and $\bar{\text{Patv}}^i_{t_0+j}$ is the predicted power of wind turbine i at timestep $t_0 + j$.

The score of each wind turbine is then summed to end up with a score for a single timestep:

$$S_{t_0} = \sum_{i=1}^{134} s^i_{t_0}$$

Finally, the scores for each observation in the test set are averaged.

5 RESULTS & ANALYSIS

Table 6 shows the reults of training and evaluating the baseline model and the two models from the KDD CUP. The number of trainable model parameters are included, too, which are found through native functions in `Tensorflow` and `PyTorch`. Lastly, the carbon emission from Carbontracker during training is reported. The score is the average RMSE score summed for all wind turbines in kWh.

Model	# of Model Parameters	Score	Carbon Emission
Baseline ARIMA	12	43,199.6	**23.5 g**
BERT	842,784	**42,823.7**	70.6 g
GNN	4,238,976	43,111	65.7 g

Table 6: Model, number of model parameters, score on test set and carbon emission of training for each evaluated model. The score is the summed RMSE of each wind turbine, measured in kWh.

The BERT model performs best but the scores are very close. The neural network models clearly have a higher carbon emission during the training, and BERT leads with 3 times as much emission compared to the baseline ARIMA. With such a small increment in performance and a high proportional increase in carbon emission, it leads to the question; Is the performance gain worth the carbon emission?

To answer the question, we can convert the carbon emission to a more understandable metric. Carbontracker does so by translating the emission to kilometers driven in the average car. The emissions of training BERT and the baseline ARIMA translate to driving in car for around 0.65 and 0.22 kilometers, respectively. In other words, training BERT over the baseline corresponds to driving 0.43 more kilometers by car.

The additional emission looks rather negligible. I suspect the low emissions of the models occur because of the relatively small feature space. The 842,784 BERT parameters is still a relatively small number of parameters compared to the example from Strubell et al. (2019) with 100 million parameters - a factor of 118 higher.

Before a final conclusion, it is worth noting that the single, final training does not encapsulate the whole computational pipeline of the models. Preprocessing, predicting and - especially for the neural networks - hyperparameter-tuning are assumed to impact the emissions. The evaluated models do not document the process of hyperparameter-tuning, but we can do a conservative guess that they have searched for at least 10 different hyperparamters - and therefore re-trained 10 different time. For the BERT model, this means 706 g carbon emission, or 6.5 kilometers driven by car.

6 CONCLUSION

This project sought to answer the research questions: **What is the trade-off between model complexity and prediction accuracy of Wind Power Forecasting models? And what are the effects in carbon emission?** Three models were chosen: A baseline ARIMA model with 12 model parameters, a BERT model with 842,784 parameters and a GNN with 4,238,976 parameters. All the models were evaluated using an averaged RMSE which was summed for all wind turbines and measured in kWh. The carbon emission of training the models was also measured using the Python-library `Carbontracker`.

The models have a similar performance with the BERT model achieving the highest performance, but also the highest carbon emission. The model emitted 70.6 g carbon during a single training, or 706 g estimated carbon during 10 iterations of hyperparameter-tuning. This is more than the baseline ARIMA, but still a negligible amount, as it compares to just 6.5 kilometers driven by car. The model had a performance increase of around 376 kWh (or 0.38 mWh) for the whole wind turbine farm as compared to the baseline. Whether this performance increase justifies the increase in carbon emission depends on the needs of the grid system and how necessary the increased accuracy is.

One limitation of this study is its focus on offline training. Wind power prediction inherently is an online training problem that requires continuous updating of the models to fit the most recent data. Thus, the continuous carbon emission of updating the model and making the predictions should also be addressed.

BIBLIOGRAPHY

Anthony, L., B. Kanding, and R. Selvan (2020). Carbontracker: Tracking and predicting the carbon footprint of training deep learning models.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

English, L. and M. Abolghasemi (2023). How to forecast power generation in wind farms? insights from leveraging hierarchical structure.

Hyndman, R. and G. Athanasopoulos (2018). Forecasting: principles and practice, 2nd edition.

Jiang, J., C. Han, and J. Wang (2022). $Buaa_bigscity : Spatial - temporalgraphneuralnetworkforwindpowerforecastinginbaidukddcup2022$.

Strubell, E., A. Ganesh, and A. McCallum (2019). Energy and policy considerations for deep learning in nlp.

Tan, L. and H. Yue (2022). Application of bert in wind power forecastin-teletraan's solution in baidu kdd cup 2022.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Kaiser, and I. Polosukhin (2017). Attention is all you need.

Wu, Z., S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang (2020). Connecting the dots: Multivariate time series forecasting with graph neural networks.

Zhou, J., X. Lu, Y. Xiao, J. Su, J. Lyu, Y. Ma, and D. Dou (2022). Sdwpf: A dataset for spatial dynamic wind power forecasting challenge at kdd cup 2022.