POLITECNICO DI TORINO

Master's Degree in DATA SCIENCE AND ENGINEERING

Master's Degree Thesis

Spatio-Temporal Graph Neural Networks for Wind Power Forecasting

Supervisors

Prof. Paolo Garza

Dr. Luca Colomba

Candidate

Giovanni Mantegna

December 2023

Summary

Wind power forecasting represents a significant and challenging aspect of power supply management. The electricity grid necessitates a continual balance between electricity consumption and generation. Since wind power generation is notably influenced by wind speed, unlike conventional power generation systems, it can exhibit erratic fluctuations, requiring power substitution from alternative sources. For these reasons, wind farm production forecasts are indispensable. However, forecasting wind power production is difficult due to the inherent unpredictability of wind behavior. Various methodologies exist in the literature to address this challenge, including the utilization of numerical weather prediction models.

This thesis focuses on wind power forecasting solely based on historical turbine production data. This work aligns with the 2022 KDD Cup challenge presented by Baidu, which tasks participants with predicting wind farm active power for a 48-hour horizon using past data and wind turbine spatial positioning data. The primary emphasis of this challenge is on leveraging spatial information to enhance forecasting precision. The thesis starts with an exploratory data analysis to comprehend the distinctive characteristics of wind data and proceeds to a data cleaning process to rectify corrupted data. Several forecasting models are subsequently evaluated.

The investigation begins with a basic statistical model, Lasso regression, it advances to a more intricate model such as gated recurrent units , and ultimately focuses on spatio-temporal graph neural networks. First of all, the general framework of spatio-temporal graph neural networks is investigated and within this framework the Graph WaveNet model is considered as the most appropriate for this task. In fact, this model is able to capture informative relationships between the time series and spatial data of various turbines.

The results of the thesis are presented in two distinct contexts: short-term forecasts, considering prediction horizons of 2 hours, 6 hours, and 12 hours, and long-term forecasts, spanning 48 hours. For short-term forecasts, it is demonstrated that models such as gated recurrent units network and Graph WaveNet exhibit a pronounced superiority compared to regression-based models. Notably, the Graph WaveNet model achieves superior results when compared to the other models supporting the thesis that spatial data can be effectively employed to enhance

wind power forecasting. In the experimental section, the incorporation of external features such as wind speed and other covariates is also investigated, and an optimized hyperparameter configuration is identified.

In the long-term setting experiments, it is demonstrated that a single model is insufficient for accurate 48-hour wind power predictions due to the need to simultaneously capture long-term and short-term dynamics. An ensemble model approach, combining prediction at different forecast horizons (12 hours and 48 hours), is used to improve forecasting performance. This ensemble model consists of two Graph WaveNet models trained separately, one for prediction in the short-term setting and one for prediction in the long-term setting. Two different graph configurations were used for the two models. The resulting best-performing model in this thesis achieves results comparable to the top 9 participants out of more than 2000 entrants in the 2022 KDD Cup challenge.

In the final part of the thesis, potential future implementations are explored. On November 14, 2023, Google DeepMind released the source code of GraphCast, an auto-regressive graph neural network model capable of providing more accurate weather forecasts than numerical weather prediction models. Combining GraphCast with the local graph neural network studied in this thesis could be effective, since GraphCast provides an accurate prediction on a large scale while a local graph neural network provides an accurate prediction on a short scale. This hypothetical model could provide precise forecasts and enhance wind power forecasting performance, thus improving wind energy management.

Acknowledgements

I extend my gratitude to my supervisors, Professor Paolo Garza and Dr. Luca Colomba, for their guidance and valuable feedback throughout the completion of this thesis. Special appreciation goes to my brother Francesco, my father Rosario, and my cousin Fabio for their advice and support in navigating the complexities of the research.

I am also deeply thankful for the support of those who have been by my side during these two years. A sincere acknowledgment goes to my friend Domenico, without whom my time in Turin would have been less enjoyable. Additionally, I want to express my gratitude to my dear friends Dario, Vincenzo, Federico, Silvia, and Alessandro for their encouragement.

Lastly, I extend my heartfelt thanks to my entire family, especially my mother Francesca, including my feline companion Sam, for their enduring support and patience throughout this educational journey.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the context of addressing climate change, wind energy plays a fundamental role among renewable energy sources. In recent years, substantial investments have been directed towards the establishment of wind farms, particularly in offshore locations. However, unlike some other renewable energy sources, wind energy production is inherently characterized by its intermittent and unpredictable nature. This inherent variability poses a significant challenge for both grid operators and power grid management. To tackle the challenge of managing wind energy effectively, it has become necessary to develop methods for forecasting wind energy production over specific forecast horizons. This task is referred to as wind power forecasting (WPF) and can be accomplished through a variety of approaches. One common method involves the use of physical models, such as precise numerical weather prediction (NWP) models. While these models offer high accuracy, they come with substantial computational costs and are better suited for longer-term forecasts. However, when the need arises for shorter-term forecasts, spanning hours or days, alternative approaches become more relevant. These alternatives encompass statistical methods, machine learning, and deep learning techniques. These methodologies leverage historical turbine production data and other sensor-derived information to predict future wind energy generation. In this context, the Chinese company Baidu introduced the Baidu KDD CUP 2022 challenge [1], aimed at identifying the most effective model for predicting wind energy production time series. A distinctive feature of this challenge is the provision of spatial coordinates for the turbines, enabling the integration of spatial data with temporal information to enhance forecasting accuracy. In fact, the spatio-temporal interdependence of different physical locations can be particularly important for power forecasts since physical properties such as wind fields, are non-stationary in both space and time, meaning that historical time series for different physical locations should be jointly considered to better learn global trends and propagation in both space and time. Although the challenge ended in 2022 with the participation of over 2000 teams

worldwide, the dataset remains available for further research and improvements.

Following the specification of Baidu KDD CUP 2022 challenge, the primary objective of this thesis is to present a comprehensive analysis of methodologies suitable for the task of wind power forecasting, focusing on a 2-day forecast horizon. The thesis aims to show the key factors and critical challenges inherent in wind power forecasting. Specifically, the thesis demonstrates how state-of-the-art deep learning methods can yield more accurate predictions compared to traditional statistical models. In this regard, spatio-temporal graph neural networks are employed, showing their capability to analyze and integrate both temporal and spatial information. An initial phase encompasses an exploratory analysis of the dataset and a data cleaning process. Subsequently, three distinct forecasting methods are investigated: 1) Lasso regression, serving as a foundational statistical model and acting as the benchmark for subsequent models, 2) a Gated Recurrent Unit (GRU) [2] network evaluated for its predictive performance and 3) the innovative Graph WaveNet [3], a member of the spatio-temporal graph neural network family, assessed for its forecasting capabilities. The thesis conducts a comparative analysis of these models, considering both short-term (2 hours, 6 hours, 12 hours) and long-term (48 hours) forecasting scenarios, demonstrating the superior performance of the spatio-temporal time series forecasting approach for each forecasting horizon. Finally, an ensemble model that enhances long-term forecasting by combining short-term and long-term predictions is presented.

The structure of the thesis is the following: Chapter 2 reviews the related works in the field of wind power forecasting. Chapter 3 provides an overview of the theoretical concepts of the methods presented in subsequent sections. Chapter 4 details the KDD Baidu challenge dataset and its distinctive characteristics. Chapter 5 explores in detail the theory behind spatio-temporal graph neural network methods. Chapter 6 presents the experiments and their results. Lastly, Chapter 7 offers concluding remarks and final considerations.

Chapter 2

Related works

The chapter discusses the extensive body of scientific literature concerning Wind Power Forecasting (WPF) and its evolution over time in response to advancements in time series forecasting. The primary focus of this chapter is on statistical models, excluding numerical weather prediction (NWP) models and physical models. Historically, one- and two-day forecasts of wind speed relied on basic statistical methods such as AutoRegressive Integrated Moving Average (ARIMA) models. Notable studies by Kavasseri and Seetharaman [4] and Singh, Mohapatra, et al. [5], applied variations of ARIMA to enhance wind power forecasting. Machine learning algorithms like Support Vector Regressor (SVR) and K-nearest neighbor (KNN) were also employed in this context [6, 7, 8]. In recent years, the scientific community has increasingly turned to neural networks for their ability to capture complex non-linear relationships. Among neural network architectures, Recurrent Neural Networks (RNN) have gained popularity, with variants like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) showing promise in improving wind power forecasts, as demonstrated by Li et al. [9]. WaveNet, introduced by Oord et al. [10] in 2016 shows the utility of 1D convolution for time series forecasting, particularly for capturing long-term dependencies. Convolution 1D has proven effective in wind power forecasting, as seen in the work by Shivam et al.[11] and the work by Dong et al.[12]. In 2017, transformer architectures were introduced in Vaswani et al. innovative paper "Attention is All You Need" [13]. Although originally designed for natural language processing, transformers have found applications in various deep learning domains. For time series forecasting, multiple transformer models, including Temporal Fusion Transformer[14], LogSparse Transformer[15], and Autoformer[16], have been developed and have shown significant promise in wind power forecasting as highlighted by different papers [17, 18, 19]. Recent research has emphasized the use of spatial data to enhance wind power forecasting. The use of Spatio-temporal Graph Neural Networks (STGNN) has gained prominence, allowing the incorporation of both spatial and

temporal information. These STGNN architectures, initially developed for tasks such as traffic forecasting, have been adapted to wind power forecasting. Specialized examples include Spatio-Temporal Graph Convolutional Networks (STGCN) [20], Adaptive Graph Convolutional Recurrent Networks (AGCRN) [21], Graph WaveNet [3], and Diffusion Convolutional Recurrent Neural Network (DCRNN) [22]. These architectures have demonstrated the capability of spatial information to enhance forecasting. Several recent studies, including those by Stańczyk and Mehrkanoon [23], Wang et al. [24] and Wang and He [25] have leveraged STGNNs for wind power forecasting, showing superior performance compared to previous models.

In the context of the Baidu KDD CUP 2022 challenge [1], various high-performance models have emerged. Huerta et al. proposes a solution combining GRU models for short-term forecasting and a KNN regressor for long-term forecasting [26]. Zhao et al. utilized GRU for short-term forecasting and a linear temporal layer for long-term forecasting, achieving fifth place in the challenge [27]. Spatial data proves to be a focal point for enhancing forecasting in the challenge, as demonstrated in the work of Jiang et al. who tested two types of STGNNs models, the AGCRN and MTGNN models [28]. The winning solution of the challenge, proposed by Li et al., employed an ensemble model that integrates two modules: the DMST module for short-term forecasting and the ST-Tree module for long-term forecasting, both of which use spatial and temporal aggregation techniques [29].

Chapter 3

Background

In this chapter, the theory behind the methods used in the experiments section will be briefly introduced. These methods are currently used for the time-series forecasting problem and are based on machine learning and deep learning.

3.1 Lasso Regression

Within the domain of machine learning, regression encompasses a collection of mathematical methodologies that empower data scientists to anticipate continuous outcomes denoted as y by harnessing one or more predictor variables denoted as x. Among these methodologies, linear regression distinguishes itself as the most widely embraced approach due to its simple and effective characteristics in the domains of prediction and forecasting.

In the context of linear regression, the principal objective is to determine a hyperplane. This determination is achieved through the minimization of the cumulative sum of mean-squared errors across all data points. To express this formally in mathematical terms, linear regression is employed to address the following problem statement:

Given a set of N tuples $(\mathbf{x_i}, y_i)$ where $\mathbf{x_i}$ belongs to $\mathbb{R}^n$ and y_i belongs to $\mathbb{R}$ for all i in the range from 1 to N, we aim to find:

$$\widehat{y} = \beta_0 + \overline{\beta}\mathbf{x} \tag{3.1}$$

subject to the condition:

$$min_{\beta_0,\overline{\beta}} \sum_{i=1}^{N} \left\| y_i - (\beta_0 + \overline{\beta}\mathbf{x_i}) \right\|^2 \tag{3.2}$$

In the experimental section, Lasso regression, a regularized version of linear regression, is utilized. Lasso regression not only minimizes the cumulative sum of

mean-squared errors but also incorporates a regularization term, resulting in the following optimization problem:

$$min_{\beta_0,\overline{\beta}} \sum_{i=1}^{N} \left\| y_i - (\beta_0 + \overline{\beta}\mathbf{x_i}) \right\|^2 + \alpha \sum_{k=0}^{N} |\beta_k| \tag{3.3}$$

The regularization term promotes simpler models by driving many regression coefficients to zero.

Regression techniques find utility in the realm of time-series forecasting, where predicting the next time step can be viewed as a regression task based on previous time records. Two primary strategies for employing regression methods in multi-step time-series forecasting exist. Firstly, a recursive multi-step forecasting approach employs a single regression model to iteratively predict multiple time steps. The model leverages the forecast at time $t+1$ to predict those at time $t+2$, as illustrated in Figure 3.1.

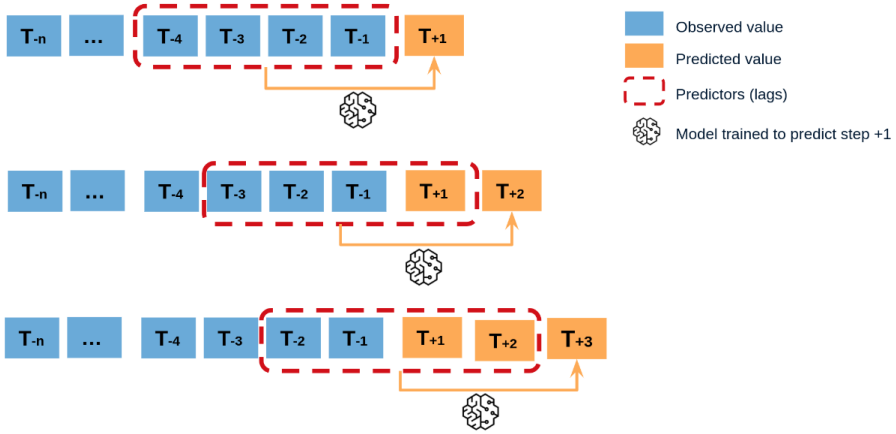


Figure 3.1: Recursive multi-step forecasting. Blu boxes represent observed values whereas orange boxes represent predicted values. The input window to the model is highlighted by a red dashed line.

Alternatively, a direct multi-step forecasting approach can be adopted. In this scenario, a multitude of regression models, equivalent to the number of time steps to predict, are employed, as depicted in Figure 3.2.

The direct multi-step forecasting approach is computationally intensive, as it necessitates training N distinct models, where N represents the number of steps to forecast. For extended forecast horizons, the direct forecasting method exhibits significantly improved performance compared to the recursive approach.

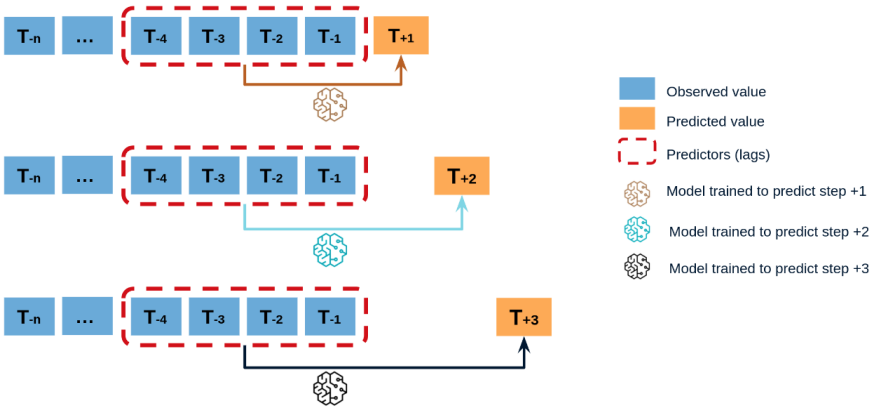


Figure 3.2: Direct multi-step forecasting. Blu boxes represent observed values whereas orange boxes represent predicted values. The input window to the model is highlighted by a red dashed line.

3.2 Neural Networks

A neural network is a computational system inspired by the structural organization of the human brain. It consists of multiple fundamental processing units known as perceptrons (Figure 3.3). Each perceptron, when given an input vector $\mathbf{x}$, operates as follows:

$$out = f(\mathbf{w} \cdot \mathbf{x} + b) \tag{3.4}$$

Here, $\mathbf{w}$ represents a weight vector, and b is the bias term. The function $f(\cdot)$ is an activation function that maps inputs to values within the range of 0 to 1.

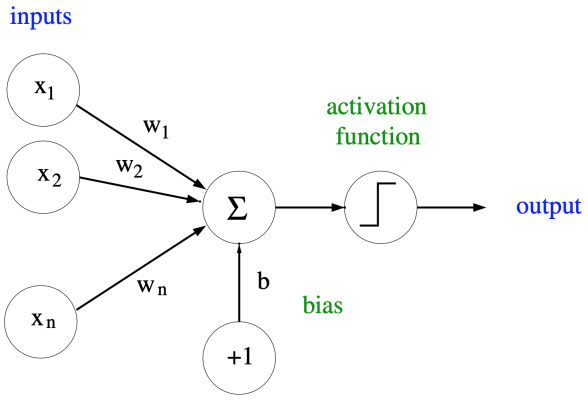


Figure 3.3: Basic scheme of the perceptron structure

The neural network's architecture is built by arranging various perceptrons into different layers. The output of one layer's perceptrons serves as the input for the perceptrons in the subsequent layer, as illustrated in the diagram in Figure 3.4:

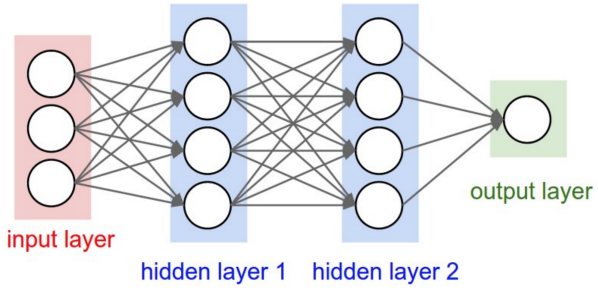


Figure 3.4: Example of a neural network structure with two hidden layers

Following an initial learning phase in which the network's weights are adjusted, the neural network, when provided with specific inputs, can generate appropriate outputs. It can accomplish complex tasks such as classification or regression.

3.2.1 Backpropagation Algorithm

The most crucial phase in neural networks is the learning phase, where the objective is to determine the weight matrix $\mathbf{W}$ composed by the vectors of weights $\mathbf{w}^{(i)}$ obtained by the learning of pairs of input and output data. This allows the network to produce correct outputs for classification or regression tasks when new input data is presented. To achieve this, the backpropagation algorithm is employed, which comprises two stages: forward propagation and backward propagation, used to refine network parameters. In the forward propagation phase, the input is fed into the network, passing through various perceptrons and layers, ultimately producing a final output. For instance, in a network with three hidden layers, the computation unfolds as follows:

$$\begin{aligned} z^{(1)} &= f(\mathbf{w}^{(1)} \cdot \mathbf{x} + b^{(1)}) \\ z^{(2)} &= f(\mathbf{w}^{(2)} \cdot \mathbf{z^{(1)}} + b^{(2)}) \\ out &= \mathbf{w}^{(3)} \cdot z^{(2)} \end{aligned} \tag{3.5}$$

The goal is to find the output *out* in such a way as to minimize a cost function that depends on all the weights $\mathbf{W}$:

$$C = cost(out, y) \tag{3.6}$$

Here, $cost(\cdot)$ represents any differentiable function, and y is the expected output. To achieve this minimization, gradient descent is employed since, in most cases, the function cannot be solved analytically. During each training step, the weights are updated as follows:

$$\begin{aligned} w_k^{new} &= w_k^{old} - \eta \frac{\partial C}{\partial w_k} \\ b_k^{new} &= b_k^{old} - \eta \frac{\partial C}{\partial b_k} \end{aligned} \tag{3.7}$$

Where w_k is a component of the vector $\mathbf{w^{(i)}}$ and b_k is the bias of a layer. This process ensures a descent towards a local optimal minimum. Neural networks and their variants serve as foundational components in all architectures utilized in deep learning. Subsequent architectures we encounter will employ variations of neural networks as their fundamental building blocks.

3.3 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks, abbreviated as RNNs, constitute a specialized category of neural networks well-suited for handling sequential data, such as time series or natural language. Unlike conventional neural networks, where each fundamental computational unit is linked to every unit in the subsequent layer, RNNs are structured to embrace the temporal nature of the sequence. To illustrate this concept visually, consider the diagram in Figure 3.5.

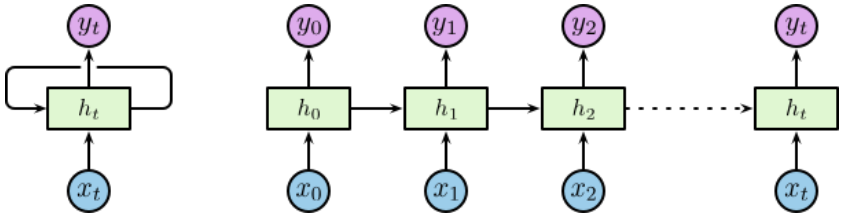


Figure 3.5: RNN structure

In this diagram, we can observe the essence of an RNN's architecture, where each time step is represented, and connections loop back to capture the sequential relationships between data points. In the context of RNNs, we have N inputs corresponding to different time points from 0 to t, and hidden states are introduced, denoted as $\mathbf{h_t}$, to account for the sequential nature of the data. These hidden states represent the outputs of neurons at time $t-1$ and serve as inputs for the neurons at time t. Consequently, the output at a subsequent time step t is calculated as follows:

$$h_t = f(h_{t-1}, x_t) \tag{3.8}$$

Here, x_t represents the input at time t, and $f(\cdot)$ denotes a non-linear function. In the PyTorch implementation, the output is expressed as:

$$h_t = tanh(x_t W_{ih}^T + b_{ih} + h_{t-1} W_{hh}^T + b_{hh}) \tag{3.9}$$

During the training phase, as is the case with traditional neural networks, the network learns the weights W_{ih} and W_{hh} which are shared across all layers. The learning algorithm employed for this purpose is known as Backpropagation Through Time (BPTT), which differs slightly from the classic backpropagation. However, it is important to note that RNNs face certain challenges during training. They are particularly susceptible to the issue of vanishing gradients, making it challenging to train weights associated with deeper layers. Additionally, their architecture primarily focuses on capturing short-term dependencies, making it difficult for RNNs to effectively model long-term patterns.

3.4 Long Short Term Memory networks (LSTMs)

LSTM (Long Short Term Memory) networks represent an evolution of Recurrent Neural Networks (RNNs), specifically designed to effectively handle long-term dependencies. They find extensive application in tasks where the consideration of historical data over extended periods is critical, such as time series forecasting and natural language processing. To address the challenge of managing long-term dependencies, LSTM introduces a specialized memory cell structure (Figure 3.6) capable of retaining information for prolonged periods. This memory cell comprises three essential components: the input gate, the forget gate, and the output gate, along with a cell state that propagates information to subsequent time steps.

The forget gate's primary function is to filter out irrelevant information and preserve useful data. This operation is accomplished through a neural network layer described by the equation:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.10}$$

Here, the function $\sigma(\cdot)$ is the sigmoid function and returns values between 0 and 1, determining the extent to which information from previous time steps should be forgotten and W_f are the weights of the forget gate. Conversely, the input gate regulates the incorporation of new data at time t and its impact on the cell state C_t. This process involves two key calculations:

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \widetilde{C_t} &= tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \end{aligned} \tag{3.11}$$

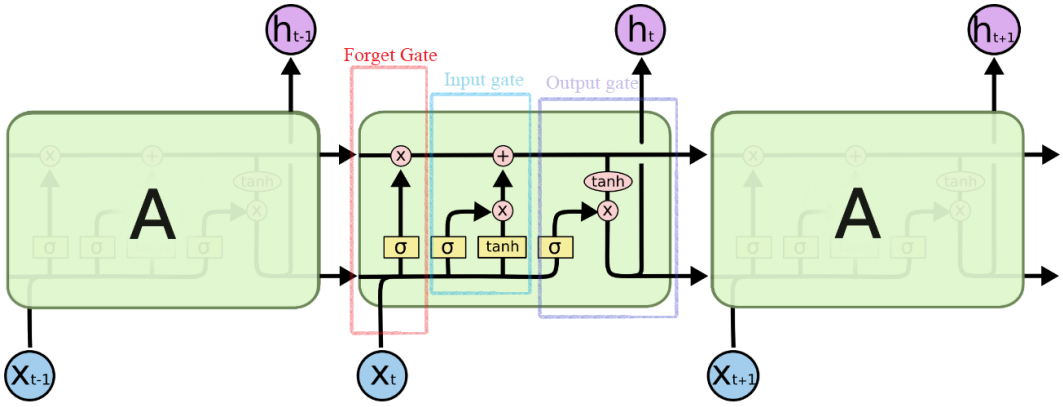


Figure 3.6: LSTM cell structure

Here the index i refers to the input gate and the index C to the cell state. The cell state, C_t, is computed by combining the contributions from the forget gate and the input gate:

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \tag{3.12}$$

Finally, the output gate (labeled with index o) is responsible for generating the memory cell's output and is calculated as follows:

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * tanh(C_t) \end{aligned} \tag{3.13}$$

Not only do LSTMs address the challenge of handling long-term dependencies, but they also effectively mitigate the issue of vanishing gradients. Nevertheless, the training procedure of LSTMs can be a more demanding task due to their significantly larger parameter space compared to conventional RNNs.

3.5 Gated Recurrent Unit (GRU)

Another variant of recurrent neural networks (RNN) is known as the Gated Recurrent Unit (GRU). While its fundamental concept bears resemblance to Long Short-Term Memory (LSTM) networks, GRU simplifies the architecture by merging the forget gate and input gate into a single entity termed the "update gate." In conjunction with this, another gate called the "reset gate" is introduced. Unlike LSTMs, GRUs do not maintain a separate cell state distinct from the hidden state. Instead, the memory cell behaves as illustrated in Figure 3.7.

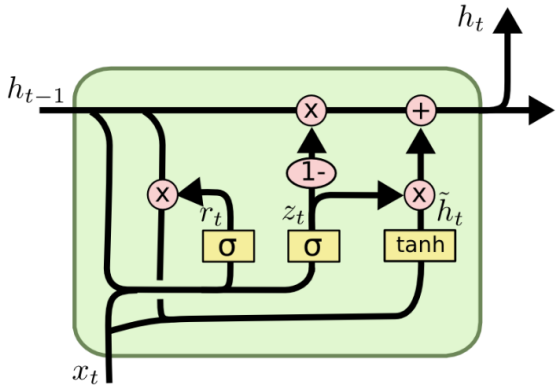


Figure 3.7: GRU cell structure

The behavior of the memory cell is mathematically defined by the following equations:

$$\begin{aligned} z_t &= \sigma(W_z \cdot [h_{t-1}, x_t]) \\ r_t &= \sigma(W_r \cdot [h_{t-1}, x_t]) \\ \widetilde{h}_t &= tanh(W \cdot [r_t * h_{t-1}, x_t]) \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \widetilde{h}_t \end{aligned} \tag{3.14}$$

Where W_z are the weights of the update gate and W_r are the weights of the reset gate. The notable advantage of this memory cell structure lies in its reduced parameter count, which simplifies the training phase and often yields improved results. It remains uncertain which memory cell structure performs optimally across diverse tasks. Various studies, such as those conducted by Greff et al. [30] (2016) and Jozefowicz et al. [31] (2015), have compared different RNN structures. Their collective findings suggest that the choice of the best structure frequently depends on the specific task and dataset characteristics.

3.6 Temporal Convolutional Networks (TCN)

Convolutional Neural Networks (CNNs) stand as a significant breakthrough in the deep learning field. Ever since the pioneering CNN architectures [32, 33], they have elevated performance across a wide spectrum of tasks, such as vision and speech. The fundamental concept underlying this architectural approach involves the incorporation of convolutional layers in addition to the fully connected layers. Within these convolutional layers, filters or kernels are employed to convolve with the input data, thereby extracting valuable feature maps. Convolution represents a specialized operation in which one matrix is processed with another matrix. This

operation entails the multiplication of cell values in a matrix with corresponding cell values in the filter matrix. In more recent work by Bai et al. [34], convolutional networks for time series forecasting are formalized, resulting in the introduction of Temporal Convolutional Networks (TCN). For 1D sequences, applying convolution implies careful consideration, as it must consider the causal constraint. In fact the prediction at time t should solely depend on preceding time points without incorporating information from the data to be predicted. To address this, causal convolution was introduced in Oord et al. [10]. Causal convolution exclusively involves elements from time t and earlier in the previous layer during the convolution process (see Figure 3.8).

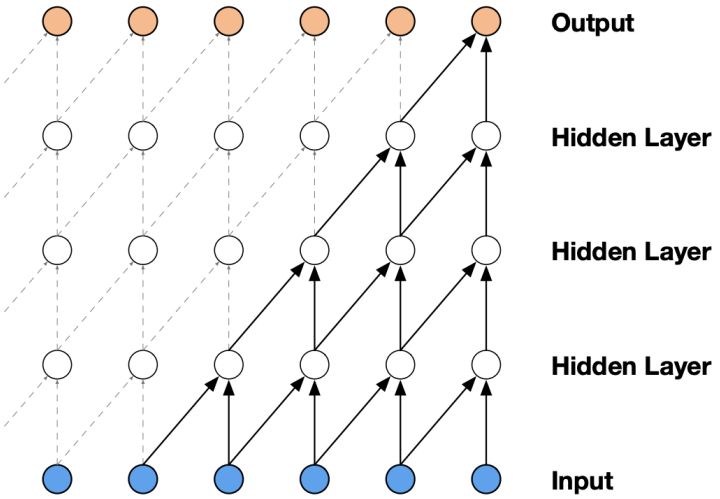


Figure 3.8: Example of causal convolution

Nonetheless, simple causal convolution is restricted in its ability to consider a historical context that scales linearly with the network's depth. This limitation requires very deep architectures, particularly in tasks that require access to a more extensive historical context. Consequently, Oord et al. [10] introduced dilated convolution, represented by the operation F on an element s:

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i} \tag{3.15}$$

Here, d represents the dilation factor, k indicates the filter size, and $s - d \cdot i$ accounts for the past direction. Dilation introduces a fixed step between each pair of adjacent filter taps (see Figure 3.9).

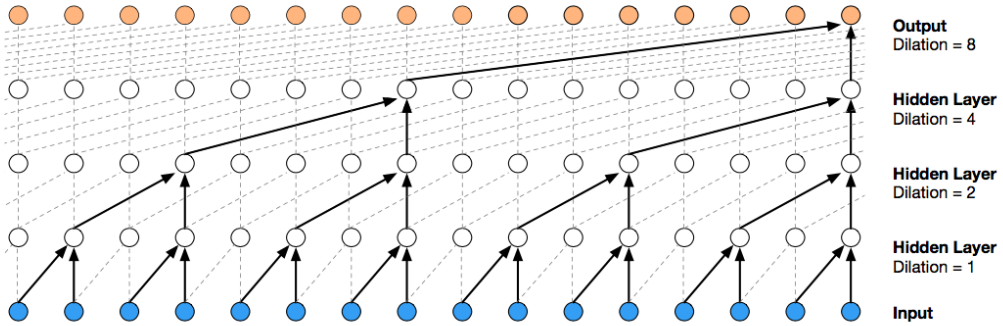


Figure 3.9: Example of dilated convolution with k = 2 and d = 1,2,4,8

3.7 Transformers

In the field of deep learning, Transformers have emerged as the most important and innovative architecture, revolutionizing numerous tasks such as, for example, natural language processing. These neural network architectures were first introduced in 2017 through the innovative paper "Attention is All You Need" by Vaswani et al. [13]. What sets Transformers apart is their distinctive Encoder-Decoder structure (see Figure 3.10) and the use of attention mechanisms. While the Encoder module is tasked with crafting a hidden representation of the input data, the Decoder module's role is to generate the correct output, using the Encoder's hidden representation as input.

In contrast to networks employing recurrent units, Transformers need not to process data sequentially. Instead, they employ a positional encoding module to account for data position. Additionally, they leverage multihead attention modules, enabling the network to focus on specific portions of the input, significantly enhancing performance.

3.7.1 Attention Mechanism

The concept of the attention mechanism, originally introduced by Bahdanau et al. [35] in 2014, is to extract a data token's representation within an input sequence using a weighted sum of hidden representations. This approach provides crucial context information for the given data token.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{3.16}$$

Here, c_i represents the context vector for the output token i, and h_j denotes

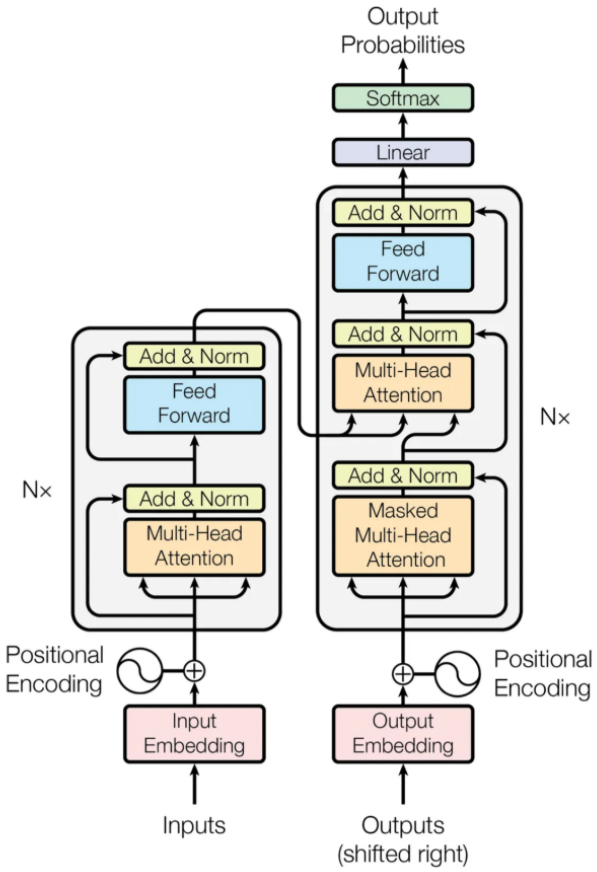


Figure 3.10: Transformers as presented in the article by Vaswani et al. [13]

the hidden representations. However, the attention mechanism described in the paper by Vaswani et al. [13] takes a slightly different approach. It is based on three distinct quantities: Query, Key, and Values. The underlying concept remains the same, but the computation of these three values differs.

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q} \cdot \mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \tag{3.17}$$

Here, Q and K represent projection vectors for the decoder and encoder, respectively. The dot product between Q and K^T yields a measure of similarity between the two representations, which is then normalized by the sequence size d_k. This results in a weighted sum of V values, offering essential context about the data concerning other data in the sequence. The paper also introduces the possibility of employing multiple attention heads in parallel, repeating the process several times to identify diverse relationships within the data.

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(head_1, ..., head_h)W^O \\ head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{3.18}$$

In these equations, W_i^Q, W_i^K, and W_i^V are weight matrices determined through linear layers.

3.8 Graph Neural Networks

Up to this point, we have explored network structures designed to tackle problems associated with Euclidean data, including 2D and 1D datasets. However, in reality, many types of data exhibit intricate relationships that are more accurately depicted as graphs. Notable examples of such data encompass social networks and protein structures. In this broader context, traditional neural networks like Convolutional Neural Networks (CNNs) treat data structures as grid graphs and RNNs view sequential data as linear graphs. Graph Neural Networks (GNNs) are a class of deep learning models tailored to process data characterized by complex, graph-based relationships.

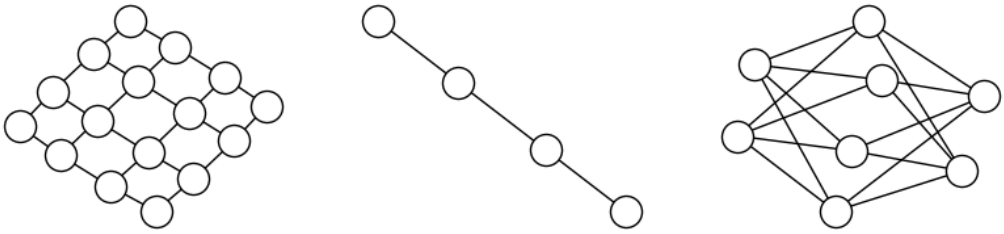


Figure 3.11: Grid graph (left), line graph (middle), and a generic graph (right)

Graph Neural Networks excel at addressing a wide range of tasks, including node classification or regression, link prediction, graph classification, and even spatio-temporal forecasting. Their approach to these tasks hinges on the acquisition of effective node representations, accomplished by integrating information from both the graph structure and node attributes. To formalize this framework, let's consider a graph, denoted as $G = (V, E)$, where V represents the set of nodes and E denotes the set of edges. We also have an adjacency matrix $A \in \mathbb{R}^{N \times N}$ representing the connections between nodes, and a matrix $X \in \mathbb{R}^{N \times C}$ that encapsulates the attributes of each node, with C denoting the number of features for each node. The primary objective of GNNs is to accurately learn $H \in \mathbb{R}^{N \times F}$, which represents

hidden information for each node, with F defining the dimensionality of this representation. This learned representation is subsequently utilized to address the specific downstream tasks. The fundamental concept underpinning GNNs is the iterative learning of node representations. This iterative process combines the representations of a node's neighbors with its own representation to update the current node representation. The general framework, as described Xu et al.[36], can be expressed through the following equations:

$$\begin{aligned} a_{\nu}^{k} &= \mathbf{AGGREGATE}^{k}\left\{H_{u}^{k-1} : u \in N\left(\nu\right)\right\} \\ H_{\nu}^{k} &= \mathbf{COMBINE}^{k}\left\{H_{\nu}^{k-1}, a_{\nu}^{k}\right\} \end{aligned} \tag{3.19}$$

Here, $N(\nu)$ represents the set of neighboring nodes of node ν. Equation 3.19 illustrates how, at each step k, the representations of neighboring nodes from the previous step $k-1$ are aggregated and combined with the current node's past representation to derive the updated representation at step k. The final representation, H^K, is subsequently leveraged to address specific downstream tasks. For instance, it can be fed into a feedforward network to yield the desired output:

$$\hat{y}_{\nu} = \sigma(W H_{\nu}^{T}) \tag{3.20}$$

In this equation, W represents the shared weights across various nodes, which are typically determined through an optimization process aimed at minimizing a loss function:

$$O = \sum loss(y_i, \hat{y}_i) \tag{3.21}$$

3.8.1 Graph Convolutional Networks (GCN)

Graph convolutional networks (GCN) introduced in 2016 by Kipf and Welling [37] is now the most popular graph neural network architecture due to its simplicity and effectiveness in a variety of tasks and applications. The two Equations 3.19 and 3.20 in this work are summarized by the following relationship:

$$H^{k+1} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{k}W^{k}\right) \tag{3.22}$$

Here $\tilde{A}$ is the adjacency matrix to which an identity matrix is added, $\tilde{A} = A + I$. This augmentation introduces self-loops to the corresponding graph and it allows to directly carry out the information of the node itself. $\tilde{D}$ is a diagonal matrix containing the degrees of the individual nodes. The term $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ can be seen as a normalization factor, weighting the influence of neighboring nodes based on their respective degrees. Equation 3.22 can be expressed in non-matrix form to provide a single-node perspective:

$$h_i^{k+1} = \sigma\left(\sum_{j \in \tilde{N}(i)} \frac{1}{\sqrt{d_i d_j}} h_j W^k\right) \tag{3.23}$$

Where $\tilde{N}(i)$ is the set of neighbors that includes the i-th node itself. and d_i is the degree of the i-th node.

Chapter 4

Dataset

The dataset employed in this thesis is the SDWPF dataset [1] proposed in the Baidu KDD CUP 2022 competition. This dataset originates from the Supervisory Control and Data Acquisition (SCADA) system of a wind farm, featuring 134 wind turbines. The SCADA system records data at 10-minute intervals, yielding a total of 4,727,520 records over a span of 245 days. The dataset encompasses 13 distinct columns, each corresponding to a specific feature, which is outlined in Table 4.1.

Column	Column Name	Specification
1	TurbID	Wind turbine ID
2	Day	Day of the record
3	Tmstamp	Created time of the record
4	Wspd (m/s)	The wind speed recorded by the anemometer
5	Wdir(°)	The angle between wind direction and turbine nacelle
6	Etmp(°C)	Temperature of the surrounding environment
7	Itmp (°C)	Temperature inside the turbine nacelle
8	Ndir (°)	Nacelle direction, i.e., the yaw angle of the nacelle
9	Pab1 (°)	Pitch angle of blade 1
10	Pab2 (°)	Pitch angle of blade 2
11	Pab3 (°)	Pitch angle of blade 3
12	Prtv (kW)	Reactive power
13	Patv (kW)	Active power (target variable)

Table 4.1: Features specifications of the SDWPF dataset

The target variable for prediction is the active power (Patv) measured in kilowatt (kW). Specifically, the goal is to forecast the power output of the entire wind farm, which can be calculated as the sum of active power for all wind turbines, i.e., $P = \sum_i Patv^i$. Additionally, the dataset includes spatial information regarding the

turbine locations, as depicted in Figure 4.1.

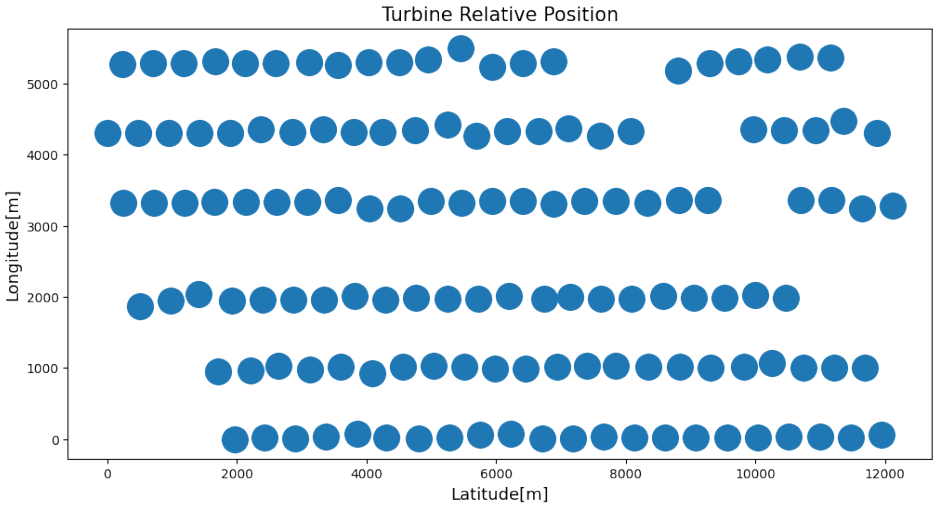


Figure 4.1: Turbine relative position, the x-axis indicates the relative latitude and y-axis the relative longitude. Each blue dot represents the location of a wind turbine.

Data Quality Considerations

It's important to note that the data was collected from sensors, and as such, some records may exhibit acquisition issues. Specifically, certain records may contain below zero values for power measurements, which is physically impossible. To address this problem, any instance where $Patv < 0$ is set to $Patv = 0$. Moreover, there are also missing values in the dataset. During the testing phase, records with missing data are excluded from the score calculation. There are also instances when wind turbines are not generating power due to external factors, like maintenance or grid management. In these cases, the active power is considered unknown. Specifically, if at time t, $Patv \leq 0$ and $Wspd > 2.5$, then the actual active power is unknown. Similarly, if at time t, $Pab1 > 89°$ or $Pab2 > 89°$ or $Pab3 > 89°$, then the actual active power is unknown. These unknown values are not used for model evaluation. Furthermore, there may be abnormal data due to recording system errors, such as $Ndir > 720°$ or $Ndir < -720°$, and $Wdir > 180°$ or $Wdir < -180°$. Records with abnormal values in any column are also excluded from model evaluation.

Evaluation metrics

The evaluation metric proposed for the challenge is a composite score that combines Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). Given the presence of unknown, abnormal, and missing values in records for different turbines, the score is calculated separately for each turbine and then aggregated to yield the overall score for the entire wind farm. The evaluation score for a specific turbine i from time t_0 to t_{0+j} is defined as follows:

$$s_{t_0}^i = \frac{1}{2}\left(\sqrt{\frac{\sum_{j=1}^{288}\left(Patv_{t_0+j}^i - \overline{Patv}_{t_0+j}^i\right)^2}{288}} + \frac{\sum_{j=1}^{288}\left|Patv_{t_0+j}^i - \overline{Patv}_{t_0+j}^i\right|}{288}\right) \tag{4.1}$$

$$s_{t_0} = \sum_{i=1}^{134} s_{t_0}^i$$

In cases where there are abnormal, missing, or unknown values, the prediction is considered correct, and the difference is set to zero: $Patv_{t_0+j}^i - \overline{Patv}_{t_0+j}^i = 0$.

Test set structure

The test set used for this research corresponds to the final phase (phase three) of the competition. It comprises 142 input files (infiles) and 142 output files (outfiles). Infiles represent 14-day datasets that serve as input for the model, while outfiles provide the ground truth for the forecasts and span 2 days (288 time instants). The files of the test set are randomly sampled from various times of the year, and the challenge's objective is to achieve the lowest possible average evaluation score on the outfiles using a subset of the infiles as input data.

4.1 Data Exploration

In a data science pipeline, it is essential to conduct an initial exploration of the dataset as part of the process. The primary objective is to uncover potential correlations and critical aspects in the dataset under investigation. Specifically, one valuable technique for identifying data relationships is correlation analysis. In this context, a correlation analysis has been performed 1) on the features within the SDWPF dataset and 2) on the time series of the target variable "Pavt" for various turbines. Pearson's correlation coefficient, denoted as r_{xy} for two features x and y, is expressed as:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4.2}$$

Correlation between features

Figure 4.2 illustrates the correlation matrix among the various features within the SDWPF dataset. It reveals that the correlation between wind speed and active power is notably high, at 81%. Additionally, there are other notable associations, such as the correlation between temperature and the day of the year, which is influenced by the annual cycle.

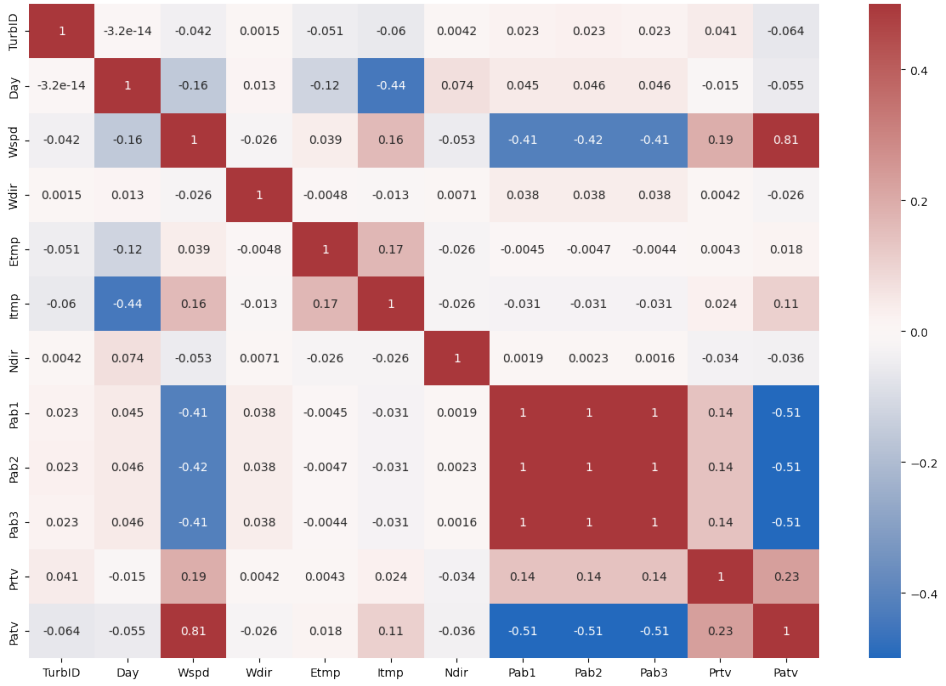


Figure 4.2: Pearson correlation matrix of wind turbine features. High values of correlation are highlighted by red boxes whereas anti-correlation is highlighted by blue boxes.

Correlation between turbines

An additional crucial analysis concerns the correlation between the active power generated by different turbines. This analysis aims to discern whether there are variations in the performance of individual turbines within the wind farm. In practice, wind turbines within a wind farm may exhibit different efficiency levels and some may be defective. Furthermore, their spatial placement within the farm can influence their behavior. Similar to the previous analysis, Pearson's correlation coefficient is employed. Moreover, hierarchical clustering is applied to identify groups of turbines with similar behavior at multiple levels. Figure 4.3 displays the

correlation matrix for all 134 turbines, with hierarchical groupings highlighted on the left and top of the image. The red areas highlight groups of turbines with strong correlations, while blue regions indicate isolated turbines with weaker correlations to the rest.

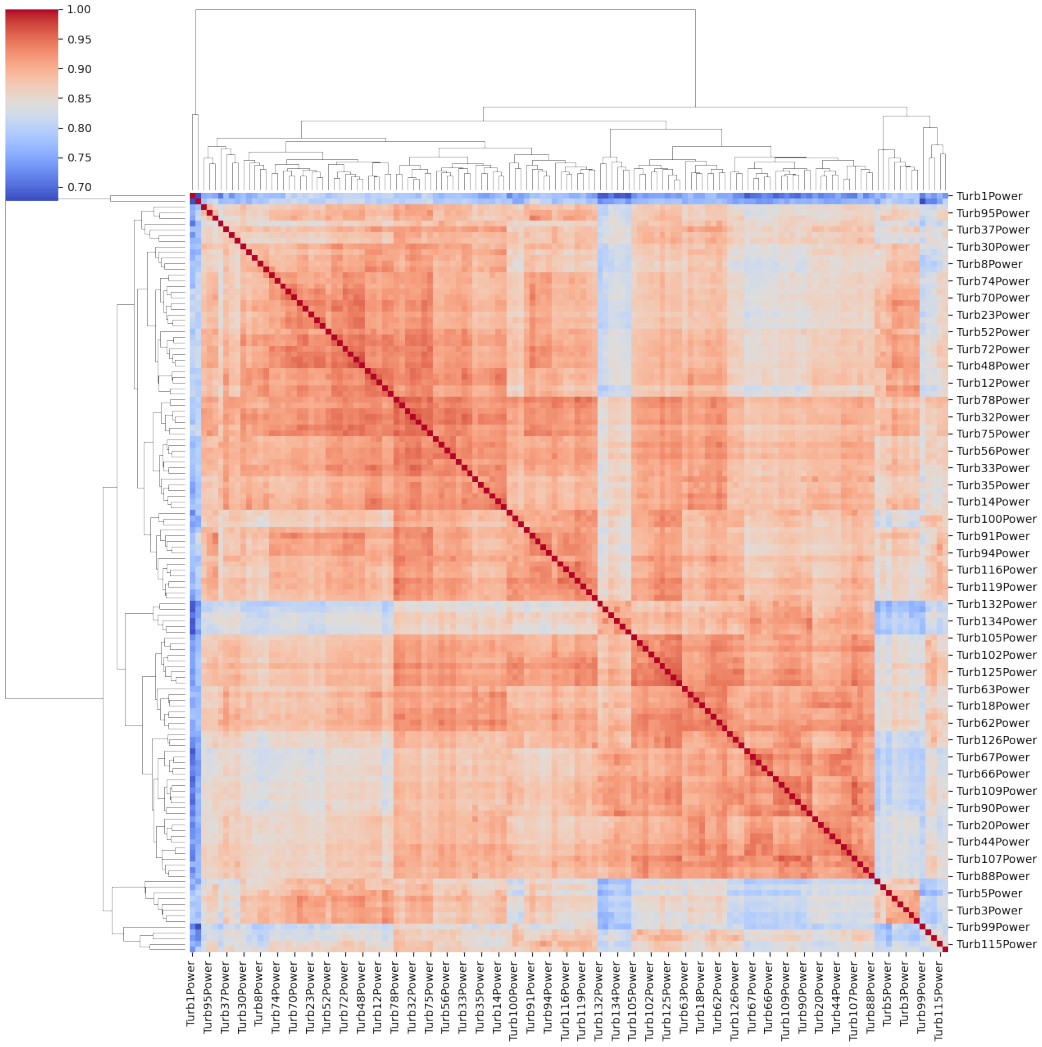


Figure 4.3: Pearson correlation matrix of active power of 134 turbines. The red areas highlight groups of turbines with strong correlations, while blue regions indicate isolated turbines with weaker correlations. The hierarchical trees on the leaf and top of the matrix are obtained with the average linkage method.

Spatial Correlation

To investigate whether the observed correlations between turbines are influenced by their spatial proximity, an analysis was conducted using hierarchical clustering with the average linkage method. The resulting clustering was truncated to focus on a limited number of clusters. A separate clustering was performed exclusively considering the distances between turbines. Subsequently, the results of the two clustering methods were compared (Figure 4.4), employing the adjusted mutual information metric, motivated by a potential unbalanced cluster with few elements [38].

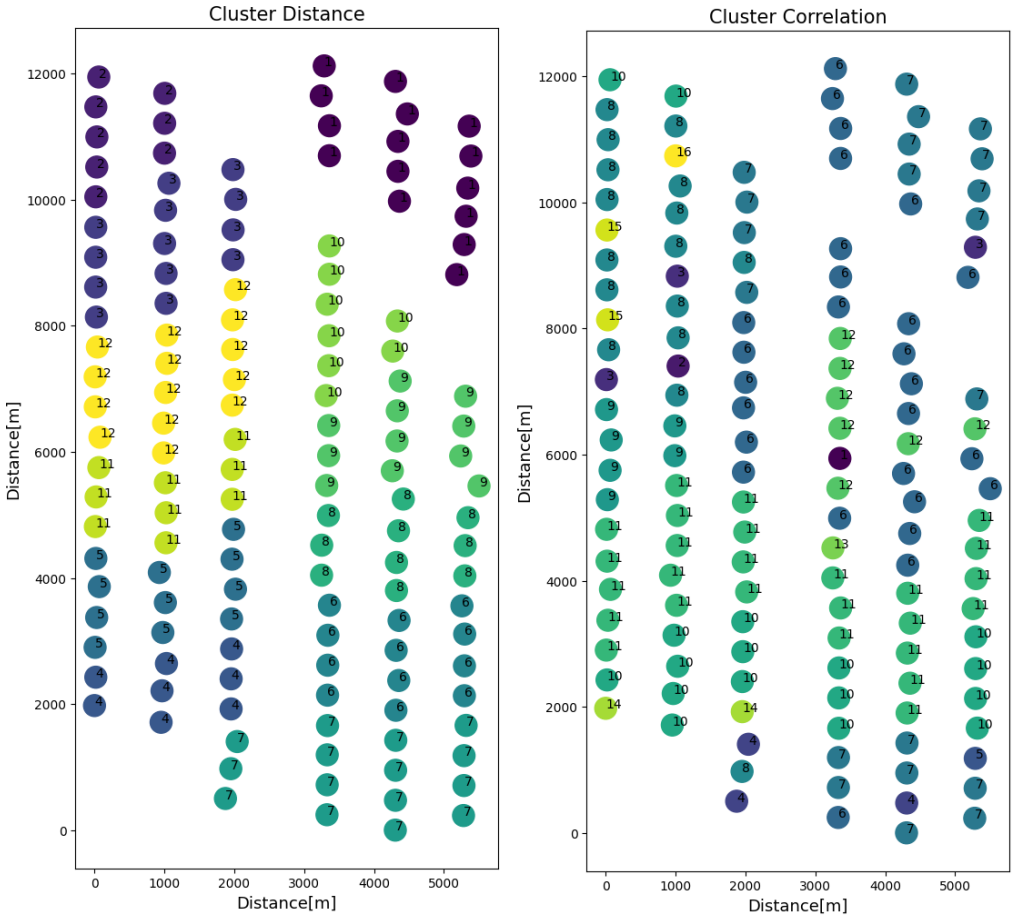


Figure 4.4: Left panel: spatial proximity hierarchical clustering of turbines obtained with the average linkage method. Right panel: correlation hierarchical clustering of turbines active power obtained with the average linkage method. Different clusters have different colors.

The comparison indicates that the correlation cluster exhibits a spatial component, as several neighboring turbines are grouped together (e.g., clusters 11 and 10 on the right). This observation is further substantiated by the adjusted mutual information value (ADI = 0.394). Nonetheless, there are instances of turbines showing correlations with others situated at a greater distance, suggesting that factors such as turbine height, placement within the wind farm, or turbine type (new or old) could also influence these correlations.

4.2 Data Cleaning

In the preliminary analysis of the dataset, it was determined that 23.93% of the data was unsuitable for use due to its classification into abnormal, missing, or unknown data categories. This represents a substantial portion of the dataset that cannot be utilized for training purposes. Consequently, a data recovery procedure was implemented to mitigate the large amount of unusable data. The distribution of the type of unusable data is summarized in Table 4.2.

Type of Data	Percentage
Missing Data	1.047%
Unknown Values	22.89%
Abnormal Values	0.002%
Union	23.93%

Table 4.2: Percentage of unusable data

The approach employed for data recovery is grounded in the concept of correlation clusters introduced in the previous section. Data points that fall into the categories of unknown, missing, or abnormal are substituted with the average value associated with the respective cluster to which the turbine is assigned at that specific time. This strategy is chosen due to the expectation that the average value from the relevant cluster provides a meaningful approximation of the wrong data point. In cases where an entire cluster contains exclusively unusable data, the recovery of data within that cluster is unfeasible. This data recovery process results in the preservation of a considerable number of records. Subsequently, a backfill process is employed to address any remaining missing points ("NAN"). The updated percentages of unusable data following the data cleaning procedure are presented in Table 4.3.

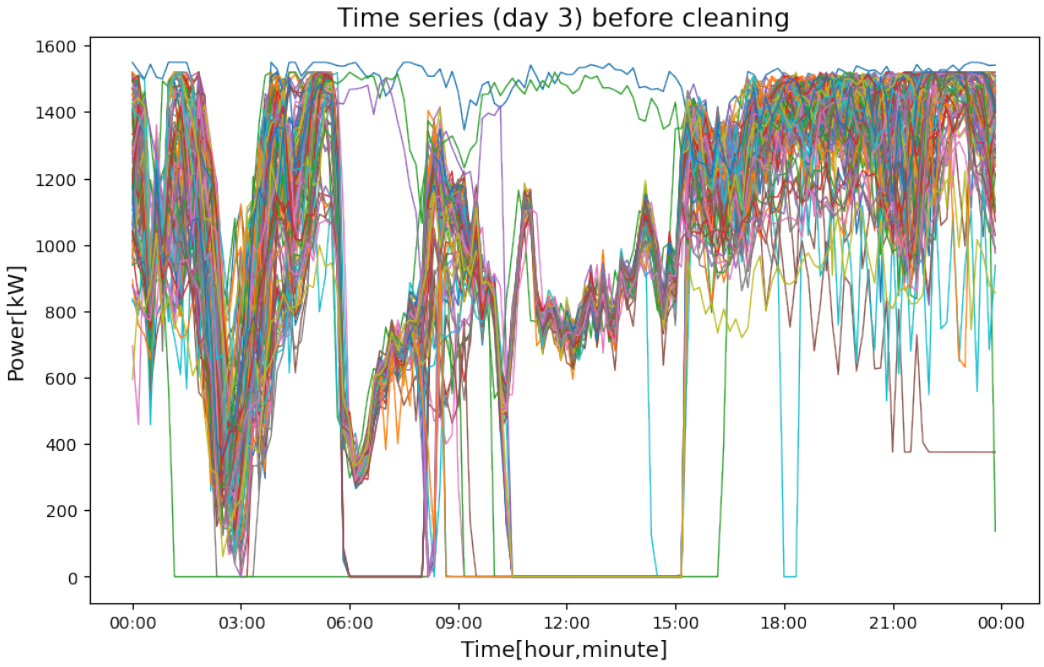


Figure 4.5: Time series of day 3 before cleaning

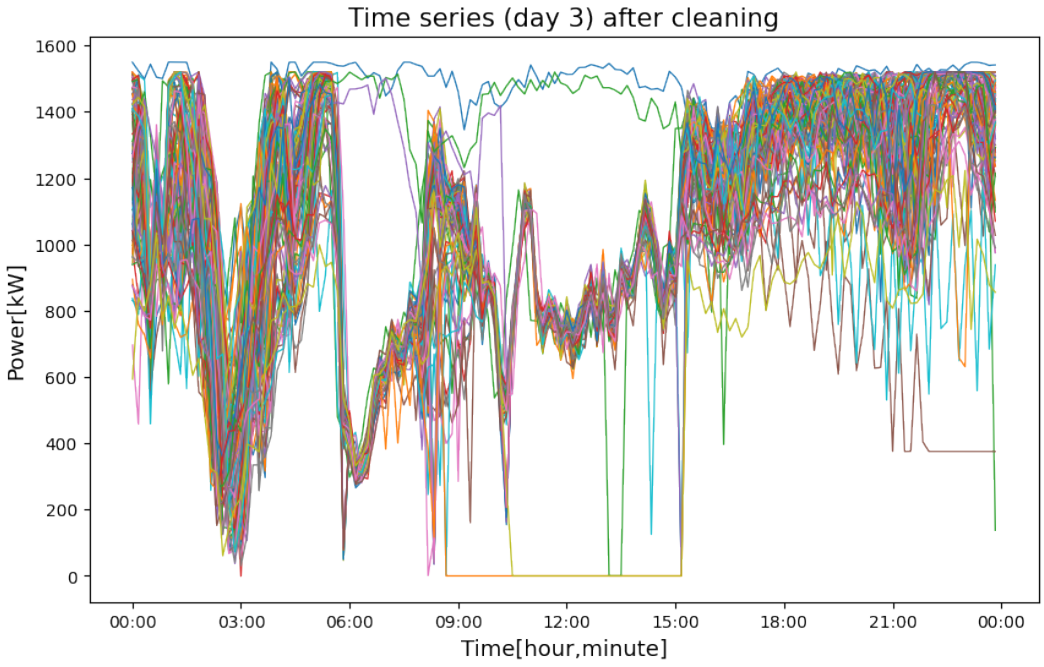


Figure 4.6: Time series of day 3 after cleaning

Type of Data	Percentage
Missing Data	0.00%
Unknown Values	14.40%
Abnormal Values	0.002%
Union	14.402%

Table 4.3: Percentage of unusable data after cleaning

Figures 4.5 and 4.6 provide visual representations of the data cleaning process's impact on reducing wrong values within time series data. Figure 4.5 depicts the time series data before cleaning. In contrast, Figure 4.6 illustrates the same time series data after the cleaning procedure, revealing an improvement in data quality and a reduction in errors.

Chapter 5

Methodology

In this chapter, we delve into the mathematical description and elaborate on the primary method utilized throughout this thesis. It starts by introducing the problem context associated with spatio-temporal forecasting, following the notation established by Cini et al. in their work on taming spatio-temporal data [39, 40]. Next, the core solutions employed to address the aforementioned problem are presented. Finally, we explain the details of two models designed to solve this problem.

5.1 Problem Statement

A dataset comprising 134 distinct time series is considered, each representing the active power generated by a different turbine. Each of these time series is composed of a sequence spanning T time instances. Consequently, we can define the matrix $X_{0:T-1} \in \mathbb{R}^{N \times T}$. This matrix encompasses the observed data from time $t = 0$ to $t = T-1$ for the N distinct time series. The interrelationship between the N turbines is encoded within an adjacency matrix $A \in \mathbb{R}^{N \times N}$. Moreover, should there be any additional features associated with each node, we can represent them through another matrix, constructed in a similar manner as X. This auxiliary matrix is labeled as $U_{0:T-1} \in \mathbb{R}^{N \times T \times F}$ with F denoting the number of additional features. Collectively, we can define the aggregation of these elements at a specific time t as $\mathcal{G}_t = \langle X_t, U_t, A \rangle$ which encapsulates all the information available at that time. The focal point of this thesis aligns with the KDD Baidu 2022 challenge [1]. The goal of the challenge is to tackle a multi-step-ahead time-series forecasting problem that entails predicting 288-time instances. Given the introduced notation, we seek to determine $X_{t:t+288} \in \mathbb{R}^{N \times 288}$ using $\mathcal{G}_{(t-w_{in}):t}$ as input, where w_{in} indicates the size of the input window used in predicting the output. The existing literature offers various methodologies to address problems of this nature, and an exploration of the

general framework involving Spatio-Temporal Graph Neural Networks (STGNN) is presented in the subsequent sections.

5.2 Spatio-Temporal Graph Neural Networks Framework

The fundamental concept behind Spatio-Temporal Graph Neural Networks is to extract both spatial and temporal information from the input data, represented as $\mathcal{G}_{(t-w_{in}):t}$. This extraction can be achieved through a variety of methods. In Cini et al. [40], an exposition on the general framework of graph-based models for temporal time series forecasting is elucidated, encompassing the following key operations:

$$\begin{aligned} h_{t-1}^{i,0} &= \textsc{Encoder}\left(x_{t-1}^{i}, u_{t-1}^{i}, v^{i}\right) \\ H_{t-1}^{l+1} &= \text{STMP}^{l}\left(H_{\leq t-1}^{l}, \mathcal{E}_{\leq t-1}\right) \quad l = 0, ..., L-1 \\ \widehat{x}_{t:t+H}^{i} &= \textsc{Decoder}\left(h_{t-1}^{i,L}, u_{t:t+H}^{i}\right) \end{aligned} \tag{5.1}$$

An encoding operation, $\textsc{Encoder}(\cdot)$ prepares the input for subsequent processing via a multilayer perceptron (MLP) layer. Similarly, a decoding operation $\textsc{Decoder}(\cdot)$ serves to generate the final prediction. Spatiotemporal message-passing operation $\text{STMP}(\cdot)$ extracts spatio-temporal features from the input. This operation is performed by several space-time layers ranging from 0 to L-1. The differentiating factor among models lies in the nature of the $\text{STMP}(\cdot)$ function. In general, it has the following structure:

$$h_{t}^{i,l+1} = \textsc{Up}^{l}\left(h_{\leq t}^{i,l}, \underset{j \in \mathrm{N}_t(i)}{\textsc{Aggr}}\left\{\textsc{Msg}^{l}\left(h_{\leq t}^{i,l}, h_{\leq t}^{j,l}, e_{\leq t}^{j,l}\right)\right\}\right) \tag{5.2}$$

Here, $\textsc{Msg}(\cdot)$ represents a message passing function, $\textsc{Aggr}(\cdot)$ represents an aggregation function while $\textsc{Up}(\cdot)$ represents an update function. Notably, this equation bears a resemblance to Equations 3.19 and 3.20, highlighting a common underlying paradigm. However, in the STMP operation, a temporal aggregation of data is introduced, distinguishing it from previous formulations. This temporal aggregation can occur before (Time-Than-Space), after (Space-Than-Time), or in non-separable phases with respect to spatial aggregation (Time-And-Space).

Time-Than-Space

In the Time-Than-Space (TTS) model, temporal aggregation takes precedence. Initially, temporal aggregation is carried out, after which the extracted hidden representations are employed for message passing, aggregation, and combination:

$$
\begin{aligned}
h_t^{i,1} &= \textsc{SeqEnc}\left(h_{\leq t}^{i,0}\right) \\
H_t^{l+1} &= \mathrm{MP}^l(H_t^l, \mathcal{E}_t)
\end{aligned}
\tag{5.3}
$$

Here $\textsc{SeqEnc}(\cdot)$ can be implemented using various sequence modeling architectures such as RNN, TCN, or attention-based methods. While $\mathrm{MP}(\cdot)$ is responsible for spatial aggregation and can be realized through architectures like Graph Convolutional Networks (GCN).

Space-Than-Time

Conversely, in the Space-Than-Time (STT) model, the temporal and spatial aggregation operations are reversed compared to the TTS model:

$$
\begin{aligned}
H_t^{i,l} &= \mathrm{MP}^l(H_t^{i,l-1}, \mathcal{E}_t) \\
h_t^{i,L} &= \textsc{SeqEnc}\left(h_{t-W:t}^{i,L-1}\right)
\end{aligned}
\tag{5.4}
$$

Time-And-Space

Time-and-space (T&S) models are the most prevalent in the literature due to their superior ability to concurrently capture spatial and temporal information. Various methods can be employed to implement T&S models, with the simplest approach involving alternating spatial and temporal aggregation operations several times [20]. Other solutions employ a single graph that connects present states to future states through the Cartesian product between graphs or the Kronecker product. These diverse approaches offer flexibility in modeling spatio-temporal data, allowing for tailored solutions to address specific forecasting challenges.

5.3 Spatio-Temporal Graph Convolutional Networks (STGCN)

Introduced by Yu et al. in 2017 [20], Spatio-Temporal Graph Convolutional Network (STGCN) is a graph-based model for spatio-temporal forecasting. Initially designed for traffic forecasting, this model has found applications in various research domains. Within the framework discussed in the preceding section, STGCN aligns with the Time-And-Space model paradigm. Its structure consists of two sequential convolutional blocks (as detailed in Section 3.6) and one spatial graph convolution layer (as explained in Section 3.8.1) as shown in Figure 5.1.

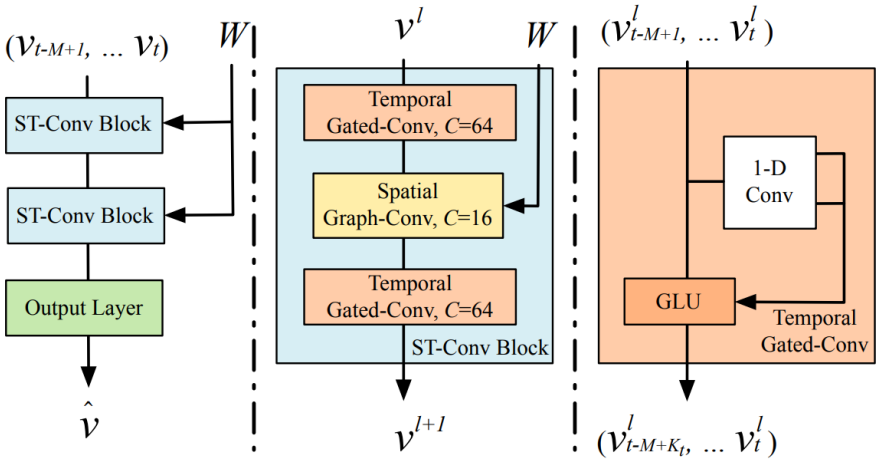


Figure 5.1: Structure of the spatial-temporal convolutional blocks as described in Yu et al. [20]

The STMP$(\cdot)$ function in this model is described by the following equation:

$$v^{l+1} = \Gamma_1^l * \tau \text{ReLU}\left(\Theta^l * \mathcal{G}\left(\Gamma_1^o * \tau v^l\right)\right) \tag{5.5}$$

In this equation, the two components $\Gamma * \tau$ correspond to a 1-D causal convolution (as outlined in Section 3.6) followed by a gated linear unit (GLU) activation function. The innermost term represents the initial temporal gated convolution, while the second term represents the convolution executed after message passing. The term $\Theta * \mathcal{G}$ corresponds to spatial message passing, as in Equation 3.22 but in the present setting it is expressed in spectral terms, with Θ representing the kernel of the spatial convolution.

5.4 Graph WaveNet

Graph WaveNet is a computational model introduced by Wu et al. [3] in 2019, taking inspiration from prior work by Oord et al.[10]. The model is structured according to Equation 5.1 and includes an encoder with a simple linear layer for linear data transformation, followed by the stacking of L layers of spatiotemporal message-passing blocks. The decoder consists of two linear layers with Rectified Linear Unit (ReLU) activation functions. All hidden states from the L layers are connected to the decoder through skip connections, enabling the model to handle spatiotemporal information at multiple levels of aggregation and avoiding gradient issues. The spatiotemporal message-passing module is composed of a

Graph Convolutional Layer (GCN) and a Gated Temporal Convolutional Layer (Gated TCN) as shown in Figure 5.2.

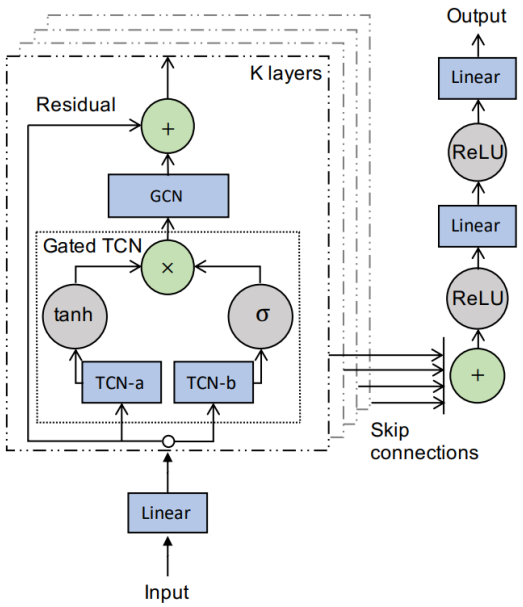


Figure 5.2: Graph WaveNet model structure

Graph convolutional layer

In the Graph Convolutional Layer, the authors introduced a mechanism to enable the model to autonomously form a self-adaptive adjacency matrix. By using Chebyshev polynomials approximation, it is possible to approximate the spatial convolution $\Theta * \mathcal{G}$ as:

$$\Theta * \mathcal{G} \approx \sum_{l=0}^{L} P^l X W_l \tag{5.6}$$

Here, X represents the input matrix, and $P = A/rowsum(A)$. The authors proposed an enhancement by adding a new element in the factorization in equation 5.6:

$$\Theta * \mathcal{G} \approx \sum_{l=0}^{L} P^l X W_{l1} + \widetilde{A}_{apt}^l X W_{l2} \tag{5.7}$$

The matrix $\widetilde{A}_{apt}$ is constructed as follows:

$$\widetilde{A}_{apt} = SoftMax(ReLU(E_1 E_2^T)) \tag{5.8}$$

where $E_1, E_2 \in \mathbb{R}^{N \times c}$ are randomly initialized node embedding dictionaries with learnable parameters. Through this self-adaptive adjacency matrix, the model can learn additional relationships between various nodes.

Gated Temporal Convolutional Layer

For temporal aggregation, the model employs dilated causal convolution (see Section 3.6) with a gating mechanism:

$$h = tanh\left(\Theta_1 * \mathcal{X} + b\right) \odot \sigma\left(\Theta_2 * \mathcal{X} + c\right) \tag{5.9}$$

Here, $\sigma(\cdot)$ is a sigmoid function that regulates the information passed to the next layer. The terms $\Theta * \mathcal{X}$ represent dilated causal convolution with kernel Θ, and $\odot$ denotes element-wise multiplication. This gated mechanism facilitates the learning of complex temporal dependencies, especially in the case of very long sequences.

Chapter 6

Experiments and Results

In this chapter, the outcomes of the experiments are presented. The experiments were conducted utilizing computational resources provided by Google Colab and HPC@PoliTO [41]. The results were divided into short-term and long-term scenarios. The short-term context involves forecasting for 2 hours, 6 hours, and 12 hours (equivalent to 12, 26, and 72 time steps, respectively). In contrast, the long-term context pertains exclusively to the 48-hour forecast. Before performing the main experiments, an experiment concerning the relationship between wind speed and active power was conducted.

6.1 Wind role for power forecasting

This initial experiment aims to show if wind speed can serve as a proxy measure of the active power generated by wind turbines. Typically, a representation of the relationship between wind speed and active power resembles the scheme in Figure 6.1 and Figure 6.2 with dataset data. The left part of the plot displays a preliminary cut-in region where the wind is insufficient to set the turbine rotor in motion. This is followed by a transition phase in which power production increases with wind speed, culminating in the rated power region, where the wind turbine operates at full capacity. Based on fluid dynamics theory, Betz's law [42] provides an upper limit on the power obtainable from an ideal rotor:

$$P_{max} = \frac{16}{27} \cdot \frac{1}{2} \rho S v^3 \tag{6.1}$$

Here, ρ represents air density, S corresponds to the rotor's surface area, and v is the wind speed. However, in practice, accounting for factors like friction, the power obtained from the wind is below this ideal limit. Additionally, the relationship between active power and wind speed in the transition phase is influenced by

several variables, including turbine height and the specific location of the wind farm. Research by Teyabeen et al. [43] investigates different functional relationships between wind speed and active power. Primarily they consider linear, quadratic, cubic, or exponential functional forms.

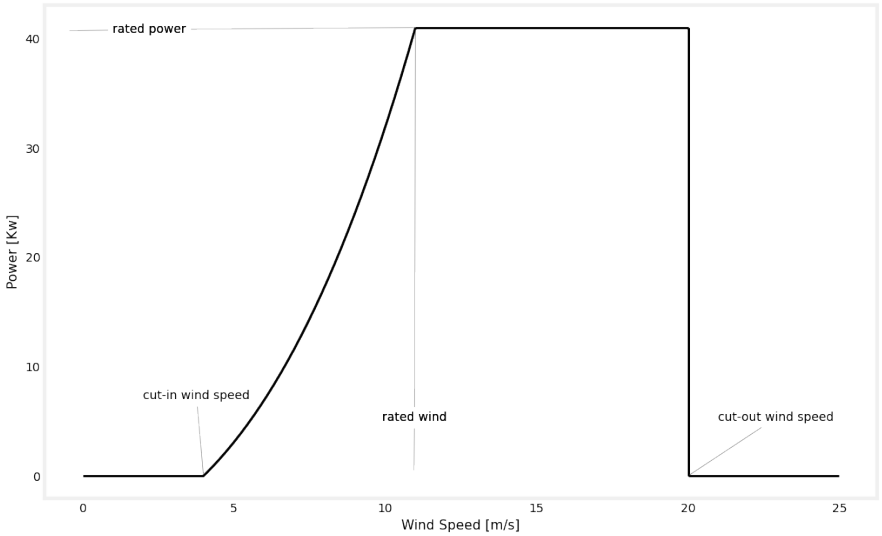


Figure 6.1: Scheme of wind power curve

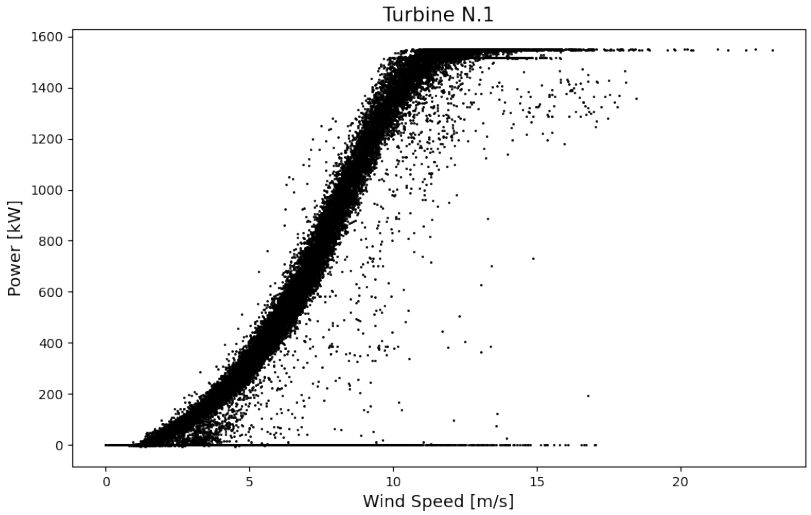


Figure 6.2: Scatter plot of wind power curve from data of turbine N.1

In this experiment, a quadratic form has been chosen to model the relationship between wind speed and power output during the transition phase. The relationship is then written as:

$$P = av^2 \tag{6.2}$$

Here, a represents a coefficient determined through linear regression without an intercept term.

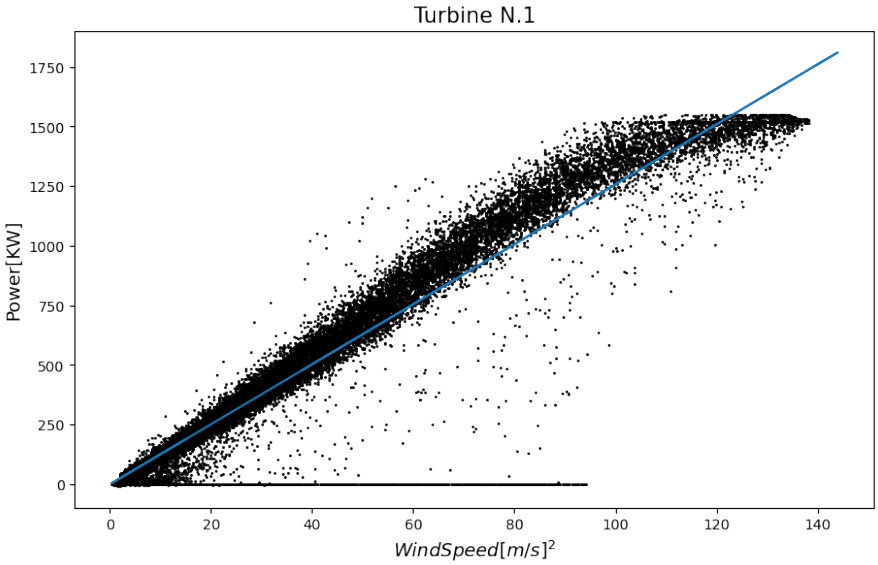


Figure 6.3: Regression between the wind power and the square of wind speed for data of turbine N.1. Black dots are the dataset points and the blu line represents the regression line

The regression was performed separately for each individual turbine and Figure 6.3 offers a visual representation of the quadratic fitting for turbine N.1. With these regression coefficients established for all turbines, the wind speed from time t to $t+T$ is used to predict the corresponding power output. Figure 6.4 shows the prediction for a step of 288-time records using wind speed as a proxy. Unsurprisingly, with knowledge of wind speed, predicting the power output of the wind farm becomes a relatively straightforward task. In fact, wind speed forecasting and wind power forecasting can be considered closely related and often interchangeable tasks.

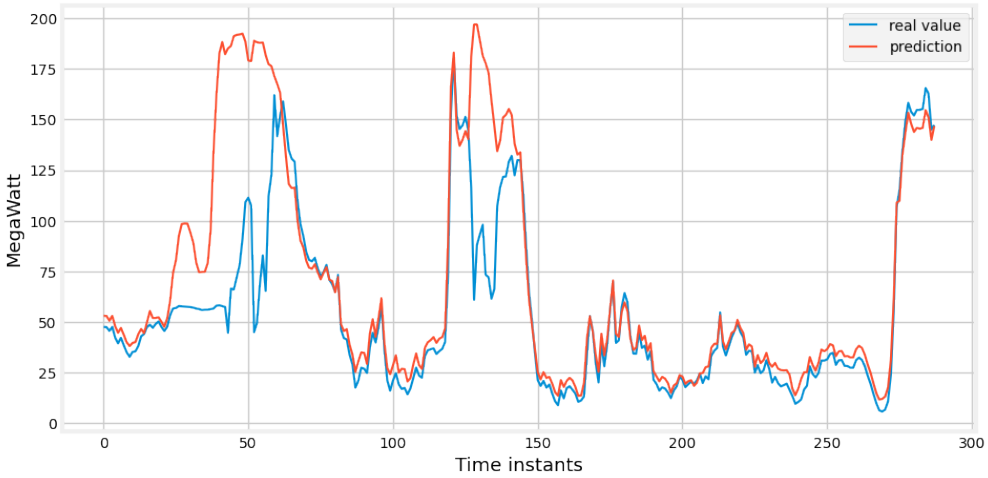


Figure 6.4: Computation of active power using wind speed data as input. Where the relationship between active power and wind speed is the one given in Equation 6.2

6.2 Short-Term Setting

In short-term experiments, forecasting trials with various time horizons were conducted, specifically targeting 2 hours, 6 hours, and 12 hours ahead. These experiments involved evaluating the performance of several models: 1) Lasso Regression, a GRU Network 2) with and 3) without additional features, and 4) Graph WaveNet with features data. In all cases, a fixed input window size w_{in} is set at 200 time instants. All the experiments follow an 80% training data and 20% validation data split.

6.2.1 Lasso Experiment

Lasso regression served as the primary focus of this experiment. Lasso for its fast execution allowed for the implementation of direct multi-step forecasting (as discussed in Section 3.1), where a unique model was created for each time step to be predicted. Furthermore, separate models were generated for all 132 wind turbines within the factory, leading to a total of 38,016 nested models. The experiment uses Grid Search techniques to optimize the regularization parameter α, resulting in an optimal value of $\alpha = 0.0215$ and for this particular experiment, active power data is used without the inclusion of additional features. To perform these preprocessing and training tasks Skforecast library [44] is used. The Lasso Regression model achieved its best forecast with the following error scores as defined in Chapter 4: 21.73 for the 2-hour forecast case, 32.3 and 39.17 for the 6-hour and 12-hour

forecast cases respectively. This outcome establishes a solid baseline for subsequent models, given that Lasso regression, while being a simple model, effectively utilizes regularization to capture time series trends and averages. For illustrative purposes, Figures 6.5 and 6.6 display a 12-hour forecast encompassing 72 time steps. The figures show that the models manage to follow the average values quite well.

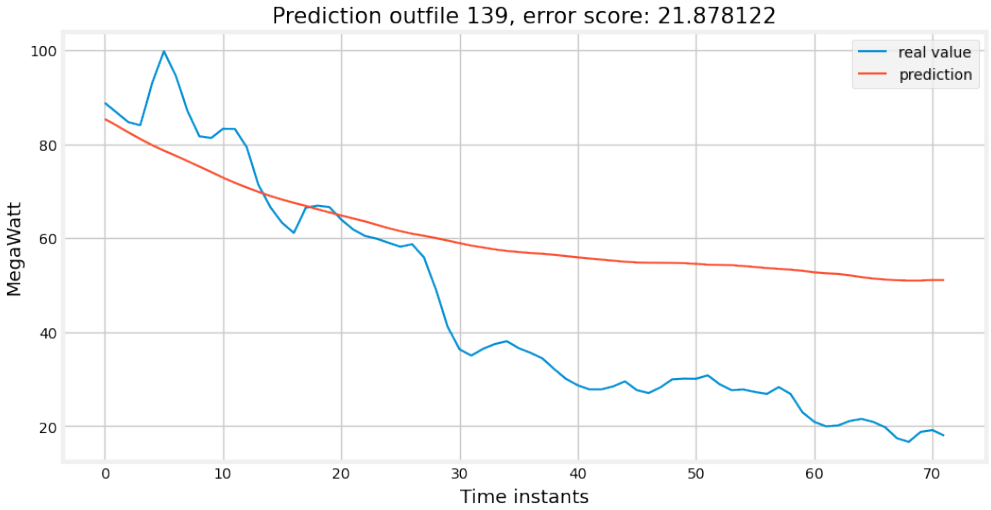


Figure 6.5: Example of prediction of Lasso regression for outfile N. 139

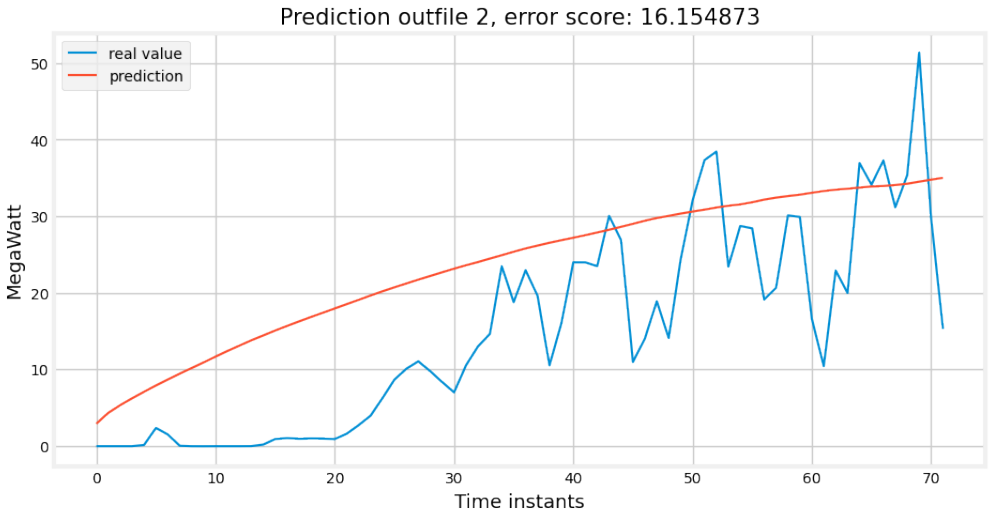


Figure 6.6: Example of prediction of Lasso regression for outfile N. 2

6.2.2 GRU Experiment

In this experiment, a Gated Recurrent Unit (GRU) neural network was employed to capture more intricate relationships within sequential data. The choice of GRU over traditional Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks was guided by its swifter training process and superior performance compared to a standard RNN. Unlike Lasso's previous experiment, different models were not created for different turbines but a single model was trained for all the turbines.

Model Structure

Regarding the architectural configuration of the model, various setups were tested. Initially, a model with two GRU layers and a Fully Connected layer was examined. This configuration exhibited better results than a Lasso model but still did not meet the desired score result for solving the problem in comparison to the literature. Consequently, the final chosen configuration entailed two GRU layers and two Multi-Layer Perceptron (MLP) layers employing the Rectified Linear Unit (ReLU) activation function (see Figure 6.7).

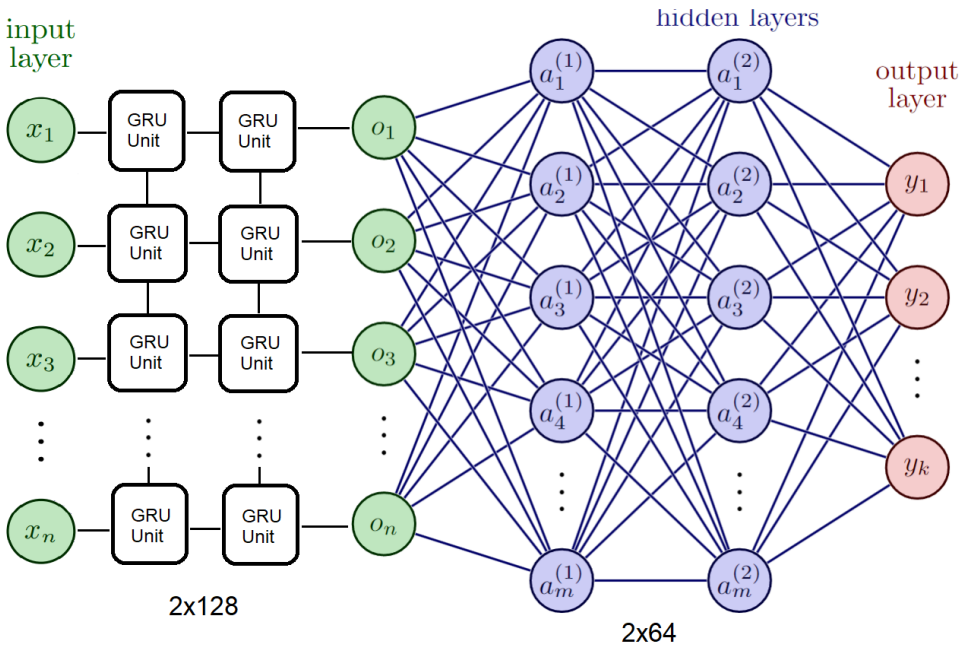


Figure 6.7: Network configuration for GRU Experiment. This network structure presents 2 GRU layers of 128 units (left part) and 2 fully connected layers (right part) of 64 units.

The output from the GRU layers serves as input to two Fully Connected layers. In terms of network dimensions, the two GRU layers each comprise 128 neurons, while both fully connected layers consist of 64 neurons. Adjusting the network's width proved crucial, as reducing it decreases the model's ability to capture intricate patterns while enlarging it prolonged the training without substantial performance gains. Furthermore, a dropout probability of 0.3 was used to make the network more robust to overfitting.

Training and Loss Function

For the training phase, an initial experiment utilized the Mean Absolute Error (MAE) as the loss function:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{6.3}$$

However, this approach yielded unsatisfactory results due to the irregular nature of wind data, which often exhibits behaviors far from the average. To address this issue, the Mean Squared Error (MSE) loss function was considered:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6.4}$$

Nonetheless, MSE is highly sensitive to outliers. Consequently, the Huber Loss was identified as a suitable compromise, as it combines elements of both MSE and MAE:

$$\begin{aligned} H(y_i, \hat{y}_i) &= \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & for\,|y_i - \hat{y}_i| \leq \delta \\ \delta(|y_i - \hat{y}_i| - \frac{1}{2}\delta) & otherwise \end{cases} \\ Huber &= \frac{1}{n}\sum_{i=1}^{n} H(y_i, \hat{y}_i) \end{aligned} \tag{6.5}$$

The chosen delta value for Huber loss was set to 5 following the paper by Jiang et al.[28]. Given that the data underwent standard scaling, this implies that only data points exceeding 5 times the standard deviation above the mean are linearly weighted. Another critical aspect concerning the loss function involved the inclusion of an L2 regularization term:

$$L2term = \frac{\lambda}{2n}\sum_{w} w^2 \tag{6.6}$$

This term, reminiscent of the regularization term in Lasso regression, encourages models with lower weights, thus mitigating overfitting, which is a common issue in this problem. The PyTorch library accommodates the use of the weight decay

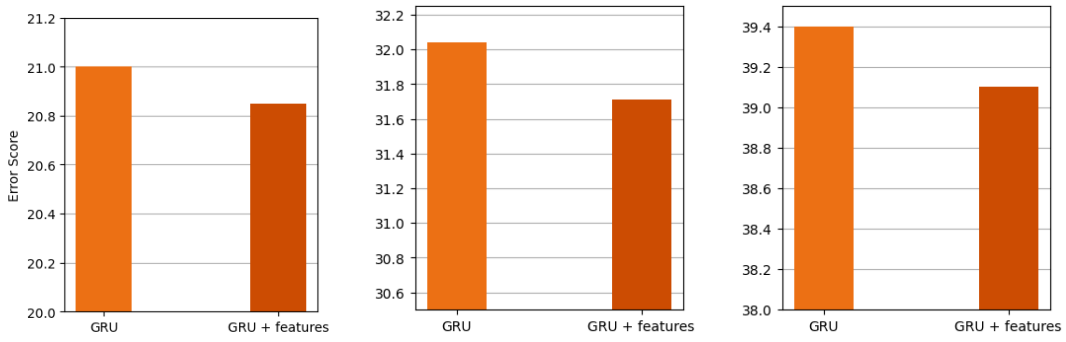
parameter, with a chosen value of 0.001. Values lower than this would lead to overfitting, while higher values would overly simplify the model. The gradient descent optimization was performed using the Adam Optimizer, and the learning rate was set at 0.0001.

Features

Two distinct experiments were conducted, differing in the features used as input to the network. In one experiment, only the target variable, the active power, was utilized. In the other, two additional features are incorporated. These additional features were wind speed, which was identified in Section 6.1 as highly indicative of the power trend, and a cyclical covariate aiding the model in recognizing the correct daily cycle. The construction of this cyclical covariate is as follows:

$$\begin{aligned} hm &= 60 \cdot hour + minute \\ sine\_feature &= \sin(hm \cdot \frac{2\pi}{1440}) \end{aligned} \tag{6.7}$$

Here, *hour* and *minute* are the timestamps of the considered record, while $sine\_feature$ is computed using a sine function to capture cyclical patterns over the course of a day. The results obtained with the use of features are better in every forecast horizon compared to those without features (see Figure 6.8)



(a) Error score 2 hours **(b)** Error score 6 hours **(c)** Error score 12 hours

Figure 6.8: Comparison between GRU results with features and without features. (a) panel shows results for 2 hours predictions, (b) panel for 6 hours, and (c) panel for 12 hours.

Other experiments were carried out using other features such as the direction of the turbine axis (Ndir) and the ID number of turbines (TurbID), without arriving at superior results. Figure 6.9 shows the forecasting of the model with the features for the outfile N.139. By making a comparison with Figure 6.5 it is possible to

see how the model is able to better understand the trend of the curve. It is also important to note that more complex models, although performing better, never deviate much from a solution that is close to the average value given by Lasso's prediction due to the complexity of the problem.

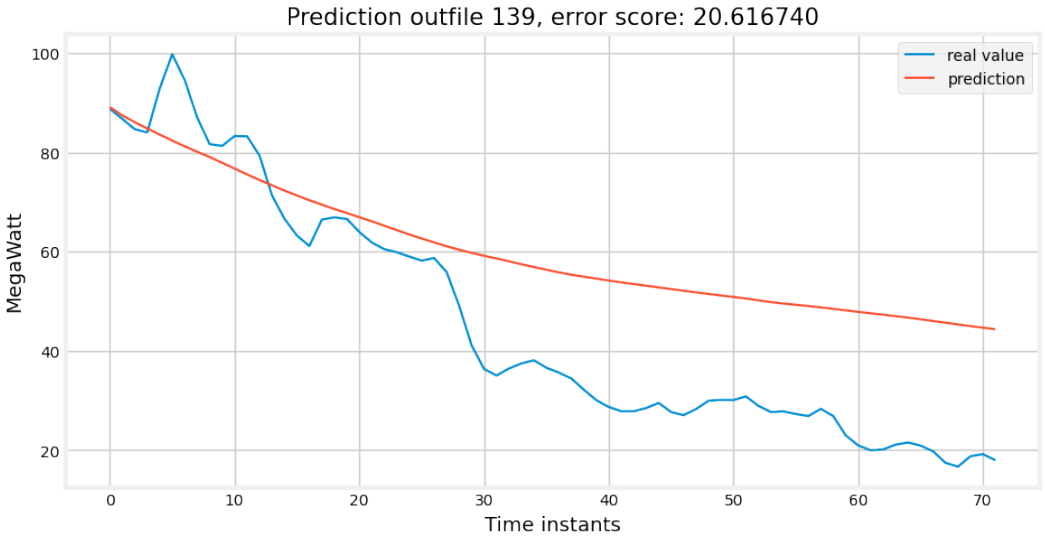


Figure 6.9: Example of prediction of GRU model for outfile N. 139

6.2.3 Graph WaveNet Experiment

The Graph WaveNet model has proven to be the most effective model in the context of this thesis, demonstrating superior performance in wind power forecasting for both short-term and long-term horizons. As detailed in Section 5.4, this model exhibits the remarkable capability to autonomously discern relationships among the various wind turbines. The implementation of Graph WaveNet utilized in this study is the one provided by the Tsl (Torch Spatiotemporal) library, as proposed by Cini and his team in their 2022 publication [45]. The input features for this experiment are consistent with those employed in the case of the GRU model, comprising wind speed and the cyclical pattern over the course of a day as a covariate.

Graph Structure

The model needs as input an adjacency matrix that describes the spatial relationship between the various turbines. In this graph representation, each node corresponds to a distinct turbine, and the connections between nodes describe their proximity

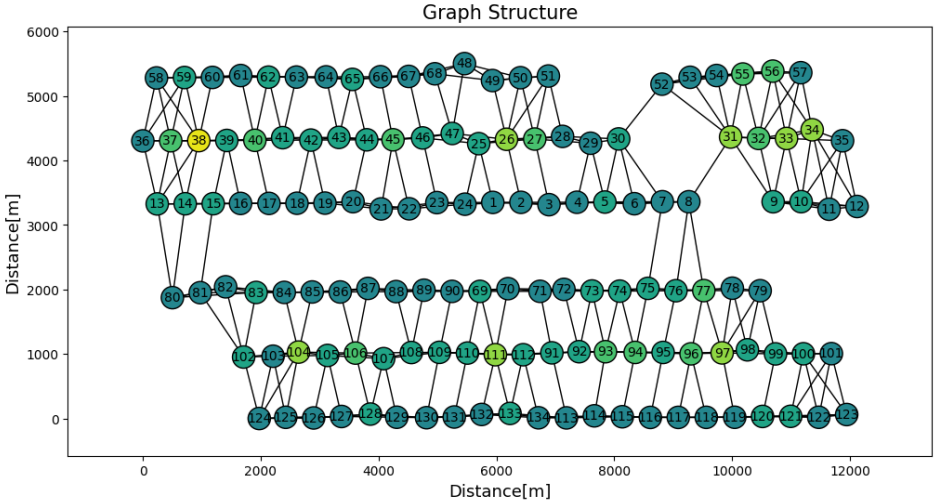


Figure 6.10: First tested graph structure of wind factory. Unweighted edges are used and each node connects to its five closest nodes. When the distance between a node and other nodes is the same the degree is slightly larger than five. The colors indicate the degree centrality of each node.

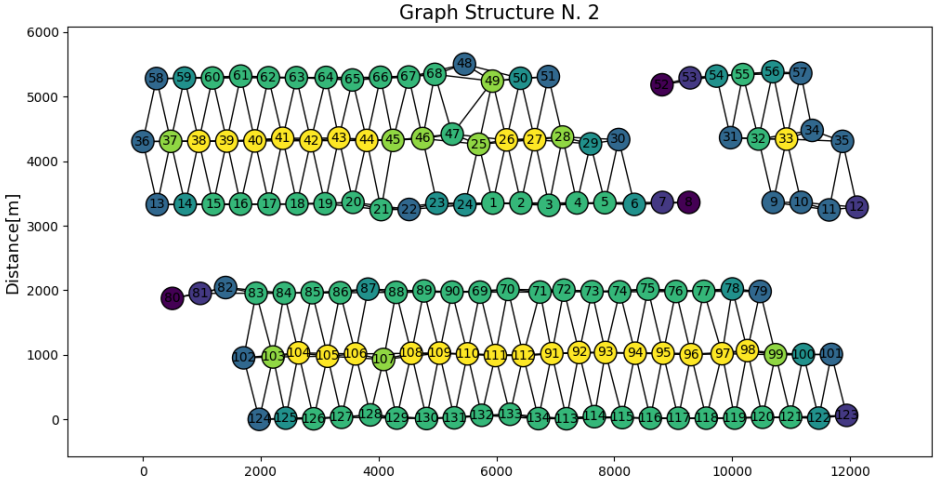


Figure 6.11: Second tested graph structure of wind factory. Two nodes are connected if they are within a radius of 1100 meters. The colors indicate the degree centrality of each node.

to one another. Two different configurations were tested in the experiments. In the first configuration, unweighted edges connect each node to its five closest nodes. This configuration is shown in Figure 6.10. For the majority of nodes, the degree centrality in this graph is equal to five, whereas in the case the distance between a

node and two or more other nodes is the same the degree is slightly larger than five. A second configuration was tested in which two turbines are connected if they are within a certain proximity radius. The radius was set at 1100 meters. The value of 1100 meters was chosen so that the average degree of the corresponding graph was as close as possible to the average degree of configuration number one. In this case, weighted edges are constructed as follows.

$$w^{i,j} = \begin{cases} exp(-\frac{dist(i,j)}{\gamma}) & dist(i,j) \leq 1100 \\ 0 & otherwise \end{cases} \tag{6.8}$$

where $dist(i,j)$ is the distance between i-th and j-th node and γ controls the kernel width. The representation of the graph is shown in Figure 6.11. The colors of the graph show the degree centrality of the nodes.

Dilated convolution

Graph WaveNet uses a dilated convolution mechanism to handle temporal aggregation (see Paragraph 3.6). In fact, it is essential to set the parameters of the dilated convolution so that it is possible to consider the correct receptive field. The Tsl (Torch Spatiotemporal) library provides two parameters to handle dilated convolution, `dilated` and `dilated_mod`. Referring to Equation 3.15, the first parameter controls the filter size k while the second manages the dilation factor d. In all experiments, the value of `dilated` is set to 2 while the value of `dilated_mod` is set to 4. The result for each temporal filter is similar to the one shown in Figure 3.9.

`hidden_size`	32
`n_layers`	10
`ff_size`	256
`emb_size`	10
`temporal_kernel_size`	2
`spatial_kernel_size`	2
`dilation`	2
`dilation_mode`	4
`dropout`	0.3

Table 6.1: Hyperparameters chosen for all the experiments done with the Graph WaveNet model.

Hyperparameters setting

Since the setting of the training hyperparameters for the GRU network was satisfactory, for Graph WaveNet, a similar configuration was used. The network utilizes a Huber Loss function with a delta value set to 5 while employing a weight decay of 0.003 and a learning rate of 0.0003. The experiment incorporates ten Graph WaveNet blocks, with a temporal convolutional kernel dilation of two and a dilation mod of four. The number of features is 10 in the node embeddings used for graph learning, while the number of units in the nonlinear readout is 256. Also in this case the dropout probability was set at 0.3. Table 6.1 shows in detail all the hyperparameters of the network by using the variable names used in the Tsl library (Torch Spatiotemporal) [45].

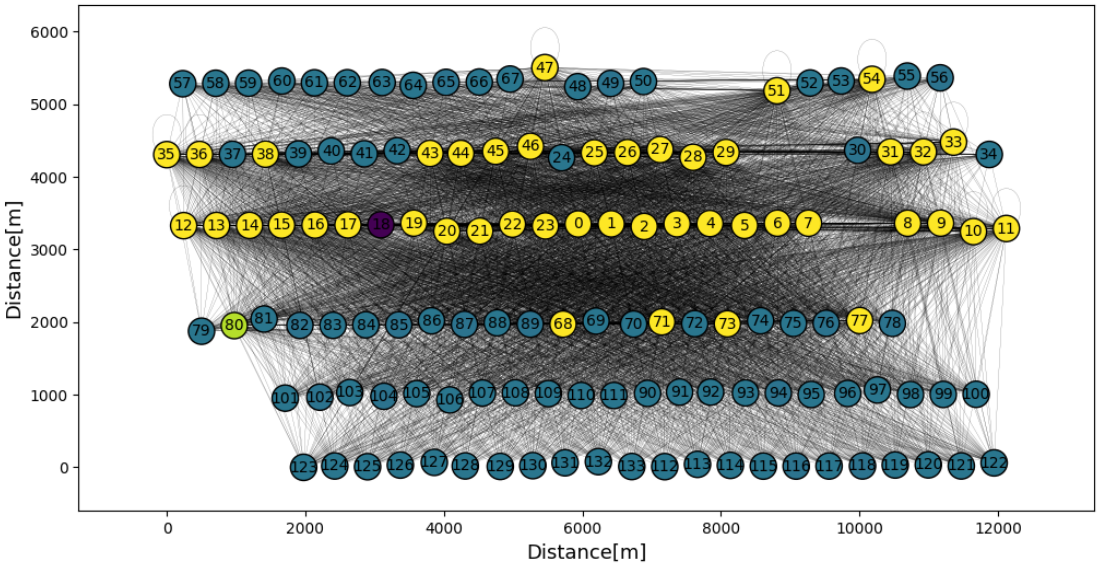


Figure 6.12: Graph structure from the self-adaptive adjacency matrix. The structure is learned from the model capturing additional connections between nodes. The color indicates the degree centrality of nodes. Yellow nodes have a higher degree than blue nodes.

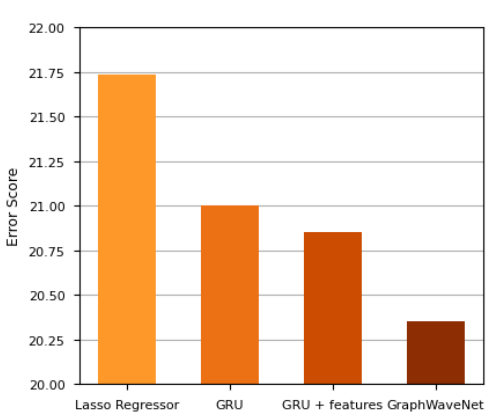
Self-adaptive adjacency matrix

In Section 5.4, we explored the Graph WaveNet model, which allows us to learn additional connections between nodes beyond what is already present in the initial adjacency matrix. During training, the model generates a new self-learned adjacency

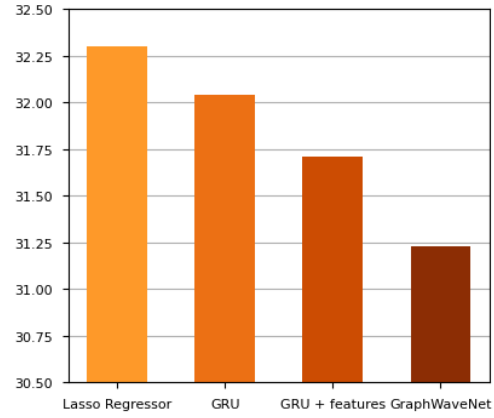
matrix. This matrix is then utilized in the inference phase alongside the initial adjacency matrix linked to a spatial structure. The exam of the self-learned adjacency matrix reveals a distinct grouping of turbines. Specifically, turbines with TurbID from 1 to 45 form a strongly connected group, while the remaining turbines constitute a weakly connected second group. To better understand the relationships learned independently by the model, we can visualize the corresponding connecting graph derived from the self-adaptive adjacency matrix. Figure 6.12 displays this graph, with colors indicating the degree centrality of the nodes. Yellow nodes have a higher degree centrality while blue nodes have a lower one. The distinct colors highlight the two turbine groups mentioned earlier. Although interpreting the self-adaptive adjacency matrix is challenging due to the inherent complexity of the deep learning model, we can hypothesize that the model identified relationships between turbines not explicitly present in the original data. For instance, it may have uncovered connections related to turbine height or wind exposure. The graph further illustrates that the connected subgraph primarily comprises turbines from the third row from the top, showing among other things a spatial component.

6.2.4 Short-Term Results

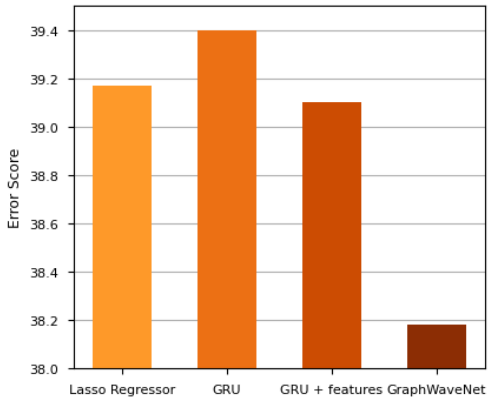
This section provides a comparative analysis of various models, the performance of models is illustrated in Figure 6.13. This figure highlights that Graph WaveNet consistently outperforms the other models in the short-term forecasting horizons, for 2 hours forecasting, 6 hours, and 12 hours. This superiority can be attributed to the model's capacity to leverage spatial information for enhanced forecasting precision and the model's ability to capture further relationships between turbines as discussed in the previous paragraph. For the short-term experiment, the first graph structure (Figure 6.10) described in the previous paragraph was used. Notably, Graph WaveNet exhibits a reduction in error score of more than one unit in comparison to Lasso for each short-term forecasting horizon. It is worth highlighting that GRU and Lasso error scores tend to converge as the forecasting horizon extends. In fact, in the 12-hour forecast, GRU without the features works slightly worse than a Lasso regression. Graph WaveNet maintains its superiority over Lasso in the mid-term, affirming its ability to focus on specific elements for short and mid-term forecasts without compromising the accuracy of average predictions. Based on these results it is possible to say that Graph WaveNet manages spatial and temporal information better than the models seen previously in wind power forecasting task.



(a) Error score in forecasting 2 hours



(b) Error score in forecasting 6 hours



(c) Error score in forecasting 12 hours

Figure 6.13: Comparison between models performances in the short term setting. Error score values are on the y-axis. (a) panel reports comparison for 2-hour forecasting, (b) panel for 6-hour forecasting, and (c) panel for 12-hour forecasting.

6.3 Long-Term Setting

When dealing with long-term forecasting, a distinct set of considerations comes into play. It is important to note that there is not a single model that significantly outperforms all others in this context. This observation has been reinforced by several studies related to the KDD Baidu Challenge, as indicated in the works of Huerta et al. [26], Zhao et al. [27], and Liu et al. [46]. In fact, when attempting to forecast over a horizon of 288 time intervals, a single model encounters various challenges in simultaneously capturing short-term and long-term patterns. To address this issue, proposed solutions include adjusting the granularity of the forecast, as suggested by Zhao et al. [27]. This is done by considering fewer time

intervals, or employing ensemble models that differentiate forecasts for the short term and long term, as demonstrated in the works of Huerta et al. [26] and Liu et al. [46].

6.3.1 Performance of Individual Models

A comparison of models in the long-term forecasting scenario reveals that the GRU network, both with and without features, underperforms in comparison to a Lasso regression. This discrepancy can be attributed to the challenges that RNN models typically face when dealing with forecasts over such extended timeframes, as noted by Ismail et al. [47]. Graph WaveNet model achieves a score error of 45.40 exhibiting a limited improvement of only -0.35 compared to a Lasso regression (see Figure 6.14). This limited improvement motivates us to investigate how to further reduce the model's prediction error.

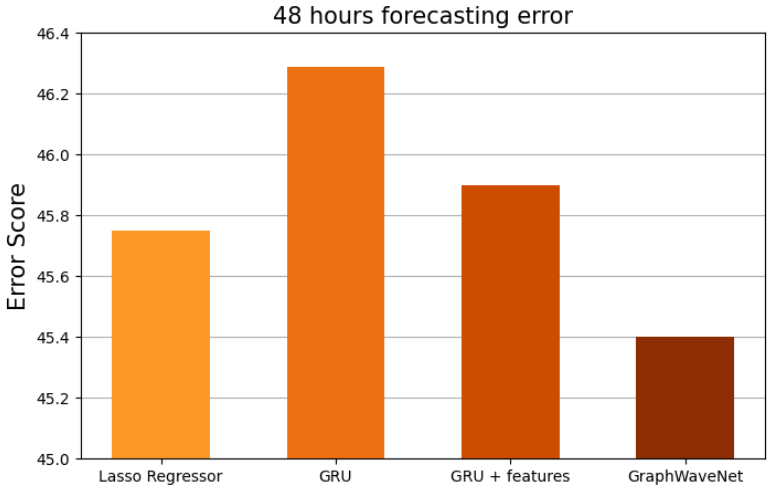


Figure 6.14: Comparison between the performance of models in a long-term setting. Error score values are on the y-axis. The forecasting period is 48 hours.

Figure 6.15 shows the error score of the Lasso and Graph WaveNet models as a function of time and it illustrates the relatively small differences between these models over time. Lasso appears to perform well in the mid-term, while Graph WaveNet excels in both short-term and long-term forecasts. When the graph includes a model trained exclusively for the first 72 time intervals in the short-term setting, it becomes evident that this approach outperforms the models trained for 288-time intervals. This observation supports the idea of employing an ensemble model to separately handle short-term and long-term predictions.

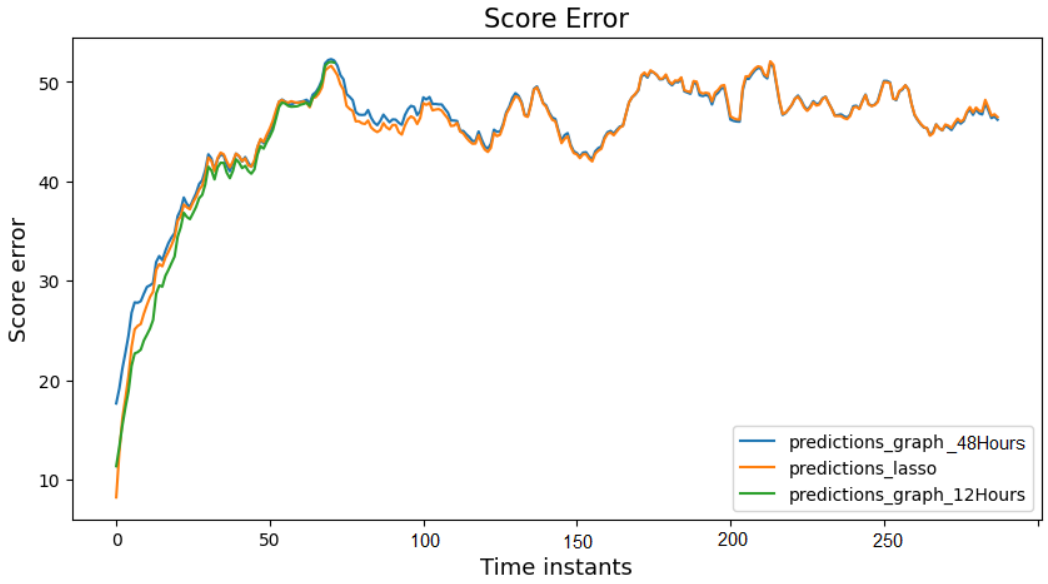


Figure 6.15: Error score as a function of the predicting time instants for Graph WaveNet 48 hours (blue line), Lasso regression (orange line) 48 hours and Graph WaveNet 12 hours (green line).

6.3.2 Ensemble Model

Drawing inspiration from the approach proposed by Huerta et al. [26], an ensemble model that combines short-term and long-term forecasting is developed. This model considers the predictions made by the Graph WaveNet model in the short-term setting for the first 72 time intervals and by a different Graph WaveNet model for the long-term setting spanning the entire time interval. The models are trained separately and with a different graph configuration.

For the short term, the model uses the graph structure that connects the 5 closest turbines (Figure 6.10), while for the long term, the model uses the second configuration which considers turbines connected within a radius of 1100 meters (Figure 6.11). The best model for both cases was found by training for 30 epochs and taking the model that performs best on the validation set. This ensemble model improves forecasting by giving a score reduction of -0.09 compared to a single Graph WaveNet model, leading to an error score of 45.31. This result appears to be very close to that obtained by the best model of the Baidu KDD 2022 challenge (44.91), placing ninth on the challenge leaderboard.

Figure 6.16 shows an overview of the performance of all the models used for 48-hour forecasting together with the best result of the challenge obtained by Li et al. [29].

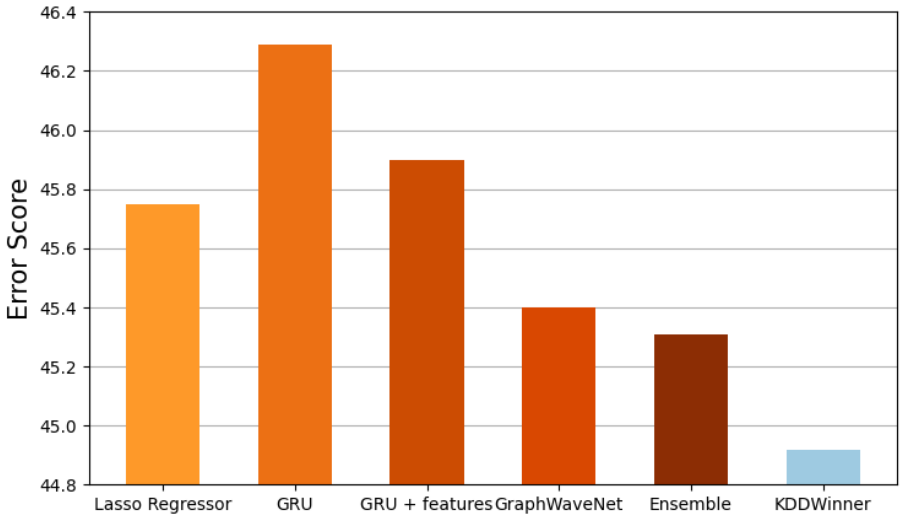


Figure 6.16: Overview of the performance of all models for 48-hour forecasting. Error score values are on the y-axis. KDDWinner error score is reported in blue color for comparison with the model investigated in this thesis.

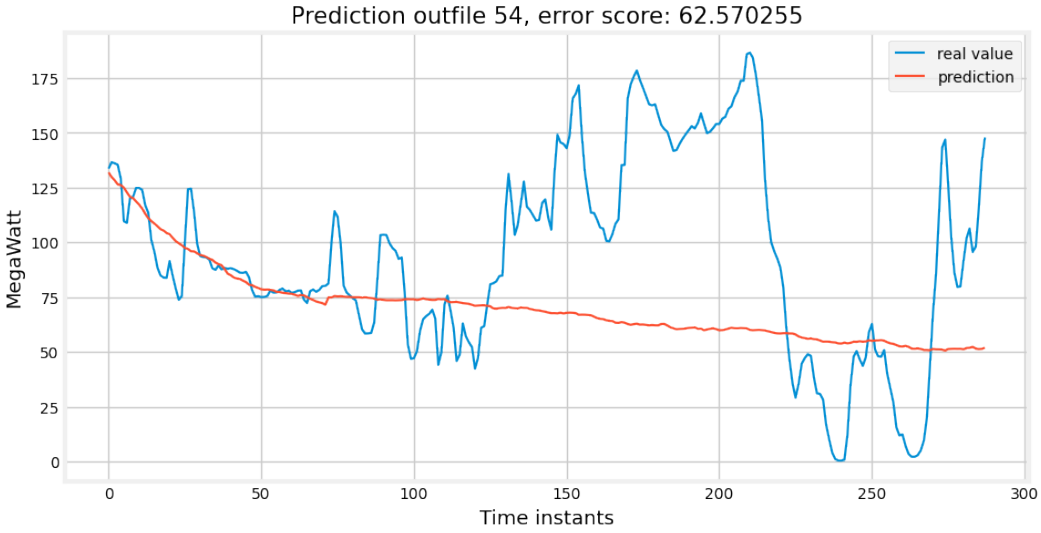


Figure 6.17: Prediction of ensemble model over 48 hours for outfile N.54. The red line is the model prediction while the blue line is the real-time series.

Figure 6.17 shows the forecast obtained by the ensemble model for the outfile N.54, the forecast is more accurate for the first 72 time instants as it uses the model in the short-term setting. For the remaining time instants, the forecast of the long-term setting is used which focuses mainly on the average value of the

trend of the curve. Additionally, Figure 6.18 shows that when the time series to be predicted is highly intermittent, when for example it is a period of turbolence, the short-term prediction fails to capture local patterns and aligns with long-term average forecasts. In such instances, the ensemble model predicts an average value, missing local fluctuations in the curve.

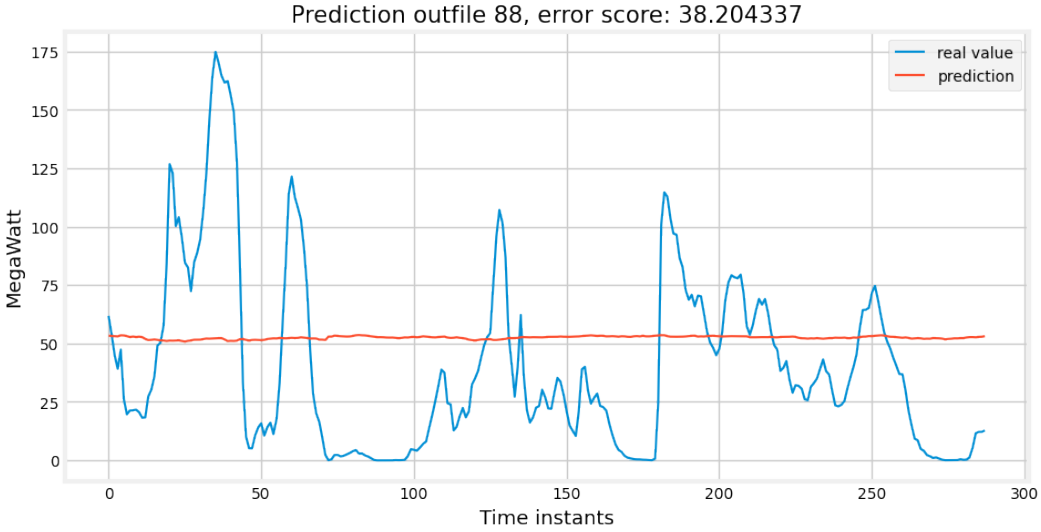


Figure 6.18: Prediction of ensemble model over 48 hours for outfile N.88. The red line is the model prediction while the blue line is the real-time series.

6.3.3 Summary of all results

In conclusion, this section summarizes all obtained results. Table 6.2 presents a comparison of error scores for forecasting horizons of 2, 6, 12, and 48 hours, respectively. The table includes all investigated models, with the number of time intervals to predict, and the best result highlighted in bold characters. The Lasso regression model serves as a baseline, and the last line of the table indicates the increase compared to the baseline of the best-performing model.

The gain obtained by the best-performing model is more substantial for short-term forecasts, progressively diminishing up to the long-term forecast of 48 hours. This trend is observed because, for longer forecasts, achieving the best result aligns closely with predicting an average value, and it is too difficult to accurately predict it at such a distant forecasting horizon. The table also underscores the varying suitability of models for capturing different temporal dependencies. For instance, GRU networks excel in short-term forecasts (2 and 6 hours), whereas they prove

to be unsuitable for 12-hour forecasts and longer-term forecasts. Graph WaveNet outperforms all other models for every forecast horizon. For the 48-hour forecast, the proposed ensemble model proved to be the best-performing model combining the best short-term and long-term features.

Methods	2 hour 12 instants	6 hour 36 instants	12 hour 72 instants	48 hour 288 instants
Lasso (baseline)	21.73	32.30	39.17	45.75
GRU	21.00	32.04	39.40	46.29
GRU + features	20.85	31.71	39.10	45.90
Graph WaveNet	**20.35**	**31.23**	**38.18**	45.40
Ensemble	-	-	-	**45.31**
Increments on baseline	−1.38	−1.07	−0.99	−0.44

Table 6.2: Table of all the results obtained

Chapter 7

Conclusions and Future Work

In conclusion, this thesis presents several considerations. Firstly, spatio-temporal graph neural networks (STGNN), originally designed for different tasks, have exhibited superior performance in both short-term and long-term wind power forecasting scenarios. This conclusion is drawn from a comparison with two other models, Lasso regression and a Gate Recurrent Unit (GRU) network. The incorporation of spatial data of turbines significantly contributed to the accuracy of predicting wind power generation.

It is essential to emphasize that this study utilized unprocessed data from the Baidu KDD Challenge 2022, directly captured by sensors. Preliminary data exploration and cleaning operations were necessary and played a fundamental role in managing the dataset effectively. In addressing the challenge's requirement for a 2-day wind power forecasting model (288-time instants), it becomes evident that a single model was insufficient for simultaneously capturing short-term and long-term patterns, a conclusion supported by the existing literature related to the Baidu KDD Challenge.

As a response to this problem, a decision was made to adopt an ensemble modeling approach for the final model, which combines forecasts from two Graph WaveNet models. One model is trained specifically for the initial 12 hours (short term), while the other focuses on forecasting the remaining 36 hours (long term). According to the prescriptions of the Baid KDD challenge, the resulting model achieved an error score of 45.31, securing a position within the top 10 best results on the leaderboard among a participant pool exceeding 2000 participants.

7.0.1 Future Works

The application of spatio-temporal graph neural networks in wind power forecasting introduces promising perspectives to further explore. One example is provided by the work of Rathore et al. [48] employing Graph WaveNet to predict wind speed across a graph connecting multiple cities. This study suggests the potential integration of wind power forecasts from statistical models with numerical weather prediction (NWP) models. Such an integrated approach holds the prospect of yielding significantly improved results for both short-term and long-term wind power forecasting. Following this research area, very recently a team of researchers from Google DeepMind released the source code of GraphCast [49], a graph neural network model that outperforms numerical weather prediction models in several forecasting scenarios. This model constitutes an important perspective for forecasting meteorological variables. Unlike spatio-temporal graph neural networks, GraphCast is an auto-regressive model. It uses the current state's output as input for the next step, considering the present and past states from 6 hours prior to predict the future state for the next 6 hours. This prediction is made for each node in the graph, utilizing a unique multi-mesh graph structure, as illustrated in Figure 7.1.

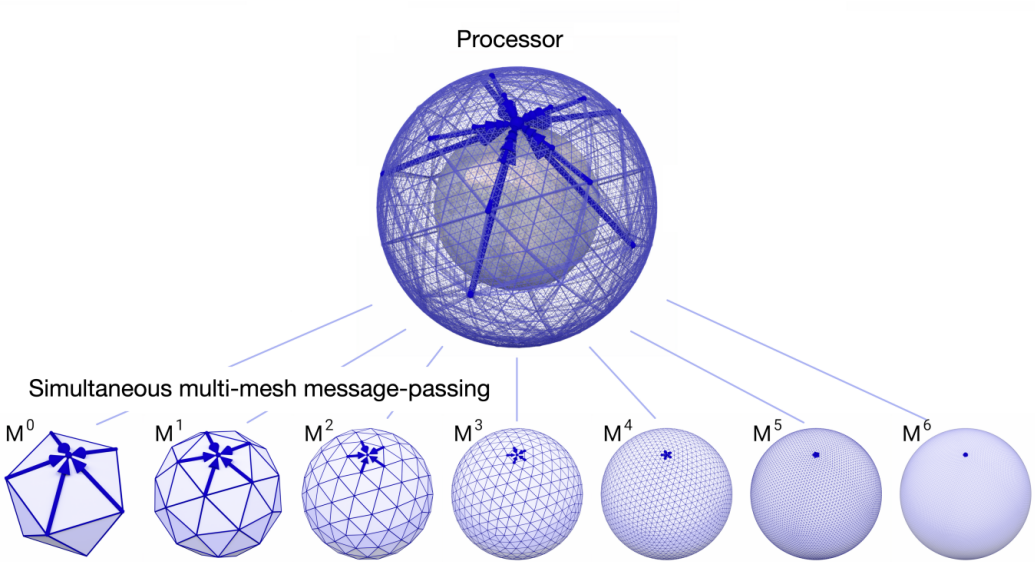


Figure 7.1: GraphCast aggregation structure as described in [49]

The distinctive feature of GraphCast is its simultaneous aggregation at multiple levels, enabling the forecast to incorporate information at different distances.

In future explorations, integrating GraphCast forecasts with the spatio-temporal graph neural network techniques presented in this thesis could lead to superior

wind power forecasting outcomes. This integration might involve using GraphCast predictions as external nodes in the wind farm graph structure. To achieve this, wind speed should be used as the target variable instead of turbine-generated power, aligning with GraphCast's focus on meteorological variables. This adjustment poses no issue, as demonstrated in Section 6.1, where wind speed serves as a proxy for wind farm power generation. Additionally, considerations must be made regarding forecast granularity, given that GraphCast makes 6-hour predictions, while wind farm forecasts typically have lower granularity. Lastly, structuring the graph to encompass the wind farm and external nodes with GraphCast predictions is essential. Figure 7.2 provides an example of a representation of what this graph structure might look like.

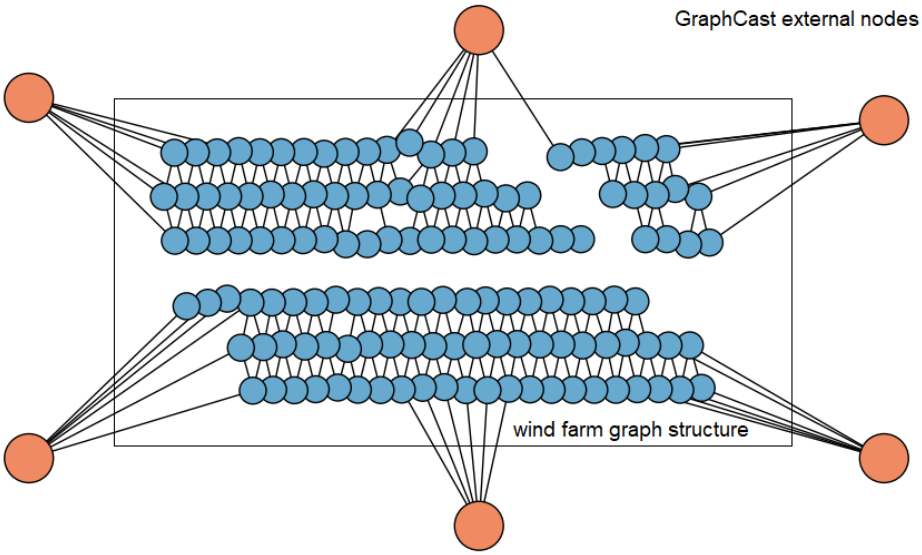


Figure 7.2: Example graph structure of a possible extension of the thesis work. Orange nodes represent external nodes with GraphCast predictions, while blue nodes depict wind farm turbines.

In this illustration, orange nodes represent external nodes with GraphCast predictions, while blue nodes depict wind farm turbines. This approach combines a global meteorological perspective from GraphCast nodes with a local perspective from turbine historical data. The wind forecast is then injected into turbine-related nodes, determining the energy produced by the wind farm. This potential model, merging GraphCast predictions with spatio-temporal graph neural networks, aligns with recent literature on meteorological variable prediction.

In conclusion, spatio-temporal graph neural networks stand as state-of-the-art

models in meteorological forecasting, particularly in the domain of wind power prediction. New models that provide both large-scale and local forecasts can drive wind power forecasting towards better performance, improving wind energy management.

Bibliography

[1] Jingbo Zhou, Xinjiang Lu, Yixiong Xiao, Jiantao Su, Junfu Lyu, Yanjun Ma, and Dejing Dou. «SDWPF: A Dataset for Spatial Dynamic Wind Power Forecasting Challenge at KDD Cup 2022». In: *arXiv preprint arXiv:2208.04360* (2022) (cit. on pp. 1, 4, 19, 28).

[2] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. «Learning phrase representations using RNN encoder-decoder for statistical machine translation». In: *arXiv preprint arXiv:1406.1078* (2014) (cit. on p. 2).

[3] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. «Graph wavenet for deep spatial-temporal graph modeling». In: *arXiv preprint arXiv:1906.00121* (2019) (cit. on pp. 2, 4, 31).

[4] Rajesh G Kavasseri and Krithika Seetharaman. «Day-ahead wind speed forecasting using f-ARIMA models». In: *Renewable Energy* 34.5 (2009), pp. 1388–1393 (cit. on p. 3).

[5] SN Singh, Abheejeet Mohapatra, et al. «Repeated wavelet transform based ARIMA model for very short-term wind speed forecasting». In: *Renewable energy* 136 (2019), pp. 758–768 (cit. on p. 3).

[6] Ilhami Colak, Seref Sagiroglu, and Mehmet Yesilbudak. «Data mining and wind power prediction: A literature review». In: *Renewable energy* 46 (2012), pp. 241–247 (cit. on p. 3).

[7] Alireza Zendehboudi, M Abdul Baseer, and R Saidur. «Application of support vector machine models for forecasting solar and wind energy resources: A review». In: *Journal of cleaner production* 199 (2018), pp. 272–285 (cit. on p. 3).

[8] Kathrine Lau Jørgensen and Hamid Reza Shaker. «Wind power forecasting using machine learning: State of the art, trends and challenges». In: *2020 IEEE 8th International Conference on Smart Energy Grid Engineering (SEGE)*. IEEE. 2020, pp. 44–50 (cit. on p. 3).

[9] Yanfei Li, Haiping Wu, and Hui Liu. «Multi-step wind speed forecasting using EWT decomposition, LSTM principal computing, RELM subordinate computing and IEWT reconstruction». In: *Energy Conversion and Management* 167 (2018), pp. 203–219 (cit. on p. 3).

[10] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. «Wavenet: A generative model for raw audio». In: *arXiv preprint arXiv:1609.03499* (2016) (cit. on pp. 3, 13, 31).

[11] Kumar Shivam, Jong-Chyuan Tzou, and Shang-Chen Wu. «Multi-step short-term wind speed prediction using a residual dilated causal convolutional network with nonlinear attention». In: *Energies* 13.7 (2020), p. 1772 (cit. on p. 3).

[12] Xiaochong Dong, Yingyun Sun, Ye Li, Xinying Wang, and Tianjiao Pu. «Spatio-temporal convolutional network based power forecasting of multiple wind farms». In: *Journal of Modern Power Systems and Clean Energy* 10.2 (2021), pp. 388–398 (cit. on p. 3).

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. «Attention is all you need». In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 3, 14, 15).

[14] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. «Temporal fusion transformers for interpretable multi-horizon time series forecasting». In: *International Journal of Forecasting* 37.4 (2021), pp. 1748–1764 (cit. on p. 3).

[15] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. «Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting». In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 3).

[16] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. «Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting». In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 22419–22430 (cit. on p. 3).

[17] Lei Wang, Yigang He, Lie Li, Xiaoyan Liu, and Yingying Zhao. «A novel approach to ultra-short-term multi-step wind power predictions based on encoder–decoder architecture in natural language processing». In: *Journal of Cleaner Production* 354 (2022), p. 131723 (cit. on p. 3).

[18] Kai Qu, Gangquan Si, Zihan Shan, XiangGuang Kong, and Xin Yang. «Short-term forecasting for multiple wind farms based on transformer model». In: *Energy Reports* 8 (2022), pp. 483–490 (cit. on p. 3).

[19] Hai-Kun Wang, Ke Song, and Yi Cheng. «A hybrid forecasting model based on CNN and informer for short-term wind power». In: *Frontiers in Energy Research* 9 (2022), p. 788320 (cit. on p. 3).

[20] Bing Yu, Haoteng Yin, and Zhanxing Zhu. «Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting». In: *arXiv preprint arXiv:1709.04875* (2017) (cit. on pp. 4, 30, 31).

[21] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. «Adaptive graph convolutional recurrent network for traffic forecasting». In: *Advances in neural information processing systems* 33 (2020), pp. 17804–17815 (cit. on p. 4).

[22] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. «Diffusion convolutional recurrent neural network: Data-driven traffic forecasting». In: *arXiv preprint arXiv:1707.01926* (2017) (cit. on p. 4).

[23] Tomasz Stańczyk and Siamak Mehrkanoon. «Deep graph convolutional networks for wind speed prediction». In: *arXiv preprint arXiv:2101.10041* (2021) (cit. on p. 4).

[24] Fei Wang, Peng Chen, Zhao Zhen, Rui Yin, Chunmei Cao, Yagang Zhang, and Neven Duić. «Dynamic spatio-temporal correlation and hierarchical directed graph structure based ultra-short-term wind farm cluster power forecasting method». In: *Applied Energy* 323 (2022), p. 119579 (cit. on p. 4).

[25] Lei Wang and Yigang He. «M2STAN: Multi-modal multi-task spatiotemporal attention network for multi-location ultra-short-term wind power multi-step predictions». In: *Applied Energy* 324 (2022), p. 119672 (cit. on p. 4).

[26] Fernando Sebastián Huerta, Manuel Angel Suarez, Daniel Velez Serrano, Alejandro Carrrasco Sánchez, and Eugenio Neira Bustamante. «Hybrid Model: Deep learning GRU neural network and K-nearest neighbors for Wind Power Forecasting». In: () (cit. on pp. 4, 47–49).

[27] Yiji Zhao, Haomin Wen, Junhong Lou, Jinji Fu, Jianbin Zheng, and Youfang Lin. «EasyST: Modeling Spatial-Temporal Correlations and Uncertainty for Dynamic Wind Power Forecasting via PaddlePaddle». In: (2022) (cit. on pp. 4, 47).

[28] Jiawei Jiang, Chengkai Han, and Jingyuan Wang. «BUAA_BIGSCity: Spatial-Temporal Graph Neural Network for Wind Power Forecasting in Baidu KDD CUP 2022». In: *arXiv preprint arXiv:2302.11159* (2023) (cit. on pp. 4, 40).

[29] Linsen Li, Qichen Sun, Dongdong Geng, Chunfei Jian, Dongen Wu, and Shiliang Pu. «Complementary Fusion of Deep Spatio-Temporal Network and Tree Model for Wind Power Forecasting (Team: HIK)». In: (2022) (cit. on pp. 4, 49).

[30] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. «LSTM: A search space odyssey». In: *IEEE transactions on neural networks and learning systems* 28.10 (2016), pp. 2222–2232 (cit. on p. 12).

[31] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. «An empirical exploration of recurrent network architectures». In: *International conference on machine learning*. PMLR. 2015, pp. 2342–2350 (cit. on p. 12).

[32] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 12).

[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «Imagenet classification with deep convolutional neural networks». In: *Advances in neural information processing systems* 25 (2012) (cit. on p. 12).

[34] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. «An empirical evaluation of generic convolutional and recurrent networks for sequence modeling». In: *arXiv preprint arXiv:1803.01271* (2018) (cit. on p. 13).

[35] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. «Neural machine translation by jointly learning to align and translate». In: *arXiv preprint arXiv:1409.0473* (2014) (cit. on p. 14).

[36] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. «How powerful are graph neural networks?» In: *arXiv preprint arXiv:1810.00826* (2018) (cit. on p. 17).

[37] Thomas N Kipf and Max Welling. «Semi-supervised classification with graph convolutional networks». In: *arXiv preprint arXiv:1609.02907* (2016) (cit. on p. 17).

[38] Simone Romano, Nguyen Xuan Vinh, James Bailey, and Karin Verspoor. «Adjusting for chance clustering comparison measures». In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 4635–4666 (cit. on p. 24).

[39] Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi. «Taming Local Effects in Graph-based Spatiotemporal Forecasting». In: *arXiv preprint arXiv:2302.04071* (2023) (cit. on p. 28).

[40] Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi. «Graph Deep Learning for Time Series Forecasting». In: *arXiv preprint arXiv:2310.15978* (2023) (cit. on pp. 28, 29).

[41] *HpcPolitoSite*. `http://www.hpc.polito.it` (cit. on p. 34).

[42] Albert Betz. *Introduction to the theory of flow machines*. Elsevier, 2014 (cit. on p. 34).

[43] Alhassan Ali Teyabeen, Fathi Rajab Akkari, and Ali Elseddig Jwaid. «Mathematical modelling of wind turbine power curve». In: *International Journal of Simulation Systems, Science & Technology* 19.5 (2018), pp. 1–13 (cit. on p. 35).

[44] Joaquin Amat Rodrigo and Javier Escobar Ortiz. *skforecast*. Version 0.10.1. Sept. 2023. DOI: 10.5281/zenodo.8382788. URL: https://skforecast.org/ (cit. on p. 37).

[45] Andrea Cini and Ivan Marisca. *Torch Spatiotemporal*. Mar. 2022. URL: https://github.com/TorchSpatiotemporal/tsl (cit. on pp. 42, 45).

[46] Hanhan Liu. «trymore: Solution to Spatial Dynamic Wind Power Forecasting for KDD Cup 2022». In: (2022) (cit. on pp. 47, 48).

[47] Aya Abdelsalam Ismail, Timothy Wood, and Héctor Corrada Bravo. «Improving long-horizon forecasts with expectation-biased LSTM networks». In: *arXiv preprint arXiv:1804.06776* (2018) (cit. on p. 48).

[48] Neetesh Rathore, Pradeep Rathore, Arghya Basak, Sri Harsha Nistala, and Venkataramana Runkana. «Multi scale graph wavenet for wind speed forecasting». In: *2021 IEEE International Conference on Big Data (Big Data)*. IEEE. 2021, pp. 4047–4053 (cit. on p. 54).

[49] Remi Lam et al. «Learning skillful medium-range global weather forecasting». In: *Science* (2023), eadi2336 (cit. on p. 54).