Wind Power Forecasting in Wind Farms using GNNs

Rodrigo Revilla Llaca

Wind Power Forecasting in Wind Farms using GNNs

by

Rodrigo Revilla Llaca

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday June 20, 2024 at 13:00.

Student number:	5737036	
Thesis committee:	Chair and Advisor: Dr. Elvin Isufi	TU Delft
	Advisor: Dr. Hadi Jamali-Rad,	TU Delft, Shell
	Member: Amirmasoud Ghiassi	TU Delft, Shell
	Member: Dr. Riccardo Taormina	TU Delft
	Member: Dr. Gosia Migut	TU Delft

Preface

This thesis marks the end of a wonderful stage in my life. Those who have known me closely for a long time know that pursuing a Master's degree was one of my main goals. I am incredibly grateful to have attended such a prestigious university and to have learned from the dedicated and talented teachers at TU Delft. While it is daunting to think that this chapter of my life is over, I am very excited to move on to the next one.

I want to express my sincere gratitude to all those who have accompanied me on this journey. First and foremost, I want to thank my fiancée, Marisú. Your unwavering support and help have been invaluable every step of the way. The passion you have for your own work is inspiring and has kept me going through the most challenging times. Working on my thesis next to you has made every minute of work extra special.

I would also like to thank Elvin for all the time and energy invested in this process. Your knowledge and passion for the subject are truly inspiring. The critical feedback and discussions we had were immensely helpful. You have taught me a lot, and your guidance and constructive feedback have made this thesis possible. Thanks also to Masoud and Hadi for guiding this project and sharing their knowledge through our recurrent discussions. Thank you, Masoud, for dedicating an hour of your schedule every week to answer my questions. Thanks as well to Dr. Riccardo Taormina and Dr. Gosia Migut for taking the time to review this document and attend the defence.

I want to thank my family: Elvira, Mariana, and Francisco. Thank you for believing in me and supporting my crazy dreams. I would not be where I am without you. Your love for your own work motivates me in all my pursuits. Finally, I want to thank my friends for their moral support and encouragement during the last few months. Being far away from home is difficult, but the friends I have in Delft and back home have made it a bit easier.

I hope that whoever reads this thesis finds it interesting and inspiring. It wraps up a very important and enjoyable stage in my life.

Rodrigo Revilla Llaca
Delft, June 2024

Summary

Accurate forecasts are essential for integrating wind energy into the power grid. With wind energy's growing role in the renewable mix, precise short-term generation forecasts are increasingly vital. Turbine-level forecasts are critical for optimal wind farm operation, control, and planning. However, the wind's unpredictability and the complex interactions between turbines make forecasting challenging. Existing methods either depend on resource-intensive physical models or overlook turbine interactions, resulting in isolated, overly simplistic models.

This master's thesis explores the use of Graph Neural Networks (GNNs) in conjunction with Recurrent Neural Networks (RNNs) for wind power generation forecasting in wind farms. The result is an integrated approach that can learn the interactions between turbines through GNNs to handle the temporal dynamics of the data through RNNs.

The research begins with an in-depth overview of wind turbines and their arrangement in wind farms, highlighting the importance and challenges of Wind Power Forecasting (WPF). It establishes that the spatial configuration of turbines complicates power generation estimation, especially over different time horizons. Graphs are introduced as an effective way to represent the interconnections between turbines. Various methods for constructing these graphs are reviewed, focusing on predefined heuristics and their limitations in adapting to changing wind farm characteristics.

The core of this thesis is the AG-LSTM Network, a model that integrates RNNs and GNNs for short-term WPF for turbines within a wind farm. It utilizes historical power generation data, a variety of future and past covariates, and static turbine-specific features. The model is based on an encoder-decoder architecture with an Adaptive Graph Long-Short Term Memory Cell (AG-LSTM), which generates a dynamic adjacency matrix representing the farm's state at each moment. This matrix combines the input features and hidden states across turbines, resulting in an embedding that is transformed into the power forecast.

Through comprehensive experiments and analyses, the AG-LSTM Network proves to be more accurate than state-of-the-art methods on two real-life datasets when future covariates are unavailable. Further experiments demonstrate the quality of the forecast and break down the contribution of the different choices regarding its architecture. The results underscore the potential of hybrid neural network architectures in enhancing WPF, providing a valuable tool for the renewable energy sector.

In conclusion, this research contributes to the renewable energy field by proposing a novel hybrid neural network model that effectively addresses the complexities of short-term WPF. Future work could explore further exploitation of future covariates and the robustness of the model against incomplete input data.

Contents

List of Figures

List of Tables

1 Introduction

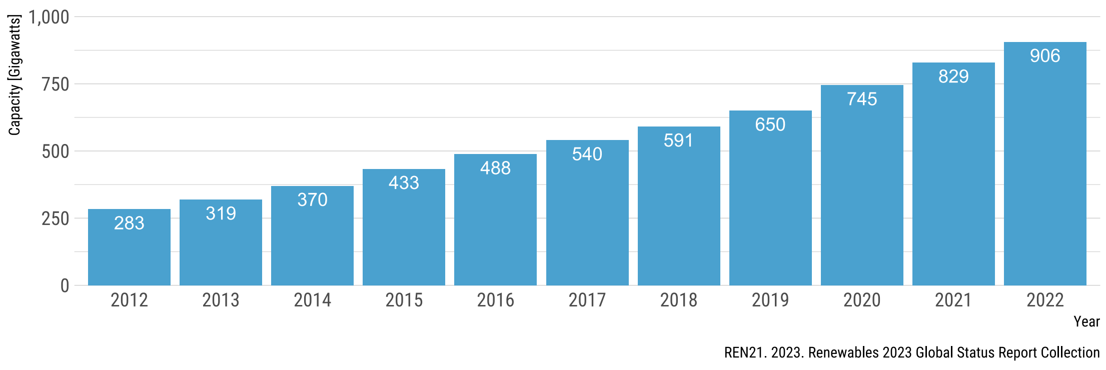


Figure 1.1: Worldwide installed wind capacity per year from 2012 to 2022 [18].

The European Union has committed to becoming carbon neutral by 2050, recognizing that renewable energy sources are crucial for reaching this target. Renewable energy sources, including solar, wind, hydro, and biomass, offer a sustainable and environmentally friendly alternative to fossil fuels, reducing greenhouse gas emissions and lessening the dependency on imported energy.

Wind power has emerged as a vital element in the renewable energy landscape and is set to gain even greater significance. Over the past decade, the installed capacity of wind power has consistently increased, as showcased in Figure 1.1. The International Energy Agency forecasts that this growth will accelerate, with projections suggesting that within the next decade, wind power will rise to become the second largest source in terms of installed capacity, as depicted in Figure 1.2. To seamlessly integrate wind energy into the power grid, players in this sector must be able to provide accurate forecasts of wind power production. Precise predictions help ensure a balanced and stable power supply, enabling better planning and optimization of the grid and enhancing the overall reliability and efficiency of the energy system.

Wind turbines are typically organized into wind farms to mitigate costs and efficiently use available space. Placing turbines sufficiently close together gives rise to a phenomenon known as the Wake Effect. The wake that results from the motion of a turbine affects the characteristics of its downstream wind flow, which in turn impacts the turbines within a range. The effect on the downstream wind current is very complex, as it depends on many conditions which are particular to the state of the farm and the environmental conditions.

The Wake Effect of a turbine can affect multiple others, which, in turn, have their corresponding Wake Effects. This increasing distortion in the wind patterns makes the resulting conditions very hard to approximate. Over the years, multiple methods have emerged to model these interactions and produce power forecasts. Still, these tend to be too complex and resource-intensive, as they rely on solving differential equations representing the flows. Nonetheless, research has demonstrated that graph-based approaches are effective in modelling interactions within wind systems.

Graphs are an effective tool for representing a system by showing the relationships between its components. They have proven useful in modelling all kinds of systems including complex ones with highly dynamic characteristics, such as road traffic networks.

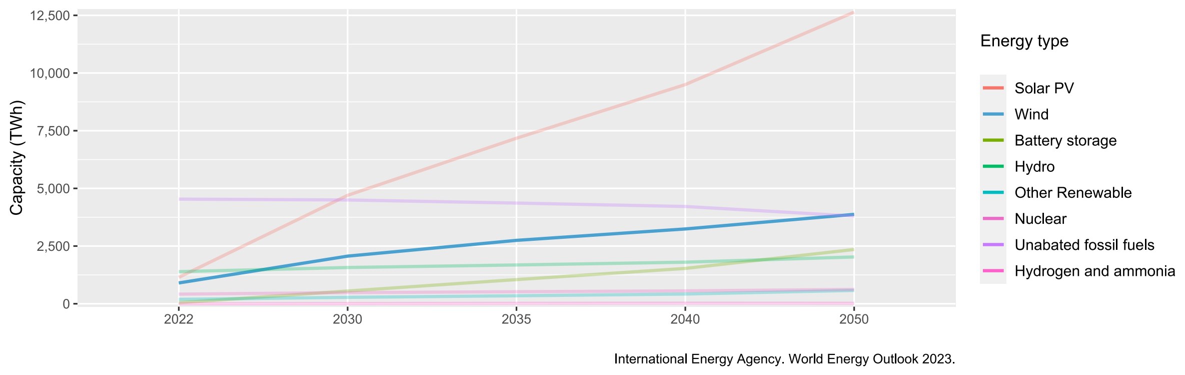


Figure 1.2: Projected Power Global Capacity by Source, 2022-2050. Adapted from [39].

In these networks, vehicles travel between roads and intersections, and small disturbances can quickly propagate through the system. The complex interactions between turbines in a wind farm can naturally be translated into a graph, with turbines represented as nodes and their physical interactions as edges.

Furthermore, Graph Neural Networks (GNNs) can leverage this farm representation by learning to combine and aggregate the information. The resulting embeddings can then be used for downstream tasks, such as Wind Power Forecasting (WPF). GNNs have proven successful for forecasting in other domains, including road traffic and social networks, demonstrating their ability to exploit the information in complex relational data. Our goal in this thesis is to expand the list by conducting additional applications of GNNs to WPF. We specifically aim to investigate the following:

Research Question

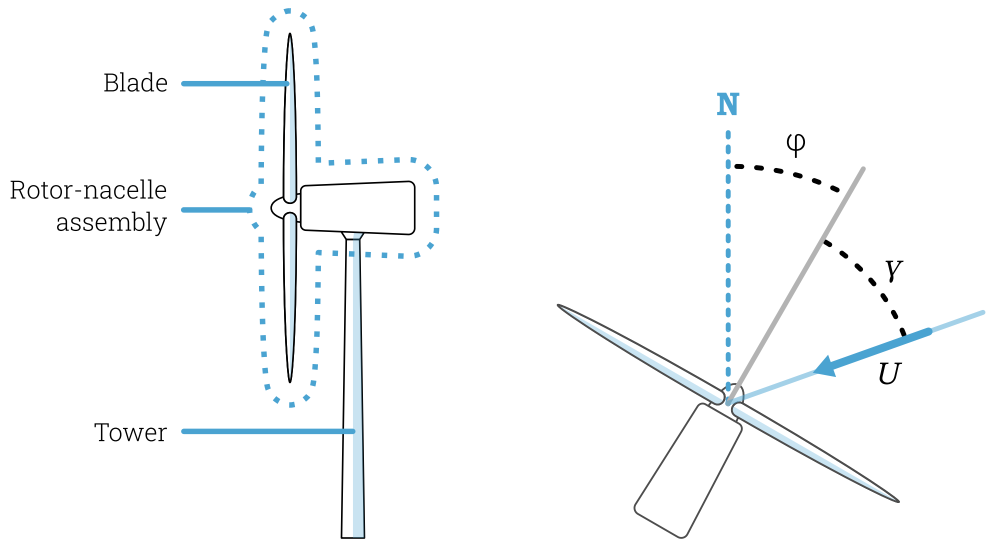
How can GNNs efficiently utilize graph structures to capture turbine interactions in wind farms to generate accurate short-term wind power forecasts?

To address the research question, we propose the Adaptive-Graph Long Short Term Memory (AG-LSTM) Network, a model that utilizes historical multivariate time-series, future covariates and static features to produce multi-step power forecasts for all the turbines in a wind farm. The model builds upon existing forecast models, integrating two Recurrent Neural Networks (RNNs) in an encoder-decoder configuration. Notably, the encoder consists of a Long-Short Term Memory cell whose states are updated through Graph Neural Networks. The model learns to dynamically adapt the GNN's adjacency matrix, reflecting the immediate interactions within the wind farm. Our model is tested and compared to other methods, showing promise for the short-term WPF task.

This thesis is organized as follows: Chapter 2 provides the building blocks for wind farms and Graph (Temporal) Neural Networks. We introduce the problem, the notation, and delve into GNNs and RNNs. In Chapter 3, we classify the existing models for WPF. We explore how these methods currently tackle the problem and identify their strengths and caveats. Chapter 4 presents our proposed model, detailing its components and their integration. Chapter 5 presents the evaluation of the model using the SDWPF and Penmanshiel datasets, which contain information from two operational wind farms. In the same chapter, we test our model and compare its performance to other models discussed in the literature. We finally assess the impact of its components and parameters towards the obtained results. Finally, Chapter 6 concludes this work by providing an answer to the research question, the general findings of the work, and some proposed future research directions.

2 Background

This chapter focuses on the essential concepts related to wind turbines, wind farms, and Graph (Temporal) Neural Networks. §2.1 describes the characteristics of wind farms. §2.2 introduces the graph terminology, which is used in §2.3 to introduce the graph representation of wind farms. §2.4 and §2.5 elaborate on GNNs and RNNs, respectively. The AGCRN, which integrates both networks by combining graph convolutions that merge node signals and cell states within the cell's update equations, is described in §2.6.



(a) Side view **(b)** Top view

Figure 2.1: a) depicts the main components of a HAWT. b) illustrates the angles formed between the orientation of the rotor-nacelle assembly φ and the wind direction U. The yaw angle γ is the difference between them.

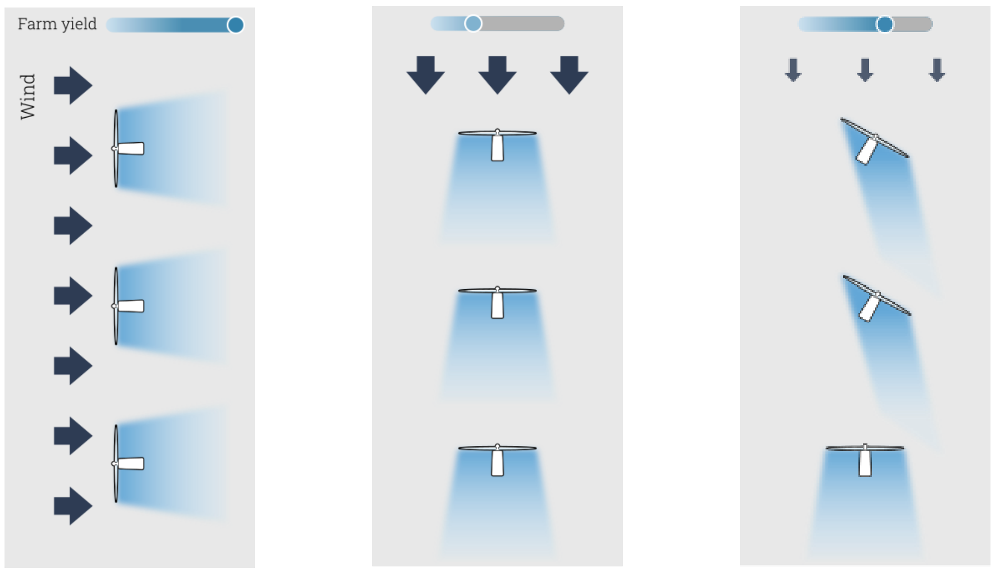
2.1 | Wind Farms

Wind turbines convert the wind's kinetic energy into mechanical energy, which is then used to generate electricity. While these devices have various configurations, horizontal-axis wind turbines (HAWTs) are the most widely used [85]. As depicted in Figure 2.1a, a HAWT consists of three large blades attached to the rotor-nacelle assembly, which is mounted on top of the tower. As the wind hits the turbine's blades, they rotate and drive the generator responsible for producing electricity.

Turbine manufacturers provide power curves, which are used to assess the performance and energy production potential of wind turbines based on varying wind speeds. However, the power production of a HAWT is not solely dependent on wind speed. Other factors that affect the power production include the orientation of the rotor-nacelle assembly with respect to the direction of the wind (yaw angle), the diameter of the rotor and the orientation of its blades (pitch angles) [64]. Figure 2.1b depicts the yaw angle.

Most HAWTs are equipped with sensors that constantly monitor their operational conditions. These instruments include anemometers, wind vanes, thermometers, and rotary encoders. A Supervisory Control and Data Acquisition system (SCADA) collects sensor data from the individual turbines and meteorological stations, providing operators with essential monitoring and control parameters [5]. SCADA measurements are recorded at ten-minute intervals [5], allowing operators to react to changes in wind speed and direction by adjusting the pitch angle and the orientation of the rotor-nacelle assembly to optimize turbine performance.

Turbines are grouped in wind farms, ranging from two to hundreds of wind turbines, with the latter covering hundreds of square kilometres [36]. The power yield of a wind farm is the sum of the production of the turbines that compose it. The performance of a wind farm is closely related to the topological and environmental conditions of its



(a) Ideal scenario **(b)** Greedy scenario **(c)** Optimal steering

Figure 2.2: Illustration of the wake effect. a) Turbines are aligned with the wind without wake interactions. b) Turbines are aligned with the wind. Power for the front turbine is maximized, but its wake considerably affects the other turbines. c) By steering the upstream turbines, the farm yield is prioritized over the individual power production.

geographical location, the physical characteristics of the individual wind turbines, and the relative distances and angles between them.

Wind farms provide numerous financial and operational benefits, though this setup can adversely affect the performance of individual turbines within the farm. When wind strikes a turbine, it creates a wake where wind speed decreases, and turbulence increases for a distance up to 20 times the rotor diameter [85]. Beyond this distance, the wind conditions gradually revert to those of the free-stream wind. The Wake Effect refers to the influence that this wake has on turbines located downstream. The altered wind conditions reduce the power production of individual downstream turbines by up to 60%, which can impact the production of an entire wind farm by as much as 54% [46]. This phenomenon is illustrated in Figure 2.2.

Under ideal conditions, the turbines in a wind farm would be placed sufficiently far apart, minimizing Wake Effects and maximizing individual power generation. In practice, however, this is not feasible due to technical limitations and costs [35]. Although wind farm layouts have transitioned from grid formations to optimized layouts due to complex optimization studies [26, 32], modern wind farms are still subject to the Wake Effect between their turbines.

The interaction between turbines has significant effects on the operation of wind farms. When maximizing the farm's yield, the turbines cannot be considered independently. Instead, the system must be considered as a whole [81], and the power production of some turbines must be limited by changing their orientation to mismatch the wind direction, reducing the wake effect on downstream turbines (Figure 2.2) [48]. Correctly estimating the impact of the individual yaw angles on the power production of the other turbines and quantifying the total yield of the wind farm results in a non-trivial problem that is actively researched in the energy field [6, 32, 49].

Accurate forecasting of wind power generation is essential for wind farm operators, enabling them to make well-informed decisions regarding their electricity supply to the grid [5]. WPF algorithms rely on the most recent SCADA data, where the number of previous time-steps considered for the calculation is called the *observation window*. The *forecast window* refers to the number of future time-steps for which values are estimated. As in any forecasting task, the estimation accuracy diminishes with the forecast horizon [79]. The resolution for the forecast horizon usually aligns with that of the observation window, typically set at 10 minutes [5]. Single-step forecasts estimate the value for one future step, whereas multi-step forecasts result in multiple of these estimates.

WPF forecasts are categorized into four groups, depending on the forecast window; i) ultra-short-term (up to 1 hour ahead), ii) short-term (within one day), iii) mid-term (one day to one week), and iv) long-term (spanning multiple weeks) [25]. Ultra-short-term and short-term WPF aids in real-time grid operations, grid stability, operational security, and regulatory actions. Mid-term forecasts are utilized for planning purposes such as unit commitment and reserve allocation. Long-term strategic decisions, including feasibility studies, maintenance programming, and resource allocations, rely on long-term forecasting [25, 16].

As detailed in Chapter 1, wind power plays a crucial role in the energy mix, and will continue to do so in the upcoming decades. This thesis focuses on the short-term horizon, due to its critical role in enabling the seamless and reliable integration of wind power into the power grid.

2.2 | Graphs

Graphs are a representation of the relationships that exist between different entities. A graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, ..., N\}$ represents a set of entities (or nodes) and $\mathcal{E}$ represents a set of edges between entities. The number of nodes and edges in a graph is denoted by $|\mathcal{V}|$ and $|\mathcal{E}|$, respectively. An edge connecting nodes i and j is denoted by $(i, j) \in \mathcal{E}$ and the nodes are considered *neighbours*. Graphs are either undirected or directed. In an undirected graph, if edge $(i, j) \in \mathcal{E}$, then an equivalent edge $(j, i) \in \mathcal{E}$. In directed graphs, edges have a direction. For an edge $(i, j) \in \mathcal{E}$, j is dubbed the out-neighbor of i, and i is the in-neighbor of j. The edges between nodes in a graph are usually represented in matrix form.

The adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ describes the connectivity between the nodes in a graph. In the unweighted scenario, the entries of $\mathbf{A}$ are binary; if $(i, j) \in \mathcal{E}$, $\mathrm{A}_{ij} = 1$. Otherwise, $\mathrm{A}_{ij} = 0$. See Figure 2.3 for an example. For weighted graphs (see Figure 2.4), the entries of $\mathbf{A}$ are scalars: A_{ij} corresponds to the strength of edge $(i, j) \in \mathcal{E}$. Examples of adjacency matrices for different types of graphs are presented in Figures 2.3 and 2.4,

The Laplacian Matrix, defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, offers an alternative way to represent a graph's structure. While $\mathbf{A}$ can be used for both directed and undirected graphs, $\mathbf{L}$ is used for undirected graphs. In general, a Graph Shift Operator (GSO) is any matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ that captures the structure of a graph. GSOs are usually sparse, meaning that their entries $s_{ij} \neq 0$ only if there is an edge $(i, j) \in \mathcal{E}$ and $i \neq j$. Examples of GSOs include $\mathbf{L}$, $\mathbf{A}$, and their normalized versions.

Graph signals represent quantities related to the elements of a graph. The signals for

$$\mathbf{A} = \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F, \end{array}$$

$$\mathbf{A} = \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 1 & 3 & 2 & 0 & 0 \\ 1 & 0 & 0 & 4 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 2 \\ 2 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F, \end{array}$$

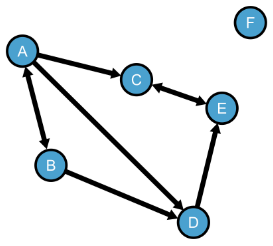


Figure 2.3: Example of a directed unweighted graph.

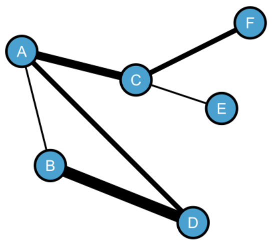


Figure 2.4: Example of an undirected weighted graph.

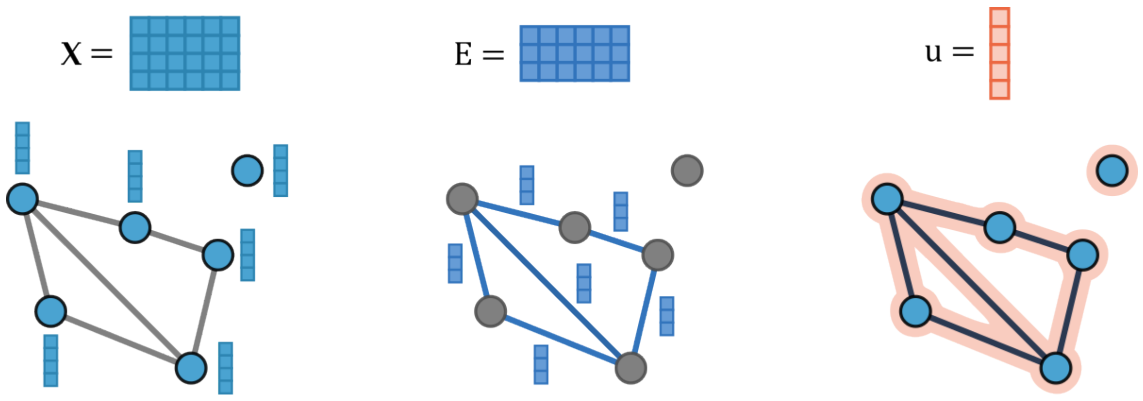


Figure 2.5: A graph can have three types of signals associated: node features (left), edge features (middle) and graph features (right).

all nodes are stored in the feature vector $\mathbf{x} \in \mathbb{R}^N$ [41]. In the multivariate scenario, the F graph signals are stacked column-wise into matrix $\mathbf{X} \in \mathbb{R}^{N\times F}$. Similarly, the signals for all edges are stored in vector $\mathbf{x}^e \in \mathbb{R}^{|\mathcal{E}|}$, and matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}|\times E}$ collects the edge vectors in the multivariate case. A graph can also have a set of global features stored in a vector $\mathbf{g} \in \mathbb{R}^G$ [20]. The different types of graph signals are illustrated in Figure 2.5.

2.3 | Wind Farm Graph Structure

Wind farms can be naturally represented as graphs by modelling each turbine as a node. Defining the edges poses challenges, as the relationship they reflect is not defined beforehand. These edges may signify physical proximity, electrical connectivity, or the influence of the wake effect between turbines. The directional or non-directional nature of the edges captures the dynamics within the wind farm. Additionally, introducing edge weights allows for a more nuanced representation, considering factors such as the strength of inter-turbine interactions or the impact of one turbine's output on another. Some common heuristics for defining edge weights, as illustrated in Figure 2.6, include:

- Geographical distance. It builds on the idea that the closer two turbines are, the more related they are. A function (e.g. Gaussian kernel) is used to map shorter distances to larger weights:

$$\mathrm{A}_{ij} = \begin{cases} \exp\left(-\frac{\delta_{ij}^2}{\sigma^2}\right), & \text{if } i \neq j \text{ and } \delta_{ij} \leq \delta_{max} \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

  where σ is the kernel bandwidth parameter (often the standard deviation), δ_{ij} is the Euclidean distance between turbine i and turbine j, and δ_{max} is a distance threshold used to limit connectivity.
- Correlation. Considers the Pearson correlation coefficient [30], which measures the linear correlation for time-series data between two turbines:

$$\rho(i,j) = \frac{\sum_{k=1}^{n}(x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j)}{\sqrt{\sum_{k=1}^{n}(x_{ik} - \bar{x}_i)^2}\sqrt{\sum_{k=1}^{n}(x_{jk} - \bar{x}_j)^2}} \tag{2.2}$$

  where x_{ik} corresponds to the time-series value for turbine i at time k, and $\bar{x}_i$ to the sample mean for turbine i. The edge weights are set as follows:

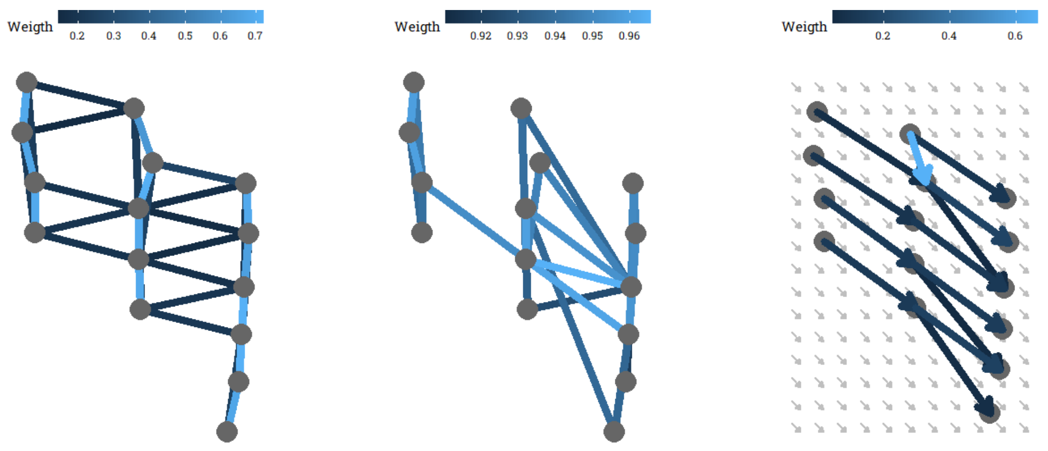
$$\mathrm{A}_{ij} = \begin{cases} \rho(i,j) & \text{if if } i \neq j \text{ and } \rho(i,j) > \rho_{max} \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

- Wake interactions [68]. The direction of the wind and the relative angle formed between the turbines are utilized to generate directed edges. The resulting weights are inversely related to distance.

$$\mathrm{A}_{ij} = \begin{cases} \exp\left(-\frac{\delta_{ij}^2}{\sigma^2}\right), & \text{if } i \neq j,\ \delta_{ij} \leq \delta_{max},\ \phi_{ij} \leq \phi_{max} \text{ and } i \text{ is upstream relative to } j \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

  where ϕ_{ij} represents the contained angle between turbines i and j, and ϕ_{max} is the influence cut-off angle. Unlike the other two, this scenario captures the dynamic nature of a wind farm. The edge weights evolve with the wind; whether a turbine is upstream relative to its neighbours depends on the wind direction at a particular instant.

Global graph features of wind farms include wind characteristics, such as the mean or maximum speed, direction, and turbulence intensity [12, 20, 68]. Node features include the measurements registered by the sensors located in each turbine: temperature, wind speed, pitch angle, nacelle orientation and output power. Although uncommon in graphs representing wind farms, edge features can convey relative angles and distances between connected nodes [20]. The features of any of these elements may pertain to a specific time, encompass a series of historical values, or summarize a time-series using statistical measures.



(a) Distance. **(b) Correlation.** **(c) Wake interaction.**

Figure 2.6: Examples of heuristics used to define edges and their weights for a wind farm graph: a) Edges exist between turbines less than 1.1 km apart. Distances are transformed into weights through a Gaussian kernel. b) Considers the Pearson correlation coefficient for historical power generation. For each turbine, the two edges with larger weights are kept. c) Considers a wind stream blowing to the SE (grey arrows). Directed edges link an upstream turbine to its downstream neighbour if they are less than 1 km apart and their relative angle aligns with the wind ($\pm 30°$). Weights are computed as in a). Node positions reflect real-life turbine coordinates.

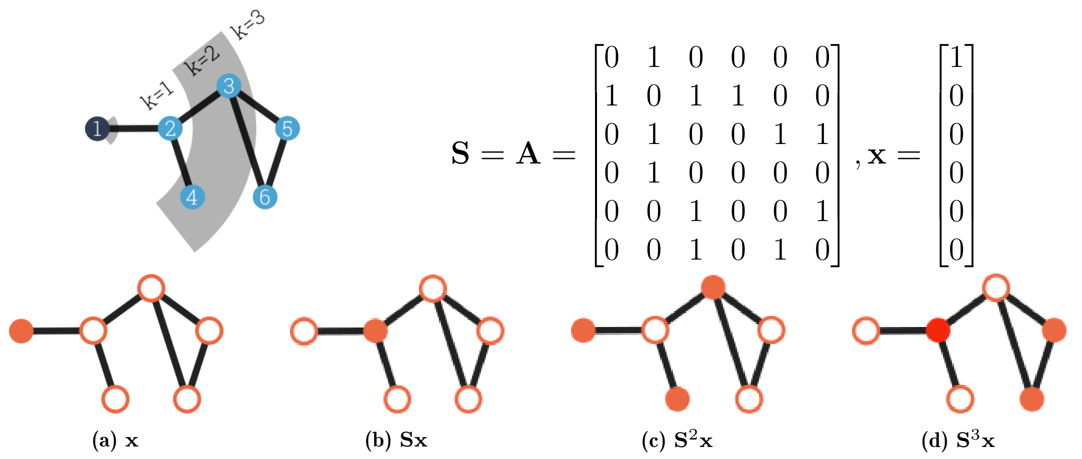
2.4 | Graph Neural Networks

Graph Neural Networks (GNNs) can effectively capture the complex relationships and structures inherent in graph data. They utilize various techniques to propagate and transform information across nodes and edges, enabling them to learn meaningful patterns and representations, which are used to solve tasks related to the graph (e.g., graph classification) or to its components (e.g., node regression). The core mechanisms used in GNNs are graph convolutions, which extend the concept of convolution from traditional signal processing to the graph domain.

2.4.1. Graph Convolutional Filters

A graph convolutional filter is a linear combination of shifted signals across a graph structure. By extending linear signal processing principles to graph data, GNNs are able to shift information across nodes [40]. The linear transformation $S(\mathbf{x}) = \mathbf{Sx}$ allows node signals $\mathbf{x}$ to propagate across a graph, following the topology encoded in a GSO $\mathbf{S}$. When the GSO corresponds to the adjacency matrix $\mathbf{A}$ of an unweighted matrix, this operation replaces the signal for a node with the sum of the signals from its neighbouring nodes (one-step propagation), as illustrated in Figure 2.7. Recursively applying this operation allows a signal to propagate across the graph. Applying the operator k times enables a signal to shift k-hops away from its original node [41]. The propagation is seen in Figure 2.7; the signal that belongs to node 1 reaches its first, second and third order neighbours when multiplied by $\mathbf{S}$, $\mathbf{S}^2$ and $\mathbf{S}^3$ respectively.

Through the graph convolutional filter, a new signal is formed that aggregates information across the graph. The different shifts of the signal are filtered through the GSO



$$\mathbf{S} = \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Figure 2.7: A graph is illustrated in the top left, where node one is highlighted and its k-th order neighbors are identified. The scalar values within signal $\mathbf{x}$ shift from a node to its neighbours when multiplied by the GSO $\mathbf{S}$. The exponential of the $\mathbf{S}$ corresponds to the order of neighbours the signal reaches.

and aggregated with different amounts of contribution:

$$\mathbf{H}(\mathbf{S})\mathbf{x} = \sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{x} \tag{2.5}$$

$\mathbf{H}(\mathbf{S})\mathbf{x}$ is the resulting graph signal, $\mathbf{x}$ are the (un)shifted signals and $h_0, ..., h_k$ determine the contribution of each shift (up to K-th order) of the signal to the final output. Learning these parameters translates into learning an embedding of the original signal $\mathbf{x}$ leveraging the graph's topology $\mathbf{S}$. Although helpful in solving tasks such as signal denoising, graph convolutional filters have limited expressive power. After all, they are based on linear functions and thus cannot model complex non-linear relationships even if the number of parameters increases.

2.4.2. Building GNNs

The graph perceptron is the building block of any GNN. It adapts the Multilayer Perceptron (MLP), traditionally used upon Euclidean data to graph signals [41]. A response is obtained when a non-linearity applied to the output of a graph convolutional filter:

$$\mathbf{y}(\mathbf{x}) = \sigma \left[\sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{x} \right] \tag{2.6}$$

σ can be any point-wise non-linear function, such as the rectified linear unit (ReLU). GNNs are formed by layering L graph perceptrons on top of each other. In this approach, non-linear operators are applied to the linear combination of signals in a recursive manner. For each layer l in the GNN, a signal propagation occurs:

$$\mathbf{x}_l = \sigma(\mathbf{z}_l) = \sigma \left[\sum_{k=0}^{K} h_{lk} \mathbf{S}^k \mathbf{x}_{l-1,l-2,...,L} \right] \tag{2.7}$$

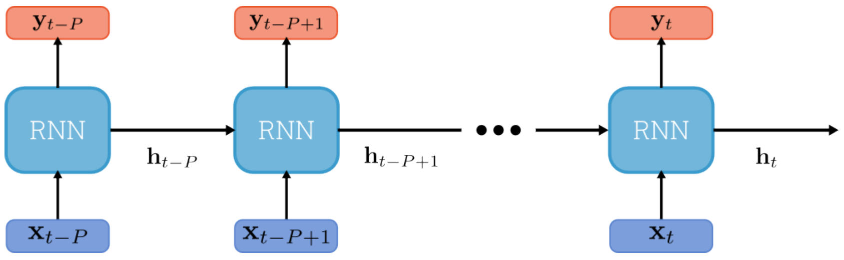


Figure 2.8: Unrolled architecture of an RNN. A cell is used for each element in the input sequence, resulting in an output and an update in the hidden state. Adapted from [66].

Thus, the output of a GNN is a function of the input signals $\mathbf{x}$, the graph structure $\mathbf{S}$ and a set of L trainable parameters $\mathcal{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_L\}$:

$$\hat{\mathbf{y}} = \mathbf{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H}) \tag{2.8}$$

2.4.3. GNN Training

GNNs are trained to learn the optimal parameter $\mathcal{H}^\star = \{\boldsymbol{h}_1^\star, \boldsymbol{h}_2^\star, ..., \boldsymbol{h}_L^\star\}$ given a training set $\mathcal{T}$ and a loss function $\mathcal{L}(\cdot)$ that quantifies the difference between the output of the model $\hat{\mathbf{y}}$ and the ground truth $\mathbf{y}$ [40]:

$$\mathcal{H}^\star = \arg\min_{\mathcal{H}} \sum_{\mathbf{x}\in\mathcal{T}} \mathcal{L}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = \arg\min_{\mathcal{H}} \sum_{\mathbf{x}\in\mathcal{T}} \mathcal{L}\left(\mathbf{y}, \mathbf{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})\right) \tag{2.9}$$

Model training is achieved by iteratively updating the parameters through backpropagation, which works on smooth convex error functions, as they are differentiable. The loss function is selected depending on the task the network aims to solve. Two standard options for regression tasks are L_1 and L_2 [19]. L_1, or Mean Absolute Error (MAE) computes the average absolute differences between $\hat{\mathbf{y}}$ and $\mathbf{y}$:

$$L_1 = \frac{1}{R} \sum_{i\in\mathcal{T}} \|\hat{\mathbf{y}}_\mathbf{i} - \mathbf{y}_\mathbf{i}\|_1 \tag{2.10}$$

L_2, or Mean Squared Error (MSE) is the sum of the square of the differences between $\hat{\mathbf{y}}$ and $\mathbf{y}$:

$$L_2 = \frac{1}{R} \sum_{i\in\mathcal{T}} \|\hat{\mathbf{y}}_\mathbf{i} - \mathbf{y}_\mathbf{i}\|_2^2 \tag{2.11}$$

Over the years, various GNN architectures have been developed, with different purposes and levels of complexity. All of them build upon the foundational concepts discussed earlier. Their goal is to leverage the graph structure to combine information across it, learning to produce meaningful representations of the data.

2.5 | Recurrent Neural Networks

RNNs are a class of Neural Networks capable of handling sequential data, such as time-series. These networks are applied recurrently to the input sequence and implement a memory cell to preserve a state across time-steps [93]. The cell's state, known as the

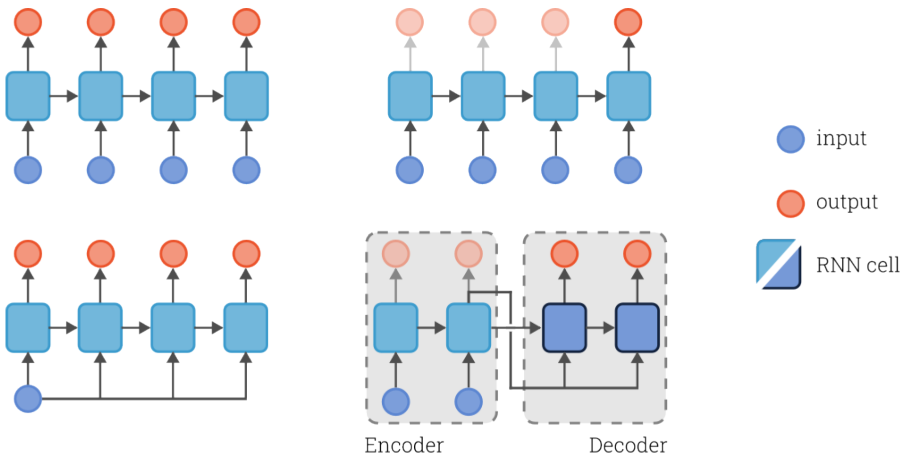


Figure 2.9: Diagram of four common RNN architectures, adapted from [54, 61]. From left to right and top to bottom: *sequence-to-sequence*, *sequence-to-vector*, *vector-to-sequence* and *encoder-decoder*.

hidden state, is denoted by $\mathbf{h}_t \in \mathbb{R}^H$, where H can take any size. At each time-step, t, the recurrent cell utilizes the input vector $\mathbf{x}_t \in \mathbb{R}^N$ and its previous hidden state $\mathbf{h}_{t-1}$ to update $\mathbf{h}_t$ and yield the output $\hat{\mathbf{y}}_t \in \mathbb{R}^O$, as illustrated in Figure 2.8. The hidden state and output at time t are computed as:

$$\mathbf{h}_t = \sigma(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_h) \tag{2.12a}$$
$$\hat{\mathbf{y}}_t = \sigma(\mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y) \tag{2.12b}$$

where $\sigma(\cdot)$ is an activation function (typically tanh); $\mathbf{W}_{hh} \in \mathbb{R}^{H\times H}$, $\mathbf{W}_{xh} \in \mathbb{R}^{H\times N}$ and $\mathbf{W}_{hy} \in \mathbb{R}^{O\times H}$ are weight matrices; and $\mathbf{b}_h \in \mathbb{R}^H$, and $\mathbf{b}_y \in \mathbb{R}^O$ are bias vectors. These parameters are shared across the different cells of the network. The number (and size) of these parameters is independent of the length of the input sequence but rather depends on the dimension of the input N [72].

2.5.1. RNN Cell Configuration

RNN cells can be configured differently, depending on the input at hand and the desired output. Four common configurations [54, 61] are depicted in Figure 2.9 and detailed below:

1. In *sequence-to-sequence*, a cell is unrolled for each one of the elements of the input sequence, and the output consists of all the individual cell outputs.
2. The *sequence-to-vector* setting implements the same architecture, although all the cell outputs are ignored except for the last one.
3. The *vector-to-sequence* configuration takes as an input a single vector, which is repeated and fed to each cell. The updates in the hidden state result in updated outputs, which are compiled as the model's output.
4. The *encoder-decoder* approach combines two independent RNNs [61]. The first one, the Encoder, captures the input following the *sequence-to-vector* approach. The resulting vector is then fed to the Decoder, with a *vector-to-sequence* configuration.

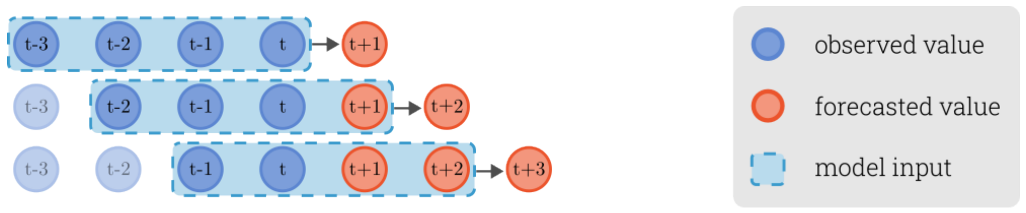


Figure 2.10: Example of the recursive strategy to obtain a multi-step output from single-step forecasts. In this example, $O = 4$ and $H = 3$. Image adapted from [52].

When the task consists of producing a multi-step forecast, the RNN configurations that output a sequence can be directly used [73]:

$$[\hat{\mathbf{y}}_{t+1}, \hat{\mathbf{y}}_{t+2}, ..., \hat{\mathbf{y}}_{t+H}] = \mathcal{F}_S(\mathbf{x}_t, \mathbf{x}_{t-1}, ..., \mathbf{x}_{t-O+1}; \mathcal{H}) \tag{2.13}$$

where $\hat{\mathbf{y}}_\mathbf{t}$ corresponds to the forecast at time t, $\mathcal{F}_S(\cdot)$ represents a RNN with a sequence output parameterized by $\mathcal{H}$, O is the size of the observation window and H corresponds to the size of the forecast window. However, vector outputting networks have to be executed recurrently, as each iteration produces a value that is used as part of the input of the next iteration [73]:

$$\begin{aligned}
\hat{\mathbf{y}}_{t+1} &= \mathcal{F}_V(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, ..., \mathbf{x}_{t-O+1}; \mathcal{H}) \\
\hat{\mathbf{y}}_{t+2} &= \mathcal{F}_V(\hat{\mathbf{y}}_{t+1}, \mathbf{x}_t, \mathbf{x}_{t-1}, ..., \mathbf{x}_{t-O+2}; \mathcal{H}) \\
\hat{\mathbf{y}}_{t+3} &= \mathcal{F}_V(\hat{\mathbf{y}}_{t+2}, \hat{\mathbf{y}}_{t+1}, \mathbf{x}_t, ..., \mathbf{x}_{t-O+3}; \mathcal{H}) \\
&\cdots \\
\hat{\mathbf{y}}_{t+H} &= \mathcal{F}_V(\hat{\mathbf{y}}_{t+H-1}, \hat{\mathbf{y}}_{t+H-2}, \hat{\mathbf{y}}_{t+H-3}, ..., \hat{\mathbf{y}}_{t+H-O}; \mathcal{H})
\end{aligned} \tag{2.14}$$

where $\mathcal{F}_V(\cdot)$ represents a RNN with a vector output parameterized by $\mathcal{H}$. An example of the recursive strategy is provided in Figure 2.10.

RNNs are trained using a special type of backpropagation called Backpropagation-rough Time (BPTT) [74]. The RNN is first unrolled, and the input undergoes a forward pass to obtain an output. The loss function $\mathcal{L}(\cdot)$ is defined to ignore part of the output and only considers the relevant portion according to the selected configuration. The gradients are propagated backwards and updated accordingly through BPTT. As these computations must be executed for each of the cells in the network, some problems may arise given long input (or output) sequences. Firstly, the network might run into the unstable gradients problem, resulting in the gradients in the initial layers becoming too small to have the weights updated or too large, resulting in a diverging solution [74]. The network may also be unable to remember information regarding the first inputs as it unrolls across the sequence [9].

2.5.2. LSTMs

LSTMs are an improved type of RNN capable of remembering information across long sequences while also attending to details encoded in the short-term. They incorporate the hidden state $\mathbf{h}_t$ found in RNNs in addition to a long-term state $\mathbf{c}_t$, and three gates ($\mathbf{f}_t$, $\mathbf{i}_t$ and $\mathbf{o}_t$) responsible for controlling changes to the cell states [93]. A schematic diagram of the mechanisms inside the LSTM cell is presented in Figure 2.11. The corresponding equations are:

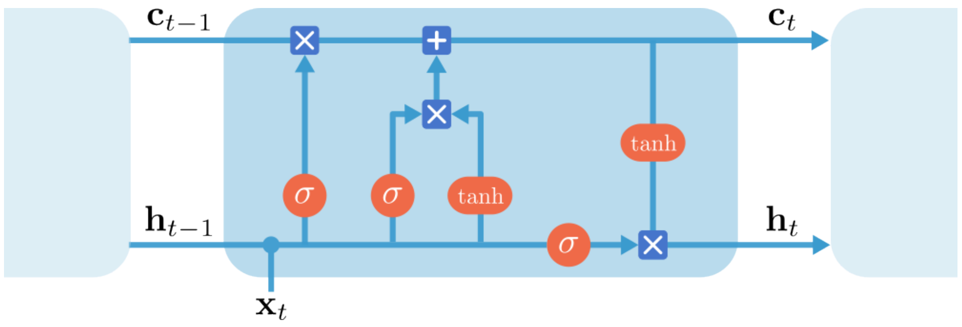


Figure 2.11: Schematic diagram of the LSTM cell. Adapted from [66].

$$\mathbf{f}_t = \sigma(\mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{xf}\mathbf{x}_t + \mathbf{b}_f) \tag{2.15a}$$
$$\mathbf{i}_t = \sigma(\mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{xi}\mathbf{x}_t + \mathbf{b}_i) \tag{2.15b}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{xo}\mathbf{x}_t + \mathbf{b}_o) \tag{2.15c}$$
$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{W}_{xc}\mathbf{x}_t + \mathbf{b}_c) \tag{2.15d}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{2.15e}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{2.15f}$$

where $\mathbf{W}_{hf}$, $\mathbf{W}_{hi}$, $\mathbf{W}_{ho}$ and $\mathbf{W}_{hc}$ are weight matrices applied to the hidden state of the cell; $\mathbf{W}_{xf}$, $\mathbf{W}_{xi}$, $\mathbf{W}_{xo}$ and $\mathbf{W}_{xc}$ are matrices that act upon the input; $\mathbf{b}_f$, $\mathbf{b}_i$, $\mathbf{b}_o$ and $\mathbf{b}_c$ are bias vectors; and $\odot$ denotes element-wise multiplication between two vectors. $\sigma(\cdot)$ corresponds to the sigmoid activation function, which sets the value of $\mathbf{f}_t$, $\mathbf{i}_t$ and $\mathbf{o}_t$ between zero and one.

The long-term cell state $\mathbf{c}_t$ can store information that can remain unchanged across its unfolding along the sequence [66]. The transformations to the value of $\mathbf{c}_t$ (Equation 2.15e) are dictated by the values of $\mathbf{f}_t$ and $\mathbf{i}_t$. If $\mathbf{f}_t$ equals zero, the previous state of the cell is erased. The closer its value gets to one, the more the previous state is preserved. Similarly, an $\mathbf{i}_t$ with a value of one indicates the inclusion of new information (contained in the candidate state $\tilde{\mathbf{c}}_t$) into $\mathbf{c}_t$, whereas a value of zero leaves the state unaffected. $\mathbf{o}_t$ filters out the information contained in $\mathbf{c}_t$ to compute $\mathbf{h}_t$ (Equation 2.15f).

2.6 | Adaptive Graph Convolutional Recurrent Network

The Adaptive Graph Convolutional Recurrent Network (AGCRN) was introduced in [7] as a framework for traffic forecasting. It addresses the challenge of capturing complex spatial and temporal correlations within interconnected time-series. The nodes represent the individual traffic sensors installed on the roads. Instead of utilizing a predefined adjacency matrix $\mathbf{A}$ based on sensor proximity and/or time-series similarity, the authors propose a Data-Driven Graph Generation module, which results in a normalized adjacency matrix that reflects the inter-dependencies between nodes:

$$\tilde{\mathbf{A}} = \text{softmax}\left(\text{ReLU}\left(\mathbf{E} \cdot \mathbf{E}^T\right)\right) \tag{2.16}$$

where $\mathbf{E} \in \mathbb{R}^{N \times F}$ is a learnable node embedding dictionary, where each embedding has length F. The multiplication of $\mathbf{E}$ with its transpose $\mathbf{E}^T$ computes the dot products between every pair of node embeddings, resulting in a square matrix that reflects the similarity between any two nodes. Higher values indicate a greater similarity (or potential connectivity) between nodes. The ReLU function filters out negative similarity scores, ensuring the adaptive adjacency matrix comprises exclusively positive connections. Finally, the softmax operator normalizes every row within the adjacency matrix.

By indirectly learning the undirected adjacency matrix through embeddings, the model has fewer parameters to learn than if the N^2 parameters were learned directly. This contrasts especially when N is large, and F is small. Adopting this approach reduces the need for extensive training data and improves the model's efficiency and scalability, which is crucial for handling large graphs.

The AGCRN cell combines the temporal processing capabilities of an RNN, particularly a Gated Recurrent Unit (GRU), with adaptive graph convolutional techniques to refine the model's spatial analysis. This combination allows the AGCRN to dynamically learn the interdependencies among traffic series and adapt to the evolving patterns in traffic flow data:

$$\mathbf{z}_t = \sigma\left(\tilde{\mathbf{A}}\left[\mathbf{X}_t, \mathbf{h}_{t-1}\right]\mathbf{E}\mathbf{W}_z + \mathbf{E}\mathbf{b}_z\right) \tag{2.17a}$$

$$\mathbf{r}_t = \sigma\left(\tilde{\mathbf{A}}\left[\mathbf{X}_t, \mathbf{h}_{t-1}\right]\mathbf{E}\mathbf{W}_r + \mathbf{E}\mathbf{b}_r\right) \tag{2.17b}$$

$$\tilde{\mathbf{h}}_t = \tanh\left(\tilde{\mathbf{A}}\left[\mathbf{X}_t, \mathbf{r} \odot \mathbf{h}_{t-1}\right]\mathbf{E}\mathbf{W}_h + \mathbf{E}\mathbf{b}_h\right) \tag{2.17c}$$

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t \tag{2.17d}$$

at time t, $\mathbf{z}_t$ is the update gate, calculated using the parameters $\mathbf{W}_z$ and $\mathbf{b}_z$. Similarly, parameters $\mathbf{W}_r$ and $\mathbf{b}_r$ are used to compute the reset gate $\mathbf{r}_t$. $\tilde{\mathbf{h}}_t$ is the candidate for the value of the hidden state $\mathbf{h}_t$. $[\cdot]$ denotes concatenation and $\odot$ denotes element-wise multiplication.

By integrating graph convolutions in the gates of the GRU cell, AGCRN can capture node-specific spatial dependencies in addition to temporal patterns. Each iteration of $\tilde{\mathbf{h}}$ captures a combined representation of the information belonging to a node and its neighbours, enabling it to store and exploit information in the temporal and spatial domains simultaneously.

In the short-term traffic forecasting task, AGCRN outperforms other graph-based RNN models where the adjacency graph is predefined. This suggests that AGCRN's adaptive graph construction mechanism is more effective in capturing complex patterns and relationships between interconnected, dynamic time-series. The success of AGCRN in traffic forecasting highlights its potential applicability across other domains, such as WPF, where turbines directly affect the current and future behaviour of their neighbours.

2.7 | Conclusion

This chapter introduced the fundamental concepts relevant to the rest of the thesis. An overview of turbines and their arrangement in wind farms was provided, outlining the importance and challenges of accurate WPF. The arrangement of turbines in wind farms makes estimating their power generation challenging. This challenge is further complicated when the aim is to forecast this power yield in the short, medium, or long term. As discussed in Chapter 1, the research on this topic is crucial, as obtaining accurate and reliable forecasts is essential for the stability and efficiency of the power grid

This chapter also discussed graphs, which provide a powerful representation of an interconnected series of entities. Graphs can be used to model complex systems and relationships in various domains. However, unlike other systems where the link between nodes is evident, the turbines within a wind farm can be related in numerous ways as evidenced by the approaches reviewed. These approaches are based on pre-defined heuristics: geographical distance and correlation are fixed regardless of the changing characteristics of the wind farm, while the "wake interactions" heuristic aims to reassemble the interactions resulting from the Wake Effect. However, none of these representations fully capture the dynamic and complex nature of wind farm interactions, highlighting the need for further research on the topic.

Both GNNs and RNNs were introduced as two neural network architectures specialized in modelling spatial and temporal dependencies in data, respectively. RNNs are adept at modelling the temporality found in sequences, while GNNs can operate in the graph domain, leveraging graph topology to spread information across its nodes. Each network has demonstrated promising results in its respective domain, leading to the development of new hybrid architectures that combine their strengths. One example is AGCRN, which integrates graph convolutions into an LSTM, enabling it to capture both spatial and temporal dimensions simultaneously.

This study merges the previously reviewed concepts to address WPF in wind farms using GNNs. We explore the creation of a graph from SCADA data to effectively encode the dynamic interactions among turbines and investigate how this graph can be utilized by a model for short-term WPF. Inspired by AGCRN, such a model should also integrate RNNs, as they are the current state-of-the-art for time-series modelling. The resulting model should produce forecasts at the turbine level, enabling its integration with control methods and use for operational decision-making.

The next chapter will review the current literature on WPP and WPF, emphasizing the most significant studies on these topics. Many of the described models build upon the concepts discussed in this chapter, underlining their relevance and applicability to our research.

3 Literature Review

This chapter explores WPF and Wind Power Prediction (WPP) methods for systems ranging from individual turbines to clusters of wind farms. §3.1 introduces the topic and a framework for classifying WPF and WPP methods. §3.2 introduces Traditional Approaches, where Physics-Based Methods offer accuracy but are resource-intensive, and Statistical Methods are effective for short-term forecast windows but inaccurate for longer ones and inflexible to design variations. Artificial intelligence approaches, described in §3.3, address the limitations of traditional methods, introducing modules specialized in modelling temporal dependencies in data. §3.4 reviews different graph formulations allowing GNNs to exploit wind systems' spatial dependency. GNN-based WPF approaches are then categorized based on the order and integration of their spatial and temporal sub-modules. The chapter concludes with the discussion in §3.5.

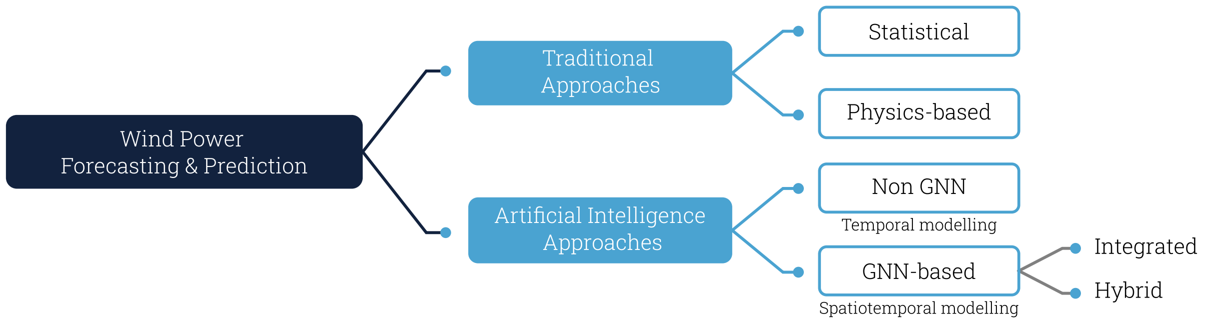


Figure 3.1: Classification of the methods for WPF and WPP.

3.1 | Introduction

Wind-based systems' power production has been a research focus in the energy field for the last 20 years [84]. In this context, a system may denote a single wind turbine, a collection of wind turbines (such as a wind farm), or even a cluster of wind farms. During this exploration, emphasis has been placed on two tasks: WPP and WPF. WPP estimates the power a wind system produces at a specific time, utilizing features corresponding to that moment, including sensor data, meteorological variables, and the power output from associated systems. In contrast, WPF estimates future wind production values, drawing on historical data and other forecasts to project power production at upcoming intervals. In the forecasting task, the past values of the target feature(s) are also considered. Despite the distinct temporal scopes, the synergy between these tasks is evident, as the accuracy of predictions inherently influences the reliability of future forecasts. The methods to tackle these tasks are classified into Traditional and Artificial Intelligence approaches. We adhere to this classification and further categorize Artificial Intelligence approaches by integrating GNNs into their architecture. Our framework for categorising these methods is shown in Figure 3.1.

3.2 | Traditional Approaches

Traditional WPP and WPF approaches rely on physical and statistical models. Physics-based approaches work by inputting down-scaled weather data into complex mathematical models such as the Jensen model [42] and the Jimenez model [44]. These methodologies typically generate wind speed estimates for the individual turbines, which are then mapped to the corresponding output power through the power curve provided by turbine manufacturers. As these methods build upon tested laws and specifications, they are easily interpreted and accurately reflect physical interactions within wind farms, such as the wake effect. Regarding WPP, physical methods are heavily used for optimizing the layout of wind farms [23, 22] and control purposes [1, 8]. In the context of WPF, they are suitable for long-term forecasting, given their high prediction accuracy [24, 62]. The drawback of these approaches lies in their dependence on detailed atmospheric data and their resource-intensive nature, resulting in higher costs and making them ineffective for short-term WPF due to their extensive running time [95].

Statistical models adopt a data-driven strategy for WPF by capturing linear relationships in the target time-series and other correlated variables. Statistical methodologies utilize the autoregressive moving average model (ARMA) [21] and its variants: the autoregressive integrated moving average model (ARIMA) [76, 34], which introduces differencing for stationarity; the seasonal ARIMA model [83], which additionally accounts for seasonality in the time-series; and the multivariable-compatible VARMA model [97], which produces a forecast for multiple systems. Additional models within this category follow the Bayesian approach [63] or apply Kalman filters [2] to forecast wind power or wind speed for single turbines. While statistical models are straightforward to implement and cost-effective for short-term forecasting, their performance diminishes significantly for multi-step WPF as the forecast window grows [16]. Furthermore, these models struggle with long time-series and are ineffective at considering multiple time-series as input. Finally, these approaches may not be well-suited for the task, given their reliance on assumed linear relationships between features, which is not typically observed on wind power time-series.

3.3 | Non-GNN Approaches

For WPP, Neural Network-based approaches centre around surrogate modelling. These models are trained with data derived from high-fidelity physical simulations to approximate their outcomes, significantly reducing computational time while sacrificing accuracy. The work in [90] estimates the power curve of a wind turbine given the wind speed and direction using a Feed-Forward Neural Network (FNN). In [3] and [69], the authors approximate the wake fields for single turbines given large high-fidelity wake images.

3.3.1. Base Neural Approaches

The earliest neural approaches for WPF utilize FNNs that operate on historical variables to produce single-step forecasts [14] or multi-step forecasts [13] for individual turbines. Some works preprocess the time-series using statistical methods before running FNNs on the result [15, 57]. Although these models outperform statistical models [57], they struggle to handle temporal data, resulting in low performance when forecasting more than a couple of time-steps into the future. FNNs cannot capture sequential dependencies and temporal patterns, treating each input independently. Their fixed-length parameters are unsuitable for varying-length time-series, and the absence of memory retention hinders recognizing time lags. RNNs (§2.5) have emerged to address these shortcomings, offering architectures tailored for the dynamic and sequential nature of temporal data, thereby enhancing accuracy in WPF, as we shall see in §3.3.2 and §3.4.2.

3.3.2. Temporal Models

RNNs, particularly LSTMs, have proven effective at time-series forecasting, typically outperforming statistical methods and FNNs [77, 60], and have been extensively used for WPF. In [89], the authors carry out dimensionality reduction on weather data and implement an LSTM to perform short-term WPF. In [55] and [56], wind speed signals undergo a decomposition into low-frequency and high-frequency components. Forecasting of the low-frequency term for a single-speed signal is accomplished using LSTMs. In

[75], the authors integrate wavelet activation kernels into the activation functions for the gating mechanism of LSTMs, which take multiple features as input. This modification leads to enhanced short-term WPF performance.

Other variants of RNNs have also been utilized for WPF. The works in [82] and [50] implement bidirectional LSTM approaches. A GRU is utilized in [65], which provides multi-step, multivariate forecasts and encompasses an attention mechanism that reveals a close relationship between wind speed and wind power. All these methods solely use historical power values to produce the estimates, ignoring information derived from covariates or static features.

Another family of neural networks commonly used for WPF are Convolutional Neural Networks (CNNs). These models combine features that are arranged in grid-like structures. Multiple studies leverage CNNs for temporal feature extraction from wind-related databases and perform WPF using Neural Networks on the resulting representations [37, 45].

A widespread technique combines the feature extraction capability of CNNs and the temporal nature of LSTMs to carry out WPF. In [59], the wind power time-series of four wind farms is decomposed and fed along meteorological variables to a CNN, which carries out feature extraction. An LSTM is used to forecast the power signal components, which are then reconstructed into a day-ahead forecast for each of the four wind systems. The work in [88] constructs a grid of meteorological features, upon which a CNN extracts features that act as inputs to an LSTM layer responsible for producing a wind farm-level short-term forecast. The authors in [98] replace the LSTM with a Bi-LSTM, which acts directly on the wind speed and wind power measurements.

Recent models combine CNNs with state-of-the-art architectures for sequence prediction, such as the Transformer. Notably, [27] proposes an encoder-decoder architecture, where temporal features are extracted using a Temporal CNN and an Informer block generates a short-term forecast for a single wind turbine.

While the methods discussed earlier address the temporal correlations found within the data, they generate aggregated forecasts for wind systems (i.e. a single output signal), even when the system comprises several subsystems like individual wind turbines. Additionally, the spatial interactions are overlooked, which limits their applicability to interconnected structures such as wind farms. These challenges have instigated a growing interest in developing models capable of modelling the behaviour of individual subsystems while accounting for the spatial and temporal interrelations they exhibit.

3.4 | GNN-based Approaches

In WPP and WPF, many models have surfaced that integrate GNNs into their architecture. These models can be categorized based on the specific task they address, how the graph representation of the wind system is constructed, and the components they incorporate into their architectures. An overview of these works is included in Table 3.1.

3.4.1. Graph Formulation

Most studies follow a standard approach for representing wind farms as graphs. All the reviewed methods consider individual turbines as nodes, and several define undirected edges based on their physical proximity or feature correlations. In [92], the authors consider all turbines within a certain distance as neighbours, while [20] produces a sparser adjacency matrix following the n-closest neighbours approach. Time-series correlations are also utilized to build the graphs, such as in [96]. In [58], two graphs are defined: the first sets edges between turbine pairs with wind speed correlations above a predefined threshold, and the second uses the wind farm's electrical wiring diagram to establish edges.

The previously discussed models rely on predefined heuristics, commonly used across domains where the interactions result straightforward and interpretable (e.g. road traffic). These heuristics may not adequately capture the complex interactions in wind systems, where multiple variables influence power generation. Here, the graph structure should additionally be dynamic, reflecting the continual changes in environmental conditions. Static models fall short because they cannot adapt to the shifting interactions between turbines as wind patterns evolve.

In response to this, some methodologies introduce directed edges to represent wake interactions between turbines within wind farms, taking into account not only their physical separation but also the relative angle between the wind and the turbines [10, 12, 33, 68, 94]. The resulting graphs evolve, adding and removing links as necessary. However, this approach requires a set of parameters (e.g. angle thresholds) that are difficult to set empirically and that might as well change over time.

Recent methods can adaptively learn the adjacency matrix relating to wind farms. In [31], the power time-series is fed to a GRU, whose last hidden layer is then used as input for a self-attention mechanism, outputting the graph structure. The authors in [80] combine a predefined adjacency matrix with two learnable matrices of the same size; one that results from the dot product of all the pairwise combinations of the node embeddings, and a second one, where all the individual entries are treated as parameters and learned from the data. As a shortcoming, these models need large amounts of training data due to their complexity and the extensive number of parameters that arise from individually learning the matrix entries.

3.4.2. GNNs for WPF

The methodologies employed for WPF utilizing GNNs can be systematically classified based on how the temporal and spatial features are combined. In this study, we delineate two classes: i) Hybrid methods, which adapt and combine existing spatial and temporal models, and ii) Integrated methods, developed to extract both types of dependencies simultaneously. A summary is presented in Table 3.1.

Hybrid Models

In hybrid models, the spatial and temporal units are decoupled. These architectures combine existing modules which have proven effective for exploiting the spatial structure from graphs and the temporal features from time-series. One such method is [47], where temporal features are extracted for each wind farm using LSTMs and then combined using spectral graph convolutions to forecast wind speed from 10 minutes to several hours ahead at the station level. Similarly, [58] applies LSTMs to individual turbines, inputting recent wind speeds and operational statuses. The LSTM's final hidden state feeds into two consecutive GCNs that leverage the farm's topology to forecast available

Table 3.1: An overview of the existing GNN-based methods for WPF.

				Forecast		
Type[1]	**Authors**	**Graph nodes**	**Architecture**	**Feature**	**Scale**	**Resolution**[2]
H	Khodayar et al. [47]	wind farms	LSTM, GCN, Rough Set Theory	speed	short-term	MS
H	Liu et al. [58]	turbines	LSTM, GCN	power	ultra-short-term	SS
H	Wang et al. [86]	wind farms	GAT, BiGRU, Transformer	power	ultra-short-term	MS
H	Zhang et al. [94]	turbines	Kalman filter, TCN, GCN, ARIMA	speed	ultra-short-term	SS
I	Bentsen et al. [11]	weather stations	GNN, LSTM, Transformer	speed	ultra-short-term	MS
I	He et al. [31]	weather stations	GNN, GFT	power	ultra-short-term	MS
I	Stańczyk et al.[80]	cities	GCN, TCN	speed	short-term	MS
I	Zhao et al. [96]	wind farms	STGCN	power	ultra-short-term	MS

[1] H: Hybrid, I: integrated
[2] SS: single-step, MS: multi-step

power and power losses per turbine.

Other models employ GNNs to integrate features within a graph, followed by a temporal module for forecasting. The work in [94] introduces a superimposed model for single-step wind speed forecasting at a specific turbine. This model uses ARIMA for unstable components and a GCN for features from eight upstream turbines. A Temporal Convolutional Neural Network then processes the outputs. The model described in [86] uses a graph attention network and a bidirectional GRU, combined with a Transformer layer, for multi-step WPF across multiple wind farms. The results are location-specific forecasts.

The study in [96] uses the Spatio-Temporal Graph Convolutional Network (STGCN) from [91] for WPF. The STGCN combines a graph convolutional layer with two gated temporal convolutional layers to extract spatio-temporal features in a purely convolutional approach. This method proves effective for multi-step, ultra-short-time forecasting at the farm level. Similarly, [80] proposes an architecture that carries out a spatial convolution, followed by a temporal one. This block is applied sequentially to a multivariate dataset to generate multi-step wind forecasts for European cities.

Although hybrid models effectively capture the spatial and temporal dimensions of the data, their fragmented approach can be inefficient, as reflected in their parameter count. Furthermore, none of the mentioned studies justify the specific sequence of merging spatial and temporal features, a factor that could impact overall model effectiveness.

Integrated Models

This category encompasses models wherein temporal and spatial dependencies are simultaneously extracted, showcasing a holistic approach to WPF. The current integrated models for this task transform the time-series to the spectral domain before as part of the methodology. The work in [11] integrates different temporal-based predictors as update functions for a GNN, including the Fast Fourier Transformer. This Transformer analyzes signal periodicity and learns trend components. While updating the GNN with a standard Transformer shows minimal effect, incorporating the Fast Fourier Transformer significantly enhances the accuracy of multi-step wind speed forecasts at 14 offshore weather stations. The authors in [31] convert the learned adjacency matrix into a spectral matrix and use it to run convolutions on the power series in the spectral domain, which are then translated into power forecasts. Although these architectures have proven effective at WPF, the intensive resource requirements of the eigenvalue decomposition and the attention mechanism (in the former case) confine their use to small-scale graphs.

3.4.3. GNNs for WPP

GNN methods for WPP in wind farms are mainly used for surrogate modelling, approximating the physical interactions between turbines and their power outputs in a fraction of the execution time. In [68], the authors introduce a physics-induced bias that, similarly to an attention mechanism, learns to weigh upstream turbines' influence on downstream turbines using the wake model described in [67]. This bias is implemented in the GNN node feature update function. The work in [33] adapts [68] for wake steering control by incorporating yaw angles as node features. The purely data-driven model can compute optimal yaw configurations. In [12], the authors implement a model combining Bidirectional LSTMs and attention-based GNNS that learns to focus on the most relevant upstream turbines for shifting signals among edges and nodes. An analysis of the resulting weights reveals that these reflect the physical intuition of wake interactions. In [10], the same authors develop a Bayesian model with an underlying GNN architecture. Unlike the previously described methods, this model addresses uncertainty by providing a probability distribution for the power output of the wind turbines, derived by sampling the parameters of the GNN.

3.5 | Discussion

This chapter reviewed current methods for WPF. We examined how graph structures are applied to represent wind systems and observed that some methods construct graphs based on physical proximity and feature correlations. Other approaches offer a simplistic model of the wake effect by additionally considering the wind direction. While graph-adaptive methods are promising for leveraging hidden data relationships, they can become overly complex when approximating each entry in the adjacency matrix or when dealing with large-scale wind farms.

Furthermore, we categorized the current graph-based methods for WPF. Hybrid models, which separate the spatial and temporal dimensions and combine them in an arbitrary sequence, are found to be inefficient. In contrast, integrated models merge these dimensions simultaneously within a single component. However, these methods are expensive as they require inverting the adjacency matrix, making them impractical for large-scale systems. We note that none of the graph-based approaches for WPF can integrate static features or future covariates.

In general, the results from the cited studies suggest that GNNs are well-suited to model interactions within wind systems. Specifically, GNNs have the potential to learn the complex interactions occurring in wind farms, as demonstrated by several works focused on WPP. This is promising for WPF, which considers the temporal dependency of the data in addition to the spatial one. As research continues to advance, GNNs are likely to play a crucial role in developing more accurate and reliable WPF methods.

Building on the reviewed works, this thesis introduces a novel graph-based integrated model for WPF at wind farms, which is set apart by two distinctive characteristics. Firstly, the model can integrate multidimensional input data, including multiple time-series (historical and future covariates) and static features for each turbine, to generate accurate power forecasts. It supports multi-step series of different lengths and simultaneously processes data for all turbines. Furthermore, the model employs a GCN-enabled LSTM cell to adapt the adjacency matrix dynamically based on the farm's evolving conditions. The number of parameters is restricted by indirectly learning the adjacency matrix, and the model applies to large-scale wind farms. The seamless integration of the spatial and temporal dimensions results in a holistic approach to WPF.

4 Methodology

This chapter presents the AG-LSTM Network for WPF. It begins with a definition of the problem in §4.1. The proposed model is introduced in §4.2, and its scalability is addressed in §4.3. Next, the process of training the model, employing the sliding window technique is described in §4.4. Lastly, §4.5 provides a discussion of the method.

4.1 | Problem Formulation

We aim to develop a model $\mathcal{F}(\cdot)$ that outputs the power forecast of a wind farm, $\hat{\mathbf{Y}} = [\hat{\mathbf{y}}_{t+1}, \hat{\mathbf{y}}_{t+2}, ..., \hat{\mathbf{y}}_{t+H}]$. Each $\hat{\mathbf{y}}_\mathbf{t}$ contains the wind power estimates for each of the N turbines at a future time-step. To this end, we consider two scenarios:

i) **History-Driven**: The model is given by:

$$\hat{\mathbf{Y}} = \mathcal{F}([\mathbf{y}_{t-O+1}, \mathbf{y}_{t-O+2}, ..., \mathbf{y}_t], [\mathbf{X}_{t-O+1}, \mathbf{X}_{t-O+2}, ..., \mathbf{X}_t], \mathbf{E}; \mathbf{\Theta}) \quad (4.1)$$

where each $\mathbf{y}_\mathbf{t}$ contains the observed power values across the wind farm at a previous time-step and $\mathbf{X}_t$ corresponds to one or more additional time-series that relate to the power series (e.g. wind speed). Both $\mathbf{y}_\mathbf{t}$ and $\mathbf{X}_\mathbf{t}$ represent historical values, starting at $t - O + 1$ and extending up to time-step t. We refer to O as the length of the observation window. $\mathbf{E}$ represents a set of features that remain static through time. The model is parameterized by $\mathbf{\Theta}$.

ii) **Future-Known Values**: Similarly, the model is characterized by:

$$\hat{\mathbf{Y}} = \mathcal{F}([\mathbf{y}_{t-O+1}, ..., \mathbf{y}_t], [\mathbf{X}_{t-O+1}, ..., \mathbf{X}_t], [\hat{\mathbf{Z}}_{t+1}, \hat{\mathbf{Z}}_{t+2}, ..., \hat{\mathbf{Z}}_{t+H}], \mathbf{E}; \mathbf{\Theta}) \quad (4.2)$$

Here, we consider $\hat{\mathbf{Z}}_t$, corresponding to the forecast of one or more features correlated to the power series. For a particular time-step. These features, also called future covariates, are given at the turbine level and span the forecast horizon (H). These can include, for example, wind speeds and temperatures obtained using Numerical Weather Prediction models.

Specifically, we focus on ultra-short-term forecasting for 2 and 4 hours, aiming to aid the wind farm's operation and control strategy. Considering the standard 10-minute resolution of SCADA data, this translates to H taking values of 12 and 24, respectively. We limit O to 24 (4 hours), considering that wind power data displays high volatility and aiming to optimize the model's runtime.

4.2 | Proposed Model

The proposed model is illustrated in Figure 4.1. It adopts an encoder-decoder configuration, where the encoder leverages an Adaptive-Graph LSTM (AG-LSTM) cell responsible for processing the input sequences to capture their underlying temporal and spatial characteristics. The decoder transforms the encoded representation, combining it with the most recent power value, static features and the forecasts of correlated time-series to generate sequence forecasts. Moving forward, we will refer to the model as the AG-LSTM Network.

4.2.1. Model Overview

Initially, the power series $[\mathbf{y}_{t-O+1}, ..., \mathbf{y}_t]$ and other features $[\mathbf{X}_{t-O+1}, ..., \mathbf{X}_t]$ belonging to all the turbines are fed into the encoder. This module can process inputs of varying

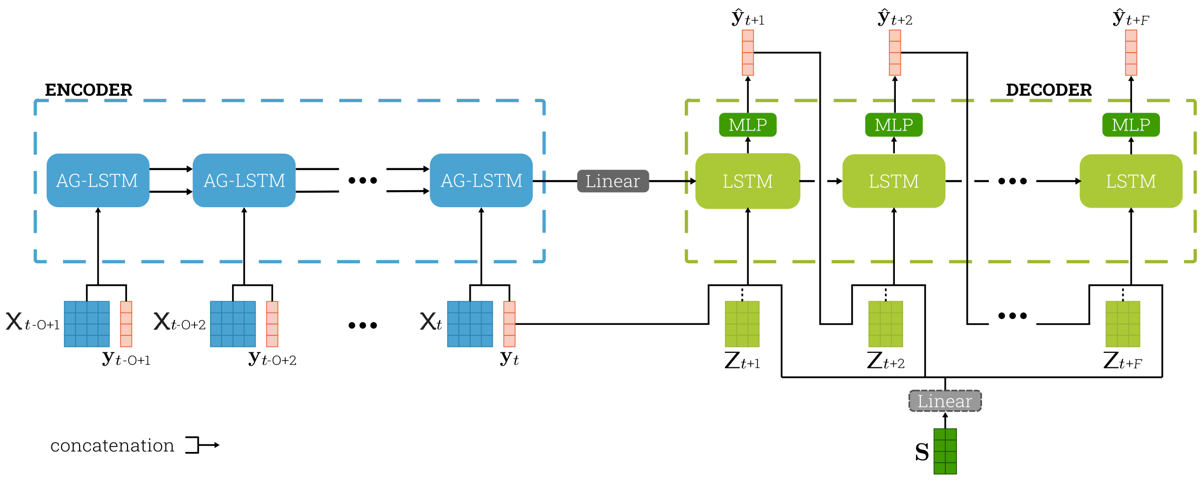


Figure 4.1: Architecture of the proposed encoder-decoder model.

length O, as the internal parameters are shared across the AG-LSTM cells that compose it. The hidden states are iteratively updated for every time-step within the input sequence. The encoders' output contains the last hidden states for each turbine, which combine the information from a temporal and spatial perspective.

Depending on the size of the encoder's and decoder's hidden states, an intermediate mapping is needed. A dedicated module is used to transform the output of the encoder, so its dimensions match the hidden states of the decoder. The parameters of the module are shared across turbines. This results in two cell states, unique to each turbine.

The result of the previous transformation is fed to the decoder, which consists of a set of RNNs serial configuration. The decoder can be run asynchronously for each turbine, as no features or embedding are shared across turbines. This module considers the encoder's last hidden states as initial states and recurrently updates them considering the information corresponding to each future time-step. This information consists of the scalar corresponding to the last power forecast (last model output) $\hat{\mathbf{y}}_{t-1}$, an embedding of the static features $\mathbf{E}$, and optionally, the future covariates $\hat{\mathbf{Z}}_t$. Given that there is no previous power forecast for one-step into the future, the first cell takes the last actual power output $\mathbf{y}_{t-1}$ instead.

As in the encoder, the sequential arrangement of LSTM cells in the decoder enables the model to adjust to various forecast windows. Unlike the encoder, which outputs the state of the last cell, the decoder generates an output for each time-step. This output is transformed by running the hidden states through a MLP, which uses shared weights across different time-steps and turbines. The concatenation of these results comprises the final output of the model, $\hat{\mathbf{Y}} \in \mathbb{R}^{N \times H}$.

An in-depth description of the individual components of the model is provided below.

4.2.2. Adaptive-Graph LSTM Cell

The core component of our model is the AG-LSTM cell, inspired by the cell initially introduced in AGCRN (§2.6). The module we propose is depicted in Figure 4.2. It can represent and combine node features in the spatial and temporal dimensions when applied sequentially as an encoder. The result is a condensed representation of the sequence, key for time-series forecasting.

The AG-LSTM cell generates a dynamic adjacency matrix at every time-step, which combines features and past hidden states among neighbouring nodes to update the cell gates and state. As a first step, the input features for node n at time-step t are transformed into two embeddings, $\mathbf{e}_{1,t}^{(n)}$ and $\mathbf{e}_{2,t}^{(n)}$:

$$\mathbf{e}_{1,t}^{(n)} = [\mathbf{x}_t, y_t]\mathbf{W}_{e,1} + \mathbf{b}_{e,1} \tag{4.3a}$$

$$\mathbf{e}_{2,t}^{(n)} = [\mathbf{x}_t, y_t]\mathbf{W}_{e,2} + \mathbf{b}_{e,2} \tag{4.3b}$$

$\mathbf{W}_{e,1}$ and $\mathbf{W}_{e,2} \in \mathbb{R}^{F\times L}$ transform the input information into embeddings of size L, whereas $\mathbf{b}_{e,1}$ and $\mathbf{b}_{e,2} \in \mathbb{R}^{L}$ act as bias terms. These four terms are learnable and shared across nodes and time-steps. $[\cdot]$ denotes the concatenation operation. The embeddings for all nodes make up the matrices $\mathbf{E}_{1,t}$ and $\mathbf{E}_{2,t}$:

$$\mathbf{E}_{1,t} = \begin{bmatrix} \mathbf{e}_{1,t}^{(1)} \\ \mathbf{e}_{1,t}^{(2)} \\ \vdots \\ \mathbf{e}_{1,t}^{(N)} \end{bmatrix}, \mathbf{E}_{2,t} = \begin{bmatrix} \mathbf{e}_{2,t}^{(1)} \\ \mathbf{e}_{2,t}^{(2)} \\ \vdots \\ \mathbf{e}_{2,t}^{(N)} \end{bmatrix} \tag{4.4}$$

$\mathbf{E}_{1,t}$ is then used to obtain the adjacency matrix $\mathbf{A}_t$:

$$\mathbf{A}_t = \mathbf{E}_{1,t}\mathbf{E}_{1,t}^{\intercal} \tag{4.5}$$

The model utilizes a row-normalized version of the adjacency matrix, $\tilde{\mathbf{A}}_t$ to prevent exploding values throughout the method.

The proposed cell integrates graph message-passing mechanisms into an LSTM, which use $\tilde{\mathbf{A}}_t$ to update the forget, input and output gates. Consequently, a node's present hidden and cell states are derived from the concurrent features and previous states belonging to itself and its neighbours.

$$\mathbf{F}_t = \sigma\left(\tilde{\mathbf{A}}_t\left[\mathbf{X}_t, \mathbf{y}_t, \mathbf{H}_{t-1}\right]\mathbf{E}_{2,t}\mathbf{W}_f + \mathbf{E}_{2,t}\mathbf{B}_f\right) \tag{4.6a}$$

$$\mathbf{I}_t = \sigma\left(\tilde{\mathbf{A}}_t\left[\mathbf{X}_t, \mathbf{y}_t, \mathbf{H}_{t-1}\right]\mathbf{E}_{2,t}\mathbf{W}_i + \mathbf{E}_{2,t}\mathbf{B}_i\right) \tag{4.6b}$$

$$\mathbf{O}_t = \sigma\left(\tilde{\mathbf{A}}_t\left[\mathbf{X}_t, \mathbf{y}_t, \mathbf{H}_{t-1}\right]\mathbf{E}_{2,t}\mathbf{W}_o + \mathbf{E}_{2,t}\mathbf{B}_o\right) \tag{4.6c}$$

$$\tilde{\mathbf{C}}_t = \sigma\left(\tilde{\mathbf{A}}_t\left[\mathbf{X}_t, \mathbf{y}_t, \mathbf{H}_{t-1}\right]\mathbf{E}_{2,t}\mathbf{W}_c + \mathbf{E}_{2,t}\mathbf{B}_c\right) \tag{4.6d}$$

$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tilde{\mathbf{C}}_t \tag{4.6e}$$

$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh\left(\mathbf{C}_t\right) \tag{4.6f}$$

where $\mathbf{F}_t$, $\mathbf{I}_t$ and $\mathbf{O}_t$ correspond to the forget, input and output gates across all nodes. Similarly $\tilde{\mathbf{C}}_t$ is the candidate cell state, $\mathbf{C}_t$ is the final cell state, and $\mathbf{H}_t$ contains the hidden cell state for all nodes. $\mathbf{X}_t$ contains the nodes' features and $\mathbf{y}_t$ are the target feature values. Finally, $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o$ and $\mathbf{W}_c \in \mathbb{R}^{L\times Q}$; and $\mathbf{B}_f, \mathbf{B}_i, \mathbf{B}_o$, and $\mathbf{B}_c \in \mathbb{R}^{L\times Q}$ are learnable parameters. We note that Q, which corresponds to the size of the hidden

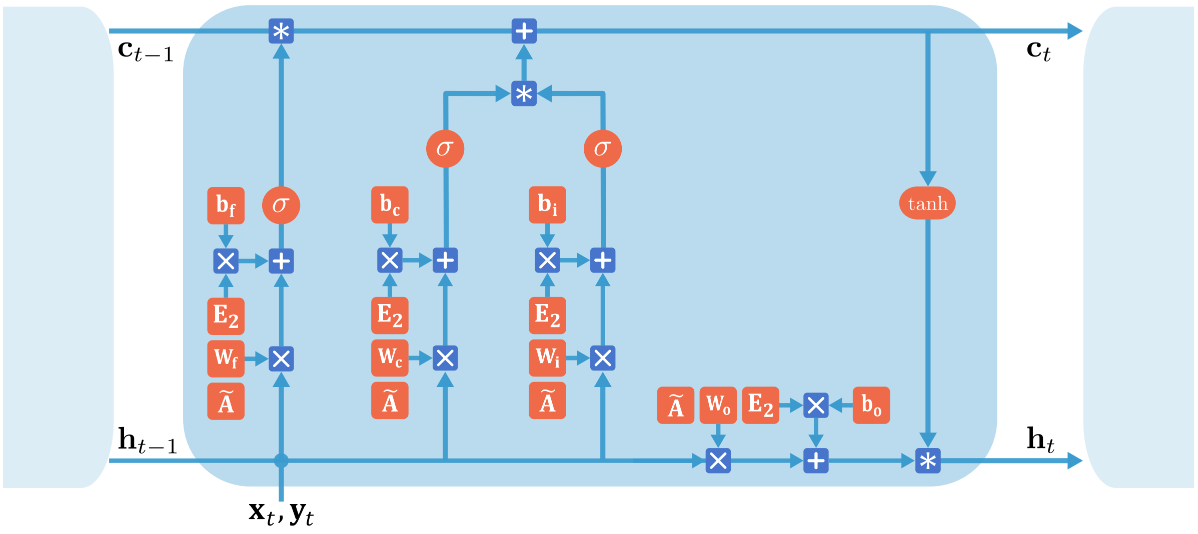


Figure 4.2: Diagram of the AG-LSTM cell.

and cell states of each node must equal $N - F$. As a result, the proposed model is only applicable to cases where $N > F$.

We use two separate embedding matrices for each time-step: $\mathbf{E}_{1,t}$ and $\mathbf{E}_{2,t}$. The reason behind this is that the embeddings have two inherently different objectives. $\mathbf{E}_{1,t}$ is used solely to build the adjacency matrix, which determines graph topology through a data-driven approach. On the other hand, $\mathbf{E}_{2,t}$ enables the transformation of the features and hidden states, allowing the model to develop new representations of the input features. In simple terms, $\mathbf{E}_{2,t}$ defines *what* information is relevant for a node while $\mathbf{E}_{1,t}$ determines *how* it is combined with other nodes.

The AG-LSTM cell can handle input data corresponding to varying nodes and multiple features. The resulting hidden state combines the features of each turbine with those of its neighbours, offering a distinct advantage by providing a more detailed context for each turbine. Later elements of the architecture utilize the last of these states for further processing.

4.2.3. Intermediate Mapping

A linear layer is responsible for transforming the hidden state produced by the last AG-LSTM cell of the encoder into a form suitable for the initial RNN cell in the decoder of the model. This transformation is crucial, mainly when the hidden dimensions of the encoder and decoder cells differ, necessitating an adjustment to ensure compatibility. The mapping is given by:

$$\mathbf{h}' = \mathbf{W}_h \mathbf{h} + \mathbf{b}_h \tag{4.7}$$

where $\mathbf{h}'$ is the initial hidden state of the decoder, and $\mathbf{h}$ is the final hidden state of the encoder for a single turbine. $\mathbf{W}_h$ and $\mathbf{b}_h$ correspond to the weight and bias, respectively. The first dimension of $\mathbf{W}_h$ determines the dimension of $\mathbf{h}'$, whereas its second dimension must match that of $\mathbf{h}$.

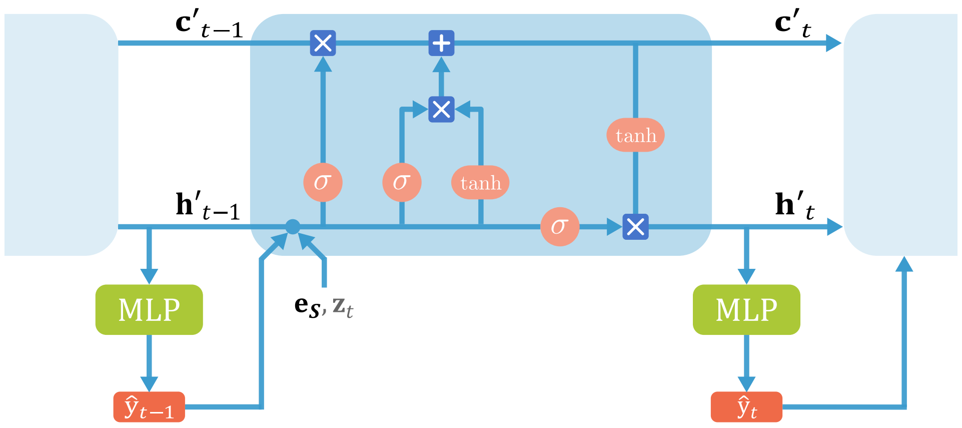


Figure 4.3: Schematic diagram of the decoder for a turbine, which recurrently generates its power forecasts. For each time-step, it takes as input the cell states, the last power forecast, the static features and optionally, the future covariates.

4.2.4. Static Feature Mapping

This module transforms the static features of each turbine into a latent representation. It is beneficial when the static variables contain categorical features such as turbine ID or type. In such cases, these should first be transformed into one-hot representations, and the linear mapping would result in dimensionality reduction. The mapping is given by:

$$\mathbf{e}_s = \mathbf{W}_s\mathbf{s} + \mathbf{b}_s \tag{4.8}$$

where $\mathbf{s}$ are the static variables belonging to a turbine and $\mathbf{W}_s$ and $\mathbf{b}_s$ are the weight and bias of the linear mapping, respectively.

4.2.5. Decoder

The decoder is illustrated in Figure 4.3. This module generates the power time-series forecasts through a sequential configuration containing H RNN cells. We put forward two variations of the encoder, responding to the two problem formulations from §4.1. As the decoder does not combine features across nodes, the equations below correspond to the transformations for a single turbine. The decoder architecture and parameters are shared across all nodes.

For the History-Driven scenario (Equation 4.1), the input of the LSTM results from concatenating the last power forecast $\hat{\mathbf{y}}_{t-1}$, and the static features $\mathbf{e}_s$:

$$\mathbf{x}_t = [\hat{\mathbf{y}}_{t-1}, \mathbf{e}_s] \tag{4.9}$$

The second variation, associated with the Future-Known Value scenario (Equation 4.2), additionally considers the forecast for the turbine's covariates $\mathbf{z}_t$:

$$\mathbf{x}_t = [\hat{\mathbf{y}}_{t-1}, \mathbf{z}_t, \mathbf{e}_s] \tag{4.10}$$

the first cell within the decoder has no previous power forecasts; thus, $\hat{\mathbf{y}}_{t-1}$ is replaced by the last actual power value $\mathbf{y}_{t-1}$. The cell state $\mathbf{c}'_t$ and hidden state $\mathbf{h}'_t$ are updated according to Equation 2.15.

An MLP is employed to transform the hidden state of each decoder cell into the final power forecast. This two-layer MLP maps $\mathbf{h}'_t$ directly to the output, $\hat{\mathbf{y}}_t$, with all time-steps sharing the same parameters for consistency. The transformation is given by:

$$\mathbf{z} = \sigma(\mathbf{W}_{O1}\mathbf{h}' + \mathbf{b}_{O1}) \tag{4.11a}$$

$$\hat{\mathbf{y}}_t = \mathbf{W}_{O2}\mathbf{z} + \mathbf{b}_{O2} \tag{4.11b}$$

where $\mathbf{W}_{O1}$ and $\mathbf{b}_{O1}$ are the weights and bias of the first layer, $\mathbf{W}_{O2}$ and $\mathbf{b}_{O2}$ are those of the second layer, and σ is the ReLU activation function. The output has a size of 1, reflecting the turbine power output at each future time-step.

4.3 | Model Scalability

The dynamic computation of $\mathbf{A}_t$ requires the training of $\mathbf{W}_{e,1}$ and $\mathbf{b}_{e,1}$. These parameters are shared across time-steps and nodes and their size depends on the number of input features and hidden layer size. Consequently, this approach necessitates learning significantly fewer parameters compared to methods where parameters depend on the node count, such as directly learning $\mathbf{A}$. However, our model requires multiplying $\mathbf{E}_{1,t}$ with itself, which translates into computing the dot product among every pairwise combination of nodes. This operation scales cubically, posing challenges for large wind farms.

Some elements of the AG-LSTM Network depend on the size of the graph it acts upon. First off, the cell considers 8 matrices of learnable parameters $\in \mathbb{R}^{L \times Q}$. Because $Q = N - F$, the size of these matrices increases linearly with the size of the graph. Additionally, the multiplication of matrices involved in the update of the gates (Equation 4.6) can explode quickly with larger node counts.

The rest of the model is independent of the graph structure, making it scalable for large graphs. The remaining parameters depend solely on the size of the hidden layers and the number of features, which can pose additional challenges as modern SCADA modules measure hundreds of features.

4.4 | Model Training

Sliding windows are employed to load the training data into the model. The length of the observation window O defines the time-span of the model's input, which considers multiple time-series across all turbines, whereas the forecast window of length H only targets the power feature. An observation window starting at the first available time-step t_1 and ending at t_O is coupled with a forecast window extending from t_{O+1} to t_{O+H}. Both windows are slid, one step forward at a time, creating a set of input-output pairs. This sliding process continues across the entire train dataset, ensuring that each set of inputs and corresponding prediction contains a continuous time-series segment. This approach is depicted in Figure 4.4. To prevent the model from overfitting to the order of the data, the input-output pairs are shuffled in each training epoch.

Algorithm 1 provides the pseudocode for the training procedure using Python syntax.

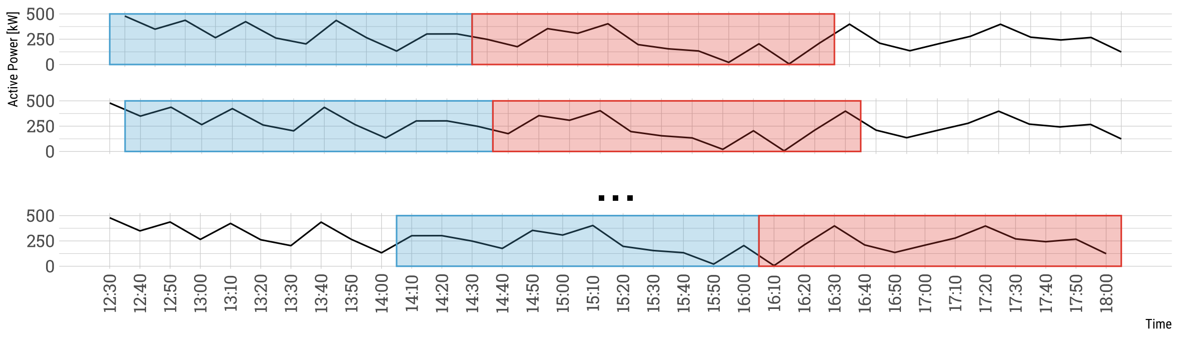


Figure 4.4: Example of the sliding window methodology. At each iteration, the blue window contains the model's input, and the red window includes the target values. These windows are slid until the end of the dataset is reached.

4.5 | Design Insights

The AG-LSTM Network can effectively produce turbine-wise short-term wind power forecasts, given a set of time-series along with static features at turbine level. Through an encoder-decoder configuration, it can deal with input and output sequences of variable lengths. The encoder employs the AG-LSTM cell, which combines LSTM's ability to model sequential data with GNN's spatial information propagation. Crucially, the adjacency matrix is computed dynamically, adapting to the changing environmental conditions at the wind farm. The rationale for the design choices is detailed below.

A graph representation is an effective tool to capture wind farm dynamics. As highlighted in §2.1, the behaviour of a turbine depends on its interaction with the others. GNNs can encode and exploit such interactions, adapting to the varying conditions the farm is exposed to. Furthermore, the changes in an individual turbine's environmental conditions foreshadow the others' behaviour (e.g., changes in wind speed). These changes can effectively be captured by the AG-LSTM cell. The hidden states within the cell allow it to store and update representations for each turbine at every time-step. By integrating the features of nodes at a given time-step with their previous hidden states through graph convolution, the cell harnesses both current and historical data to enhance forecasting accuracy.

In any graph-based model, the adjacency matrix serves as a critical element, defining how information is propagated. As detailed in §3.4.1, existing models utilize different heuristics based on relative turbine position or feature correlation to construct the adjacency matrix. However, we argue that defining the adjacency matrix, rather than learning it, can limit the model's ability to uncover complex turbine interdependencies. To address this limitation, the adjacency matrix within the AG-LSTM cell is dynamically learned during model training. This approach allows the model to uncover and adapt to the underlying structure of turbine interactions, potentially identifying more nuanced relationships than predefined heuristics. Moreover, the resulting matrix can reflect asynchronous conditions, which are difficult to identify through traditional methods like correlation analysis.

Additionally, AG-LSTM cells offer the advantages of traditional LSTM cells, including the retention of information across multiple time-steps, effectively mitigating

Algorithm 1 Model training

```
n_epochs: number of epochs to train for
train_loader: iterator that returns encoder features, decoder features and target values
optimizer: Optimization algorithm
L1_loss: MAE loss function
N: Model is evaluated every N epochs
x_enc_val, x_dec_val, y_val: All input and output pairs of the validation set

for epoch in n_epochs do
    Training
    model.train()                                        ▷Model set to training mode
    for x_enc,x_dec,y in train_loader do
        optimizer.zero_grad()                            ▷Clear gradients
        y_pred = model(x_enc, x_dec)
        loss = L1_loss(y_pred,y)
        loss.backward()                                  ▷Computation of derivative
        optimizer.step()                                 ▷Parameter update
    end for
    Evaluation
    if epoch % N == 0 then
        model.eval()                                     ▷Model set to evaluation mode
        y_pred_val = model(x_enc_val,x_dec_val)
        val_loss = L1_loss(y_pred_val,y_val)
        lr_scheduler.step(val_loss)   ▷Learning rate is reduced by a factor if loss plateaus
        if early_stopping(val_loss) then                 ▷Training halts if loss plateaus
            break
        end if
    end if
end for
```

the vanishing gradient problem common in simpler RNNs. This cell type is also notably flexible, supporting layering and diverse configurations. This inherent modularity makes AG-LSTM cells well-suited for integration into more complex architectures, such as the encoder-decoder configuration.

The proposed cell offers the flexibility of RNN cells to be configured across layers and components, making it suitable for both the encoder and decoder. However, initial experiments showed that using this cell as the core of the decoder negatively impacted results compared to LSTM and GRU cells. Therefore, the proposed cell is only used for encoding. The observed behaviour suggests that the encoder's final states already contain the necessary information from neighbouring turbines, making the static features and future covariates from adjacent turbines less relevant for the forecast.

The encoder-decoder architecture offers two primary advantages for the forecasting model. First, it can accommodate input and output sequences of various lengths, setting it apart from other models that use sequence-to-sequence configurations. This feature is particularly valuable in the context of WPF, where forecasts of different durations serve diverse purposes (see §2.1). Second, this architecture utilizes independent cells for the encoding and decoding processes, enabling differentiated inputs and transformations.

This design allows for each component to specialize in distinct tasks. For WPF, this flexibility allows the integration of additional data into either element. For instance, future covariates can be added to the decoder side, enhancing the model's predictive capability and adapting to the two discussed problem scenarios. Such flexibility and granularity are rare, even among the latest WPF models discussed in the literature.

Model Limitations

While the AG-LSTM Network suits WPF, recognizing its limitations is crucial for interpreting the results and guiding future research directions. The main limitations are as follows:

a) The AG-LSTM Network is susceptible to missing values. As a spatio-temporal model relying on graph convolutions, it assumes complete data for all turbines and time-steps. However, wind power data contains gaps due to turbine shutdowns and abnormal values obtained from the SCADA system. Implementing imputation and anomaly detection methods is essential for improving the model's reliability and accuracy in real-world scenarios.

b) The AG-LSTM Network does not directly account for phenomena that affect turbine power output, such as curtailments. Its current design overlooks these aspects, impacting its accuracy when these appear in a test setting. Future work is needed to predict and account for these operational factors.

c) The method used to generate the adjacency matrix makes it challenging to understand and analyze. While the resulting matrix effectively combines states and features across nodes through edges, relating them to the physical connections between turbines is non-trivial. Moreover, the complexity of the problem and the dynamic nature of the farm's conditions make it challenging to discern patterns in the generated matrices and their evolution over time.

   Furthermore, the matrix might overlook significant links between contiguous turbines subject to similar environmental conditions. Further work could explore a hybrid adjacency matrix that combines the learned approach with heuristic enhancements. This could improve its function and contribute to its explainability.

d) As detailed in §4.3, the model's complexity will scale significantly for farms with large turbine counts. This regards the computation of the adaptive adjacency matrix, and the convolution itself, as they directly relate to the size of the adjacency matrix. Further work could explore the scalability of the method.

5 Evaluation

In this chapter, we outline a series of experiments carried out to evaluate the AG-LSTM Network. Initially, we detail the testing conditions in §5.1, which include the datasets, metrics, and experimental settings. §5.2 presents the experiments designed to benchmark our model against various baselines, examine the model's outputs, and assess how its components and parameters influence the results. Finally, §5.3 offers a discussion of these results, interpreting them within the framework of WPF.

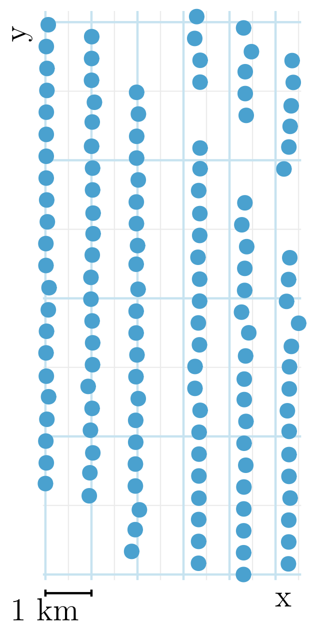
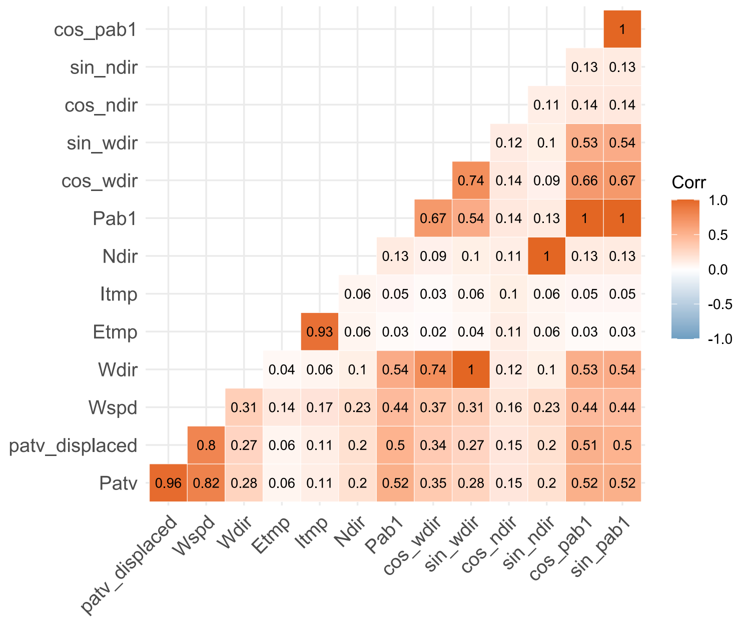


Figure 5.1: Layout of the SDWPF Wind Farm [99]. Each dot represents one of the 134 wind turbines.



	patv_displaced	Wspd	Wdir	Etmp	Itmp	Ndir	Pab1	cos_wdir	sin_wdir	cos_ndir	sin_ndir	cos_pab1	sin_pab1
cos_pab1													1
sin_ndir												0.13	0.13
cos_ndir											0.11	0.14	0.14
sin_wdir										0.12	0.1	0.53	0.54
cos_wdir									0.74	0.14	0.09	0.66	0.67
Pab1								0.67	0.54	0.14	0.13	1	1
Ndir							0.13	0.09	0.1	0.11	1	0.13	0.13
Itmp						0.06	0.05	0.03	0.06	0.1	0.06	0.05	0.05
Etmp					0.93	0.06	0.03	0.02	0.04	0.11	0.06	0.03	0.03
Wdir				0.04	0.06	0.1	0.54	0.74	1	0.12	0.1	0.53	0.54
Wspd			0.31	0.14	0.17	0.23	0.44	0.37	0.31	0.16	0.23	0.44	0.44
patv_displaced		0.8	0.27	0.06	0.11	0.2	0.5	0.34	0.27	0.15	0.2	0.51	0.5
Patv	0.96	0.82	0.28	0.06	0.11	0.2	0.52	0.35	0.28	0.15	0.2	0.52	0.52

Figure 5.2: Average correlation across the features in the SDWPF dataset.

5.1 | Experimental Setup

In this section, we outline the conditions under which the proposed model was evaluated. §5.1.1 introduces the SDWPF and Penmanshiel datasets, their preprocessing and split. §5.1.2 enumerates the baseline models. §5.1.3 describes the metrics employed for model evaluation. Lastly, §5.1.4 specifies the hardware and model parameters used to train and test the proposed model.

5.1.1. Datasets

To evaluate our model, we use two publicly accessible datasets featuring turbine-level sensor readings from two wind farms. A brief overview of each follows.

SDWPF Dataset

The Spatial Dynamic Wind Power Forecasting (SDWPF) Dataset [99] contains the static and dynamic information of a 134-turbine Wind Farm in China. This dataset was introduced in 2022 as part of a mid-term WPF challenge. Figure 5.1 depicts the farm's layout. The values in the dataset were registered by the farm's SCADA system over a 245-day period with a resolution of 10 minutes. The static features comprise the relative x and y coordinates for each turbine. Additional details about the wind farm, such as its actual location or the capacity of its turbines are unknown to us. The dynamic features in the data are summarised in Table 5.1. Additional visuals of the SDWPF dataset are included in Appendix A.

Table 5.1: Overview of the dynamic features of the SDWPF dataset, available for each turbine and time-step. Summary considers the data without any preprocessing.

Affected Feature	Units	Mean	Min	Max	S.D.	% of anomalies[1]
Wind speed	m/s	5.0	0.0	26.3	3.4	6.3
Yaw angle	$^\circ$	0.5	-3030.5	2266.9	31.6	1.0
External temperature	$^\circ$C	41.1	-273.0	394.3	85.3	1.0
Internal temperature	$^\circ$C	27.4	-273.2	324.2	18.3	1.0
Nacelle orientation	$^\circ$	188.6	-884.9	700.6	163.0	1.0
Blade angle 1	$^\circ$	26.9	-10.0	100.0	38.8	20.8
Blade angle 2	$^\circ$	26.8	-10.0	100.0	38.8	20.8
Blade angle 3	$^\circ$	26.8	-10.0	100.0	38.8	20.7
Reactive power	kW	-13.2	-625.0	485.2	70.4	1.0
Active power (Patv)[†]	kW	350.4	-9.3	1567.0	425	29.8

[1] Anomalies are determined according to the dataset specification.
[†] Target feature

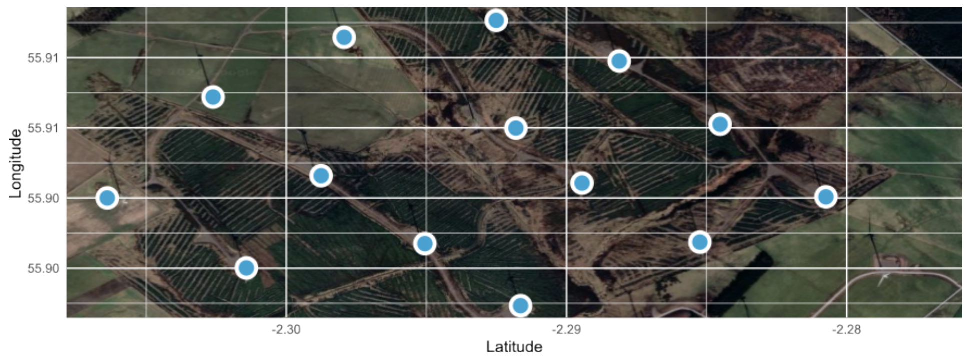


Figure 5.3: Satellite view of the Penmanshiel farm, where dots are overlaid on the 14 wind turbines.

Penmanshiel Dataset

The Penmanshiel onshore wind farm [71], located in Grantshouse, UK, features 14 turbines, as shown in the satellite image in Figure 5.3. Each turbine has a 2,050 kW capacity and an 82-meter diameter. The corresponding dataset includes SCADA readings for every turbine, captured every 10 minutes, from 2016 to 2021. An overview of the relevant features is shown in Table 5.2. Appendix B contains additional exploratory views of the data.

To reduce training time, we examine a single year of this dataset in our study. We select the first 245 days of the first available year to match the temporal span of the SDWPF dataset.

Data Preprocessing

Approximately one-third of the power values in the SDWPF dataset and 14% in the Penmanshiel dataset are marked as anomalies, according to Table 5.3. These anomalies tend to occur simultaneously across turbines due to wind farm shutdowns and maintenance. Overlooking these anomalies would degrade the performance of any graph-based method [100], so they must be addressed. There are two conflicting approaches: some researchers advocate for their exclusion, while others suggest smoothing or transforming them to reduce their negative impact on the model. Given the dataset's spatio-temporal characteristics, excluding anomalies would significantly reduce it to a series of non-continuous

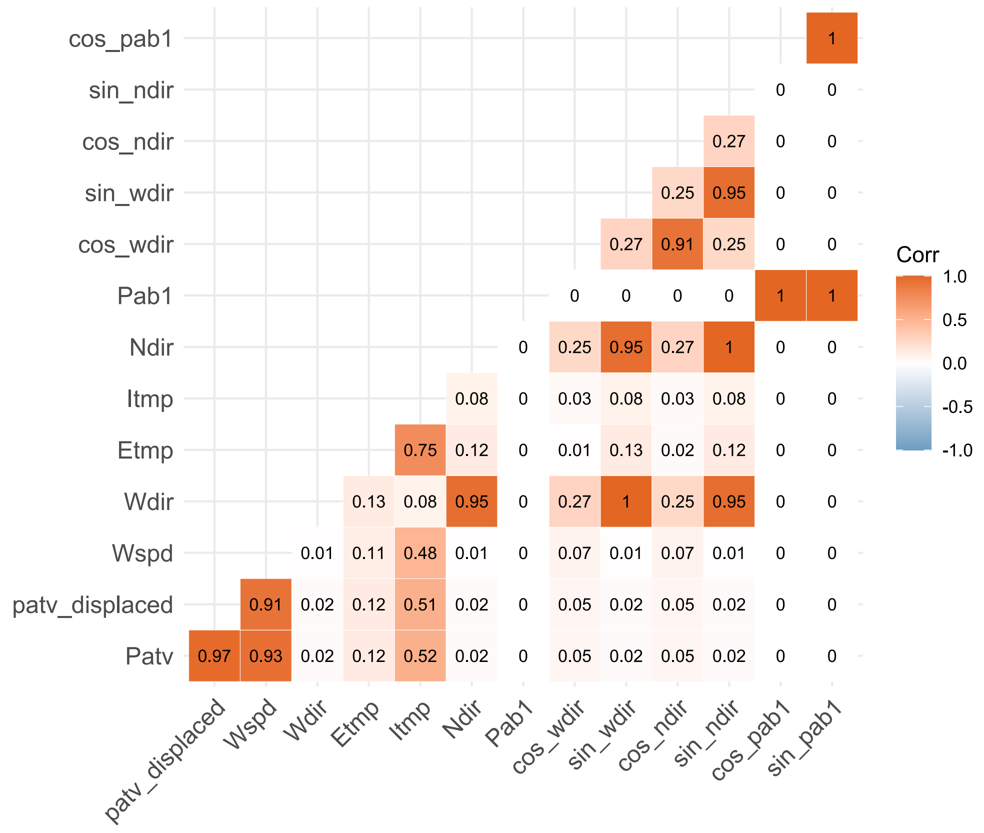


Figure 5.4: Correlation across the features in the Penmanshiel dataset.

Table 5.2: Overview of the dynamic features of the Penmanshiel dataset.

Affected Feature	Units	Mean	Min	Max	S.D.	% of anomalies[1]
Wind speed	m/s	7.1	0.07	30.4	3.6	0.5
Yaw angle	°	21.3	-180	179.9	36.8	0.5
External temperature	°C	18.2	6.7	30.0	3.8	0.5
Internal temperature	°C	18.8	2.2	38.9	4.8	0.5
Nacelle orientation	°	21.2	-180	179.9	31.1	0.5
Blade angle 1	°	90.0	90.0	90.0	0.0	0.0
Blade angle 2	°	90.0	90.0	90.0	0.0	0.0
Blade angle 3	°	90.0	90.0	90.0	0.0	0.0
Reactive power	kW	-162.7	-830.3	404.5	124.7	0.5
Active power (Patv)[†]	kW	653.5	-17.7	2,061.9	667.5	14.4

[1] Anomalies are determined according to the dataset specification.
[†] Target feature

time-steps. For this reason, we advocate for replacing these anomalous values for each turbine, considering its own valid data entries.

When missing or invalid values are present in the data, linear interpolation is typically used for wind speed and wind power time-series [11, 12]. Given its straightforward nature and decent performance in filling gaps in wind series [29], we adopt this method as well. We replace the anomalous power values for each turbine with a linear interpolation between the last valid value and the next available value in chronological order.

Yaw, nacelle, and blade angles are converted to radians, with additional features

Table 5.3: The criteria for identifying data anomalies, as originally outlined in [99]. We apply these rules to both datasets in our evaluation.

Affected feature(s)	Rationale
All	Values marked as NA due to external issues or SCADA system failures.
Active power, blade angles	The angle for any blade is larger than 90°.
Active power, nacelle orientation	The nacelle orientation is not in [-720°, 720°].
Active power, yaw angle	The yaw angle is not in [-180°, 180°].
Active power	Values smaller than zero.

derived from their sine and cosine components. Similarly, the time of day is transformed into a cyclical variable to produce sine and cosine values. Given that the data only covers a part of the year and the data splits do not overlap chronologically, day and month are omitted from the model inputs.

To account for the variability and different scales, and to facilitate model training, all features were normalized through min-max scaling. The active power values smaller than 0 were set to 0 across all turbines and time-steps.

Feature Analysis

Figure 5.2 and Figure 5.4 depict the correlations among processed features for the SDWPF and Penmanshiel datasets. Wind Speed strongly correlates with active power, and wind direction moderately correlates with active power in both datasets. In the SDWPF dataset, transformed blade angles show moderate correlations with the target time-series. However, these features are constant in the Penmanshiel dataset, rendering them obsolete for forecasting other features. The turbine's internal temperature exhibits a weak correlation in the SDWPF dataset and a moderate correlation in the Penmanshiel dataset.

The Appendix includes additional views of both datasets, such as the Partial Auto Correlation Function plot for the power time-series. Both wind farms exhibit similar behaviours, with the one-step lagged power feature highly correlating with its unlagged version, as shown in the correlation plots ("patv_displaced"). The autocorrelation of the power time-series drops significantly after two steps and becomes non-significant after 9 time-steps for the SDWPF dataset and 10 time-steps for the Penmanshiel farm.

The previous insights are crucial for the experimental settings of the model, specifically in selecting the covariates used in our experiments and determining the observation window size for the model's input.

Data Split

The dataset is divided chronologically into training, validation and testing subsets as in [7]. The train set comprises the first 171 days (~70%), followed by 25 days for validation (~10%), and the last 49 days (~20%) are set for testing. This is illustrated for the power time-series of a single turbine in Figure 5.5, but carried out considering all features and turbines. We note that this division differs from the split detailed in [99], as the original one does not consider a training subset.

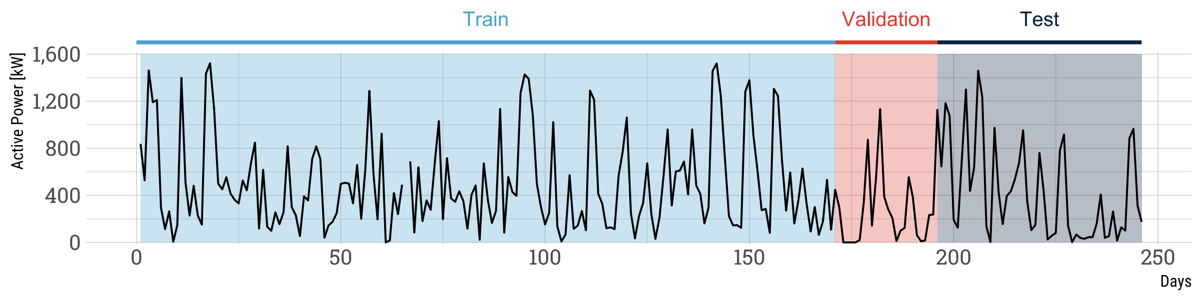


Figure 5.5: Example of the split used for the data, illustrated through the power time-series of a sample turbine.

5.1.2. Baseline Models

To contextualize the performance of our model, we benchmark it against established baselines known for their robust performance in WPF or related forecasting tasks. We group these models according to the classification introduced in Chapter 3.

Traditional Approaches

- **VARIMA** [78]. VARIMA is an extension of ARIMA for multivariate time-series. Because of its statistical foundation, VARIMA contrasts with the neural networks' ability to capture complex interdependencies and nonlinearities. The optimal configuration for this model is selected using the Hyndman-Khandakar algorithm, implemented in [38].

Non-GNN Methods

- **Encoder-Decoder LSTM**. This architecture, described in §2.5.1, effectively models long-range dependencies in sequential data through a series of RNN cells. By comparing our model to this foundational configuration, we can highlight the improvements obtained by the specialized modules.
- **Temporal Fusion Transformer (TFT)** [53]. TFT combines an encoder-decoder configuration with specialized components such as multiple encoders, variable selection networks, static enrichment GRNs, and self-attention mechanisms. This structure excels in multi-horizon time-series forecasting by effectively integrating historical data, and optionally static features and future covariates. Recent works show promise regarding its application in wind-related forecasting applications [43, 87].
- **Time-Series Mixer (TSMixer)** [17]. This novel model utilizes a sequence of MLPs to alternately execute time-mixing and feature-mixing operations on multivariate time-series. Additionally, an enhancement described in the same paper enables the architecture to integrate static information and future covariates. TSMixer has demonstrated superior performance on traffic and product sales forecasting.

GNN-based Approaches

- **Adaptive Graph Convolutional Recurrent Network** [7]. As introduced in §2.6, this model develops an RNN cell that integrates the fundamental mechanics of GNNs. Since our proposed AG-LSTM cell builds on the AGCRN cell, using the original model as a baseline enables evaluating the modifications made to the cell itself, and its integration to the proposed encoder-decoder architecture. The original method only utilizes the power time-series excluding other historical features, future time-series, and static information.

- **Deep Multi-relational Spatio-Temporal Network (DMST)** [51]. This model merges historical wind power data from various wind farms through a graph that integrates distance-based and similarity-based adjacency matrices. The resulting embeddings, along with historical covariates, are processed using a series of GRU cells in an encoder-decoder configuration. Each cell also receives an embedding that encodes the ID of each turbine. DMST is a submodule of the network that secured first place in the KDD Data Cup 2022, which focused on long-term WPF using the SDWPF dataset.

To establish the baseline to compare the performance of our model, the deep learning methods, both Non-GNN and GNN-based, are optimized using a grid search over their hyperparameters. Hidden layer values of 16, 32, 64, and 128 are considered. Other hyperparameters are kept at the optimal values from their corresponding implementations. The settings used for model training can be found in§5.1.4.

5.1.3. Evaluation Metrics

The primary metrics used for assessing the performance of WPF models include the Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE) [28, 70]. MAE results particularly advantageous because:

a) The magnitude of the error is directly interpretable since it's expressed in the same units as the target variable.

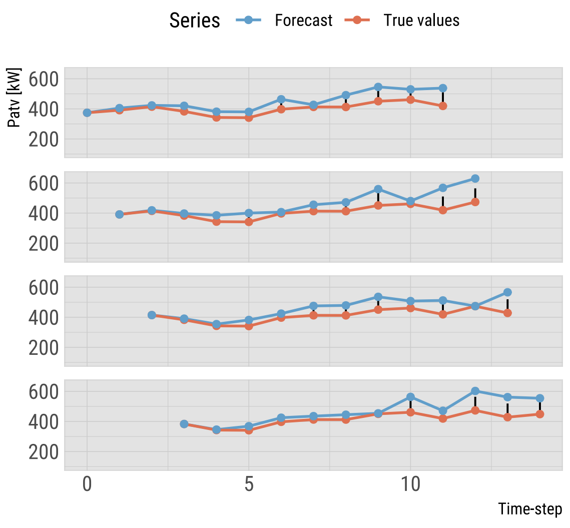
b) It is robust to outliers [4], which is crucial for wind power series that often exhibit abrupt shifts, especially with changes in wind patterns or turbine shutdowns.

Given a forecast window of size H that starts at time-step t_0, the corresponding MAE for turbine i, is denoted $\text{MAE}^i_{t_0}$. The total MAE of the turbine, MAE^i, is computed as the average of its errors across the entire time window. The equations for these two terms follow:

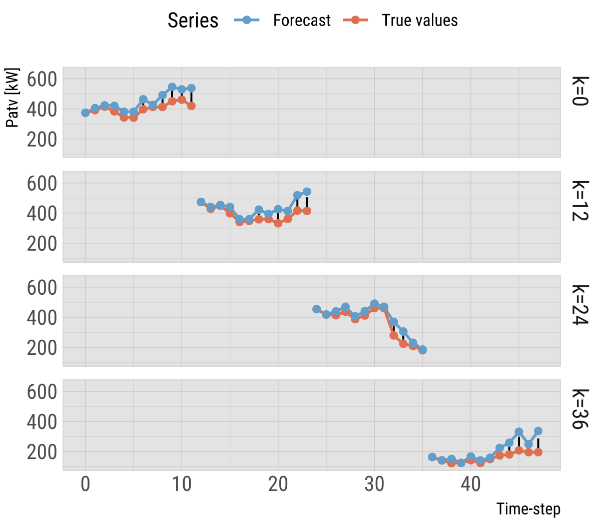
$$\text{MAE}^i_{t_0} = \frac{1}{H}\sum_{h=0}^{H-1} |\hat{y}^i_{t_0+h} - y^i_{t_0+h}| \tag{5.1a}$$

$$\text{MAE}^i = \frac{1}{K}\sum_{k=0}^{K-1} \text{MAE}^i_{t_0+k} \tag{5.1b}$$

where $\hat{y}$ is the forecasted power value, y is the actual power value and K is the number of sliding windows that fit into the dataset. We put forward a special case of the error for a turbine, which aims to reduce redundancy in the result by omitting the overlaps existing among the forecasted values within the sliding windows. Our metric, dubbed disjoint MAE (dMAE), is given by:



(a)



(b)

Figure 5.6: Difference between MAE^i (a) and dMAE^i (b). For the computation of the MAE of a single turbine, the forecasting window is slid by one unit in order to consider every possible forecast within the dataset. On the other hand, the dMAE of a turbine excludes overlapping forecast windows; and the forecast window is slid by H units each time.

$$\text{dMAE}^i = \frac{1}{K} \sum_{k=0}^{K-1} \text{MAE}^i_{t_{0+Hk}}. \tag{5.2}$$

In essence, we only consider the forecasts whose starting time-step is a multiple of H. This makes the windows sparser and avoids considering the same true values multiple times for the error computation. Figure 5.6 explains the difference between MAE^i and dMAE^i. Finally, the error for the wind farm is computed as the average of the error of each turbine:

$$\text{dMAE} = \frac{1}{N} \sum_{n=1}^{N} \text{dMAE}^i \tag{5.3}$$

N represents the total number of turbines within the wind farm.

Because of the differing characteristics between wind farms (such as turbine capacity) and the unique aspects of their individual time-series data, a metric is necessary to evaluate the performance of a model that processes them. In response to this requirement, and following the recommendations from [70], we additionally report a scaled version of the dMAE:

$$\text{scaled dMAE} = \frac{\text{dMAE}}{\overline{\text{Patv}}} \tag{5.4}$$

where $\overline{\text{Patv}}$ is the average power output across all turbines and timesteps. The $\overline{\text{Patv}}$ is 371.2 kW for the SDWPF dataset and 443.0 kW for the Penmanshiel dataset.

5.1.4. Experimental Settings

All tests were performed with Pytorch v2.1.0, using a Ryzen 9 6900HS CPU and an NVIDIA 16Gb GeForce RTX 3080 GPU. Model training and selection were made considering L1 Loss, in line with the previously outlined evaluation metric. We adopted the Adam Optimizer, with an initial learning rate of 0.001. The learning rate was reduced by 50% whenever the validation loss plateaued over 10 epochs. Model training was carried out over 150 epochs. The batch size was set to 128 samples for the SDWPF dataset and increased to 1048 samples for the Penmanshiel dataset, as the latter farm has significantly fewer turbines.

A grid search was conducted to determine the size of the model's parameters, testing values of 16, 32, 64, and 128. As a result, the embedding size of the encoder L and the hidden state of the decoder were set to 32 and 64, respectively. Additionally, the hidden layer size of the output MLP was set at 128. Finally, after additional experimentation, the embedding size for static features was fixed at 5.

We selected the features with the highest correlation to wind power as historical features (see §5.1.1). These are the sine and cosine components of the wind direction, the sine and cosine components of the angle of the blades, the turbine's internal temperature and the wind speed. We considered the turbine's x and y relative coordinates as static variables. The noise, added to the scaled features, was derived from a Gaussian distribution with a mean of 0. The standard deviation was set to 0.01 for one-step-ahead values and increased by 0.02 for each subsequent time-step. The transformed time-of-day variable was also considered a future covariate.

5.2 | Results

This section contains a comprehensive analysis of the output of the AG-LSTM Network. We first compare its results with those from the baselines for the two problem scenarios and datasets, presenting the average outcomes of five runs with different random seeds. §5.2.1 focuses on analyzing sample outputs of our model, identifying patterns common in forecasting models. In §5.2.2, we look at the adjacency matrix generated from the test data, identifying its patterns and linking it to the wind farm. Finally, §5.2.3 and §5.2.4 contain the ablation and sensitivity analyses, respectively.

History-Driven Scenario

The numerical results for the History-Driven scenario are shown in Table 5.4 and Table 5.5. The proposed method obtains the lowest error in both forecasting scenarios for both datasets, followed by the NN baselines and VARIMA in that order. This underscores the effectiveness of an encoder-decoder framework integrating GNNs via a learned topology for WPF. As described in §2.1, the error for all models increases with the forecast window, due to error accumulation and the fact that conditions for nearer time-steps are more similar to a specific time-step than those further into the future.

Models utilizing a shorter observation window ($O = 12$) tend to yield better accuracy compared to those fed with 4 hours ($O = 24$). This trend is consistent with the autocorrelation of the wind power series documented in Appendix A, suggesting that recent patterns are representative of short-term future conditions. Supplying the models

Table 5.4: Results for the History-Driven Scenario on the test-split of the SDWPF dataset. All results are averaged over 5 runs.

		SDWPF			
		H=12 (2 hours)		**H=24 (4 hours)**	
Model	**O (hrs)**	**dMAE** (SD)	**s-dMAE**	**dMAE** (SD)	**s-dMAE**
VARIMA	12 (2)	113.8 (11.1)	30.7%	144.1 (14.5)	38.8%
TSMixer	12 (2)	104.73 (8.4)	28.2%	147.5 (14.3)	39.7%
TSMixer	24 (4)	101.33 (8.1)	27.2%	146.9 (14.2)	39.6%
Enc-Dec LSTM	12 (2)	103.5 (11.2)	27.9%	138.5 (14.8)	37.3%
Enc-Dec LSTM	24 (4)	102.9 (11.6)	27.7%	154.2 (16.7)	41.5%
DMST	12 (2)	96.0 (7.8)	25.9%	130.8 (11.6)	35.2%
DMST	24 (4)	96.1 (7.9)	25.9%	127.5 (11.3)	34.4%
AGCRN	12 (2)	98.3 (10.7)	26.5%	127.7 (13.9)	34.4%
AGCRN	24 (4)	93.9 (9.8)	25.3%	128.8 (13.9)	34.7%
TFT	12 (2)	91.8 (7.6)	24.7%	122.9 (12.8)	33.1%
TFT	24 (4)	92.7 (7.2)	25.0%	123.4 (12.9)	33.2%
AG-LSTM (Ours)	12 (2)	**89.3** (9.9)	**24.1%**	**122.6** (13.5)	**33.0%**
AG-LSTM (Ours)	24 (4)	90.7 (9.3)	24.4%	141.1 (16.0)	38.0%

H = Forecast window
O = Observation window
s-MAE: scaled dMAE
SD: Standard Deviation

with older data negatively impacts their performance, as they update their state at every time-step without considering whether the information is pertinent to the task at hand. An in-depth analysis of the impact of the observation window length on the performance of the proposed model is presented in §5.2.4.

We emphasize that, despite our model achieving the highest performance among those tested, the results exhibit a large standard deviation. The errors from TFT are slightly higher but show a smaller standard deviation in tests using the SDWPF dataset. The two graph-based baselines (DMST and AGCRN) achieve similar performance, a trend observed across both datasets. The Encoder-Decoder LSTM architecture performs the worst among deep learning methods for the Penmanshiel dataset and the second worst for the SDWPF. VARIMA performs the worst in all scenarios, with a significant performance gap. This implies that the linear nature of the model is too simple for the complexity of the wind farm data.

From the above, we conclude that the datasets we are modeling are complex, as indicated by the superior performance of the most intricate models, namely TFT and AG-LSTM. Graph-based modeling seems well-suited to the data, particularly for the SDWPF dataset, given the relatively strong performance of AG-LSTM, AGCRN, and DMST. Given its strong performance, the AG-LSTM Network appears to exploit the diverse components within its architecture. The ablation study in §5.2.3 explores the contribution of each component to the model's accuracy.

Figure 5.7a shows the average dMAE obtained by each model for forecast windows

Table 5.5: Results for the History-Driven Scenario on the test-split of the Penmanshiel dataset. All results are averaged over 5 runs.

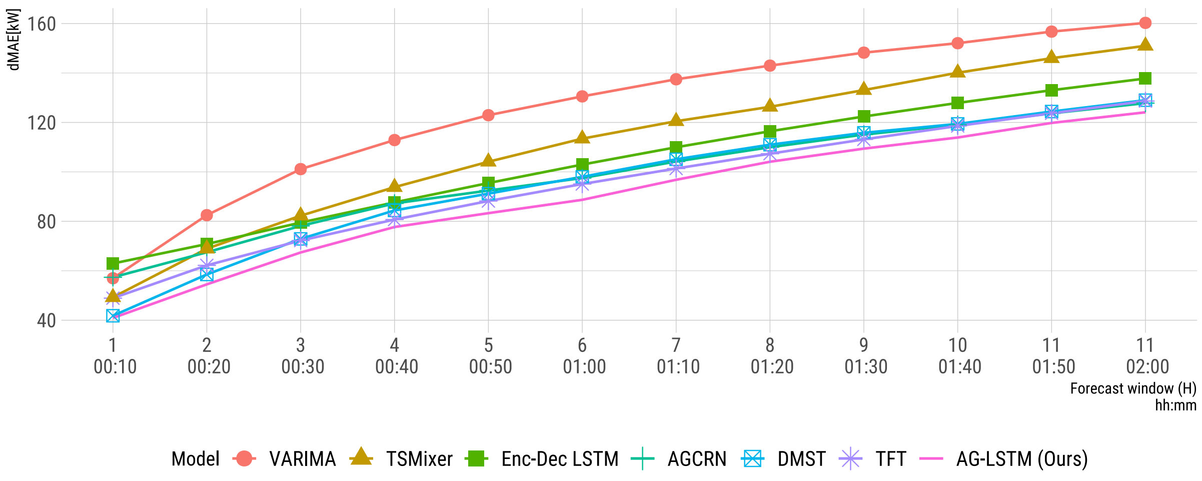
		Penmanshiel			
		H=12 (2 hours)		**H=24 (4 hours)**	
Model	**O (hrs)**	**dMAE** (SD)	**s-dMAE**	**dMAE** (SD)	**s-dMAE**
VARIMA	12 (2)	89.1 (7.2)	20.1%	134.4 (12.1)	30.3%
TSMixer	12 (2)	59.0 (4.8)	13.3%	84.4 (9.1)	19.0%
TSMixer	24 (4)	58.8 (4.8)	13.3%	86.1 (9.0)	19.4%
Enc-Dec LSTM	12 (2)	71.1 (5.1)	16.2%	91.6 (8.9)	20.7%
Enc-Dec LSTM	24 (4)	73.2 (4.8)	16.5%	91.3 (9.2)	20.6%
DMST	12 (2)	64.2 (5.0)	14.5%	88.9 (8.6)	20.1%
DMST	24 (4)	63.1 (5.1)	14.2%	88.6 (8.6)	20.0%
AGCRN	12 (2)	64.9 (4.4)	14.6%	89.2 (8.9)	20.1%
AGCRN	24 (4)	63.8 (4.3)	14.4%	90.4 (9.1)	20.4%
TFT	12 (2)	61.6 (6.2)	13.9%	84.8 (10.1)	19.1%
TFT	24 (4)	62.1 (6.1)	14.0%	84.2 (10.4)	19.0%
AG-LSTM (Ours)	12 (2)	57.9 (4.5)	13.1%	**83.7** (8.2)	**18.9**%
AG-LSTM (Ours)	24 (4)	**57.4** (4.7)	**13.0**%	87.1 (8.8)	19.7%

H = Forecast window
O = Observation window
s-MAE: scaled dMAE
SD: Standard Deviation

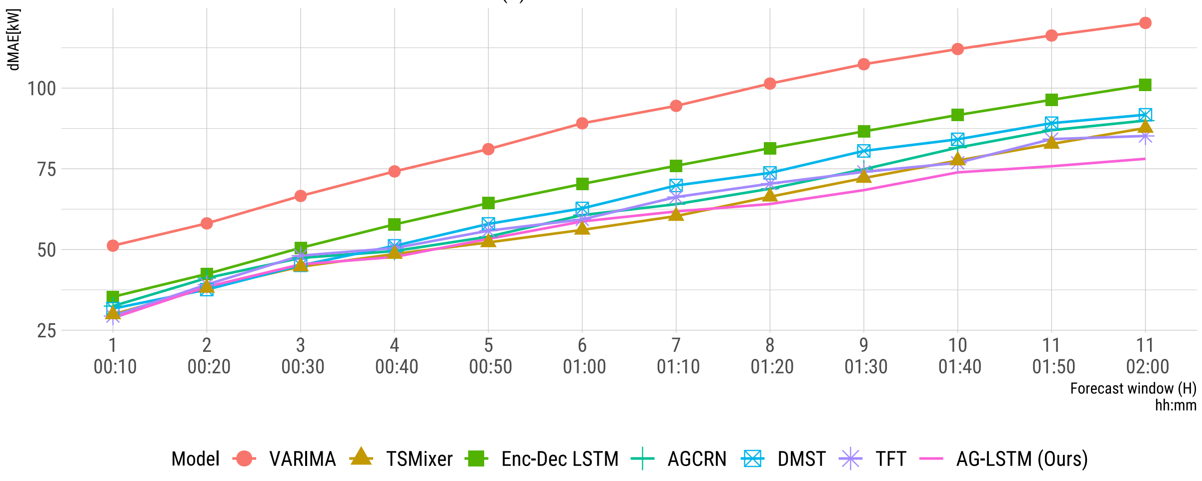
ranging from $H = 1$ (10 minutes) to $H = 12$ (2 hours) for the SDWPF dataset. Our model yields a lower error than the rest across all values of H. The errors follow a similar pattern across all models, and their relative performance remains fairly consistent through the various values of H. VARIMA exhibits a more pronounced logarithmic trend, attaining the worst error across all models for H larger than 1. This again indicates the incompatibility of the method with the power time-series. Figure 5.7a additionally showcases the effectiveness of the encoder-decoder architecture for the SDWPF dataset. Models with this configuration consistently achieve the lowest accuracy across the different values of H. We attribute this to the differentiated components they use to process the input and output, resulting in more nuanced embeddings.

A similar trend is observed for the Penmanshiel dataset. Figure 5.7b shows the performance of the assessed models for increasing values of H. The accuracy of all the deep learning models is very similar for short forecasts, in particular when $H < 4$. As the forecast window increases, the dMAE becomes more disperse. Our model displays the highest overall accuracy, which is evident for values of H larger than 7. For this dataset, our model showcases a superior ability to encode the information available from the provided observations.

The error distribution of our model is explored in Figure 5.8. The plot shows the error for each turbine in the SDWPF dataset. The mean error increases as the forecast horizon lengthens, aligning with the previously discussed point that longer-term forecasts are more challenging due to reduced knowledge about future states. Along with the



(a) SDWPF Dataset



(b) Penmanshiel Dataset

Figure 5.7: Average error per model relative to the length of the forecast window for the History-Driven scenario. Results obtained with $O = 12$ (2 hours).

mean, there is a clear increase in error dispersion and a higher prevalence of outliers as the values of H rise. Notably, the same four turbines frequently account for the upper outliers, highlighting the model's difficulty in adapting to their specific behaviours. A detailed investigation reveals that these turbines are positioned near each other and exhibit a low rate of invalid values, making the processed power series more volatile and challenging for the model to learn.

Future-Known Value Scenario

We now consider the integration of the future covariates into the model. Accounting for these values results in higher accuracy, as it provides them with an approximation of future conditions that show a strong correlation to the power feature. Nowadays weather forecasts containing temperature and wind conditions are available from a wide array of sources with high geographical and temporal resolutions. However, because the exact location and dates for the SDWPF are unknown to us, we substitute the forecasts with known data, adding noise that increases with the forecast horizon, as described in

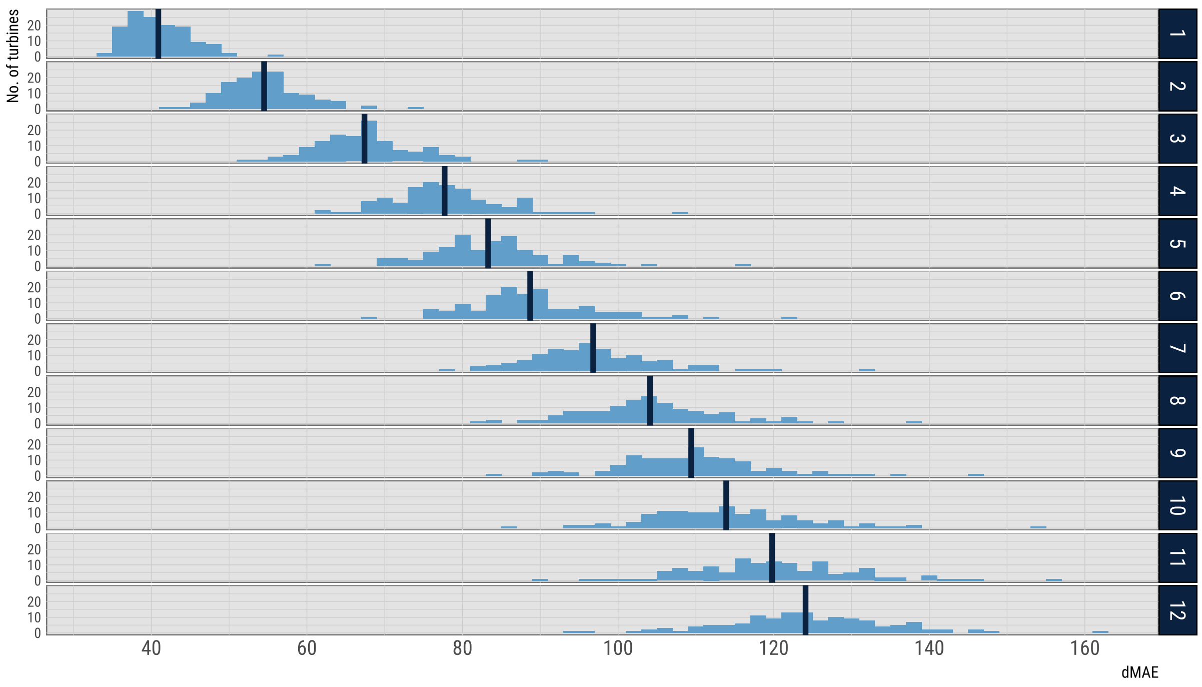


Figure 5.8: Distribution of the average error per turbine of the proposed model, for $H = 1$ (10 minutes) to $H = 12$ (2 hours). Results obtained using SDWPF data. The thick vertical lines correspond to the average across all turbines.

§5.1.4.

Table 5.6 and Table 5.7 show the performance of the baselines and the proposed model for both datasets. TSMixer attains the lowest error across all the models by a large margin for both the 2-hour ahead and 4-hour ahead scenarios. While the proposed model shows a marginal advantage in performance for the shorter window against TFT, it lags behind at $H = 24$ (4 hours).

Figure 5.9a shows the evolution of the dMAE as the forecast window increases for the different models. In the first few time-steps, all models perform similarly. For values of H larger than 3, the accuracy starts diverging. At around $H = 6$, the error of the baselines plateaus, while it continues increasing for our model.

The inclusion of future covariates offers crucial additional information for longer forecasts. Initially, the AG-LSTM Network performs relatively well, thanks to its effective encoding and use of past data. We hypothesize that future covariates add minimal value at this stage. By time $t + 8$, past information becomes less relevant, and future covariates are vital for accurate predictions. Consistent with this notion, the AG-LSTM Network displays a significant limitation in harnessing future covariates; in longer forecast scenarios, it is outperformed by all other models, and unlike them, its error rate continues to rise. Our model integrates the future covariates in the decoder cells, which may lead their values to vanish, contributing marginally to the end result.

TSMixer achieves the lowest error across all forecast windows (Figure 5.9a). At the one-step ahead forecast, all three models display comparable errors. However, as the forecast extends beyond the first step, the variation in accuracy begins to increase, reaching its peak at the 7-step ahead forecasts. The authors of TSMixer attribute its effectiveness to its unweighted method of capturing temporal information, inspired

Table 5.6: Results for the History-Driven Scenario on the test-split of the SDWPF dataset. All results are averaged over 5 runs.

		SDWPF			
		H=12 (2 hours)		**H=24 (4 hours)**	
Model	**O (hrs)**	**dMAE** (SD)	**s-dMAE**	**dMAE** (SD)	**s-dMAE**
TFT	12 (2)	60.6 (7.5)	16.3%	79.4 (11.8)	21.4%
TFT	24 (4)	62.1 (7.5)	16.7%	79.9 (12.0)	21.5%
TSMixer	12 (2)	**54.6** (6.4)	**14.7%**	**74.8** (11.0)	**20.2%**
TSMixer	24 (4)	54.9 (6.7)	14.8%	75.0 (11.1)	20.2%
AG-LSTM (Ours)	12 (2)	60.3 (7.2)	16.2%	79.7 (11.7)	21.5%
AG-LSTM (Ours)	24 (4)	61.1 (6.6)	17.3%	80.7 (12.5)	21.7%

H = Forecast window
O = Observation window
s-MAE: scaled dMAE
SD: Standard Deviation

Table 5.7: Results for the History-Driven Scenario on the test-split of the Penmanshiel dataset. All results are averaged over 5 runs.

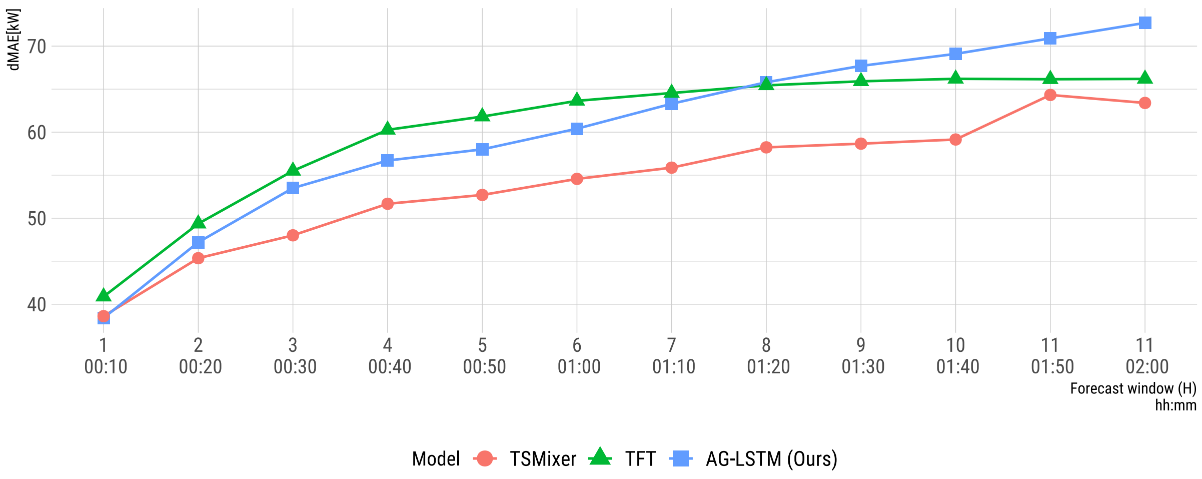
		Penmanshiel			
		H=12 (2 hours)		**H=24 (4 hours)**	
Model	**O (hrs)**	**dMAE** (SD)	**s-dMAE**	**dMAE** (SD)	**s-dMAE**
TFT	12 (2)	47.7 (3.9)	10.8%	76.6 (9.2)	17.3%
TFT	24 (4)	48.1 (4.1)	10.9%	77.2 (9.5)	17.4%
TSMixer	12 (2)	45.8 (3.5)	10.3%	75.6 (8.8)	17.1%
TSMixer	24 (4)	46.1 (3.7)	10.4%	75.9 (8.8)	17.1%
AG-LSTM (Ours)	12 (2)	47.6 (3.4)	10.7%	77.1 (8.5)	17.4%
AG-LSTM (Ours)	24 (4)	48.0 (3.5)	10.8%	78.7 (9.1)	17.8%

H = Forecast window
O = Observation window
s-MAE: scaled dMAE
SD: Standard Deviation

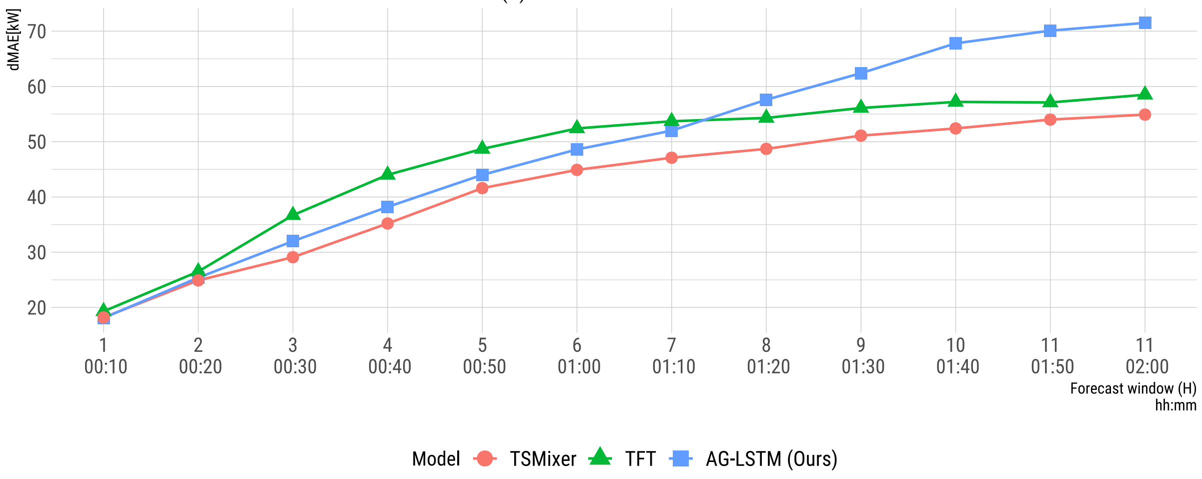
by non-linear approaches. Additionally, we hypothesize that the improvement stems from the align-and-mix technique for integrating future covariates, in contrast to the fragmented approach of the other methods, which combine features at different points and across various components. This fragmentation may cause the information to vanish through the model's weights as it transitions from one component to another.

5.2.1. Forecast Quality

In this section, we analyze the forecast produced by our model, focusing on two specific aspects: i) Comparison of Model Output to Target Values and ii) Input-Output Lag. The former is an assessment of the model's output compared to the true power values. The latter is an assessment to check whether the results returned by the model are just a lagged version of the input. For this section, we focus on the 2-hour ahead forecast of



(a) SDWPF Dataset



(b) Penmanshiel Dataset

Figure 5.9: Mean error per model across different forecast windows for the scenario considering future covariates. Results obtained with $O = 12$ (2 hours).

the test split of the SDWPF dataset, resulting from our model.

Comparison of Model Output to Target Values

We now inspect the forecast produced by the model and compare it to the actual power time-series. Figure 5.10 illustrates a series of forecasts generated by our model and the corresponding true power series. These samples are obtained for an arbitrary turbine, considering the history-driven scenario.

The true series displays the volatility inherent to power and wind time-series, with considerable fluctuations from one time-step to the next. Contrastingly, the model's output consists of a smooth line, with a defined trend and slight curvatures. Similar patterns are observed throughout the resulting forecasts for all turbines. The difference between the forecast and the true values, reveals a shortcoming in the model in fully capturing and projecting the time-series dynamics. This shows that the model learns the general trend of the series but fails to capture the fluctuations between subsequent time-steps.

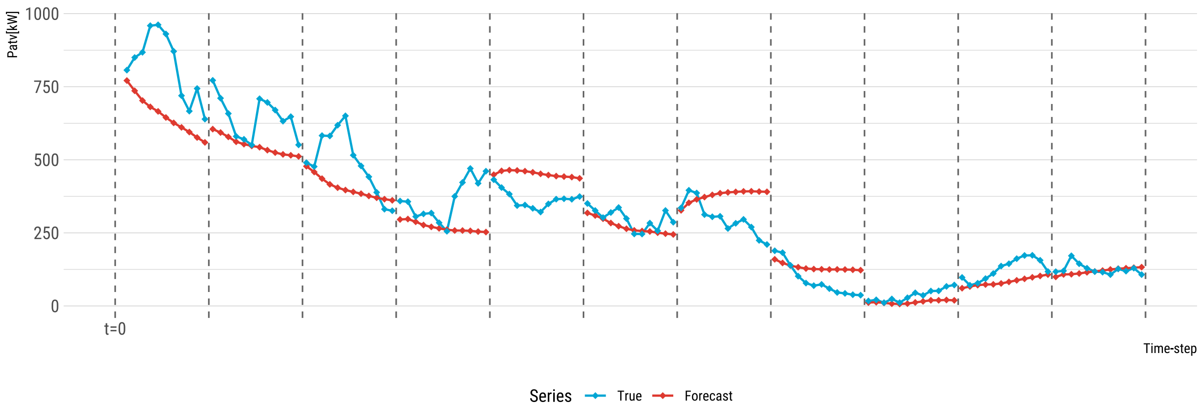


Figure 5.10: Sample forecasts generated by our model in the history-driven scenario. For any forecast, the corresponding input lies in the preceding window.

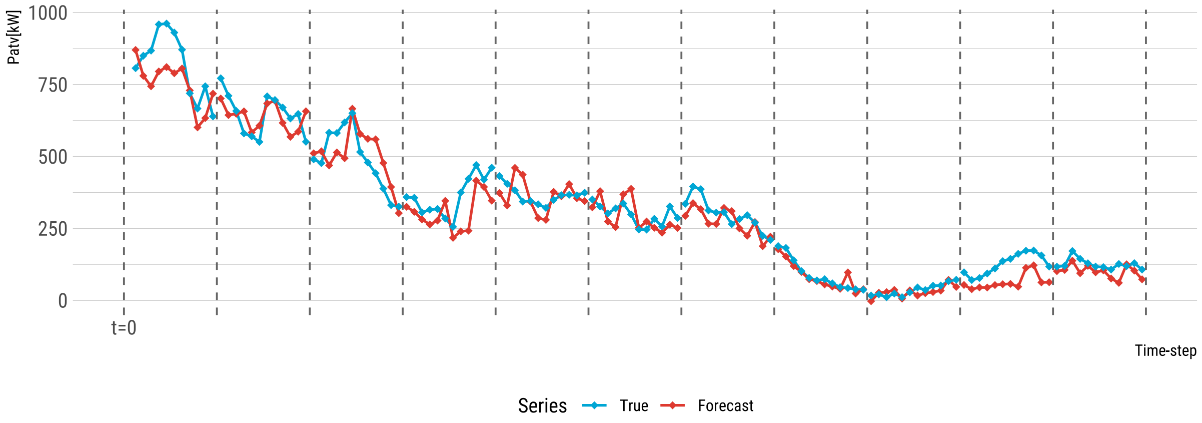


Figure 5.11: Forecast produced by our model, incorporating future covariates. The forecast samples shown are the same as those in Figure 5.10.

As shown in Figure 5.11, the inclusion of future covariates enhances the forecast, causing it to fluctuate more and better reflect the behaviour of the target feature. The positive impact of this addition is evidenced by the enhanced results in Table 5.6 compared to Table 5.4. We conclude that the high volatility is very hard to learn, even by complex models, but aided by considering future information.

Input-Output Lag

Furthermore, we analyze the model forecasts to assess whether they correspond to lagged versions of the input series. Outputting a shifted version of the input sequence can happen in RNNs when the model is not complex enough or the training data does not capture the underlying dynamics of the system. In such cases, the model can utilize its memory cells to store the input sequences. To test this, we compute the dMAE for the entire test dataset, considering a forecast window of size 12 (2 hours) and a step-size of 1. We then collect the last value of each forecast $\hat{y}_{t+12}$ in a new series and compare it to the input sequence. A sample comparison is shown in Figure 5.12. The absence of lag in the output sequence indicates that the model uses its memory units as intended, storing transformations of the input instead of the actual power values. By utilizing

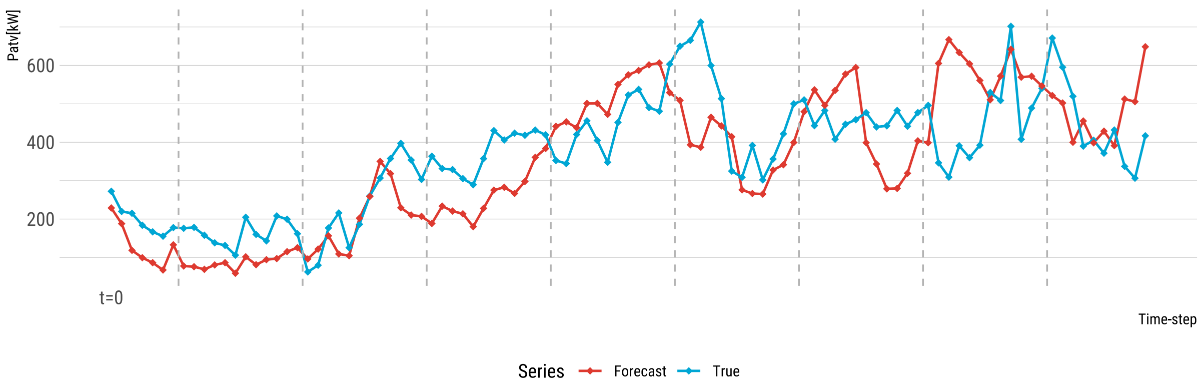


Figure 5.12: Sample lag test, where the power time-series is compared to a series conformed of the last forecast $\hat{y}_{t+12}$ for all the forecast windows in the test set.

historical and future covariates along with static features, the model learns to generate a relatively accurate signal.

5.2.2. Learned Farm Topology

An adjacency matrix serves as the foundation for GNNs, as it holds the graph structure and facilitates the propagation of information across it. The AG-LSTM cell in our model is designed to indirectly learn an adaptive adjacency matrix ($\tilde{\mathbf{A}}_t$) for each time-step, allowing it to adapt to dynamic wind farm characteristics and capture relationships between turbines that might not be apparent. By examining the contents of $\tilde{\mathbf{A}}_t$, we gain insights into these relationships, enhancing our understanding of the wind farm. Each row represents a vector, containing the weight of the relationship of a particular turbine to all the other turbines.

Some samples of $\tilde{\mathbf{A}}_t$ generated by a trained version of the AG-LSTM Network are shown in Figure 5.13, obtained over random values of t from the test split of the SDWPF dataset. We observe large values across the diagonal of $\tilde{\mathbf{A}}_t$, indicating self-loops in the graph structure, which means turbines rely on their own features and cell states for forecasting power production. In some cases (top-right and mid-bottom of Figure 5.13), the matrices are very sparse, showing turbines with few or no neighbors beyond themselves. In other samples, turbines have many neighbors, incorporating information from multiple turbines in addition to their own for forecasting.

A pattern can be observed across the matrices in Figure 5.13 with some columns displaying consistent values across their entries. A systematic assessment of this phenomenon consists of inspecting the rank of the corresponding adjacency matrices. The rank is an indicator of various graph topology aspects. Figure 5.14 shows the distribution of the ranks for all iterations of $\tilde{\mathbf{A}}_t$ for the SDWPF dataset. The ranks concentrate around 19, which is low compared to the 134 nodes in the graph, indicating few linearly independent structures and many turbines sharing similar interaction patterns. The low rank also reveals the frequent existence of isolated turbines, visible as dark vertical lines in Figure 5.13. Conversely, bright lines correspond to high values for certain turbines that consistently act as neighbors for the rest of the wind farm.

We theorize that the turbines displaying high values across a single column in $\tilde{\mathbf{A}}_t$

result fundamental for the model at that t, as they convey information that all the other turbines exploit towards their own forecast. Pinpointing the reason for their immediate prominence results challenging given the complexity of the information, but we hypothesize these turbines display a trend before the others, due to the environmental conditions. These turbines could be located in the limits of the wind farm, where they are the first to experience changes in wind patterns, thus becoming early indicators of wind regimes that affect the entire farm.

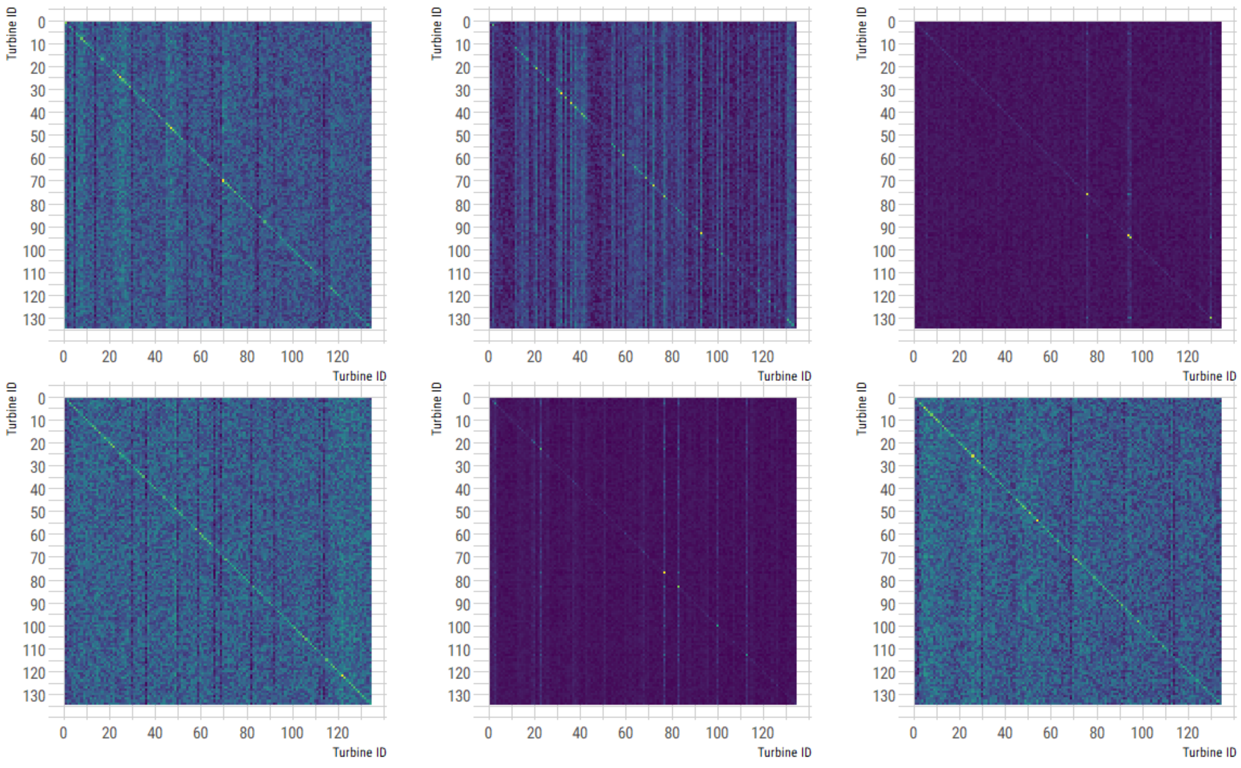


Figure 5.13: Samples of $\mathbf{A}_t$ obtained through the test split of the SDWPF dataset. Scale goes from purple (small values) to yellow (large values).

We now analyze $\tilde{\mathbf{A}}$, computed as:

$$\tilde{\mathbf{A}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{A}_t \tag{5.5}$$

where T represents the total number time-steps in a dataset. $\tilde{\mathbf{A}}$ corresponding to the test split of the SWDPF dataset is depicted in Figure 5.15a. Most values in the matrix are close to 0, reflecting mostly weak relationships among the turbines. The diagonal is again evident, reflecting self-loops in the graph structure that repeatedly appear through time. This is favorable, evidencing the constant use of a turbine's features towards its own power forecast.

The column pattern again emerges, this time over a few turbines. In Figure 5.15b, this data is overlayed on the wind farm, where turbines are colored based on their average column weights. Notably, four turbines display large averages, reflecting their deep interconnectivity and pivotal role within the dynamics of the wind farm across time-steps. Three of the highlighted turbines are in close proximity, suggesting that their position relates to their importance.

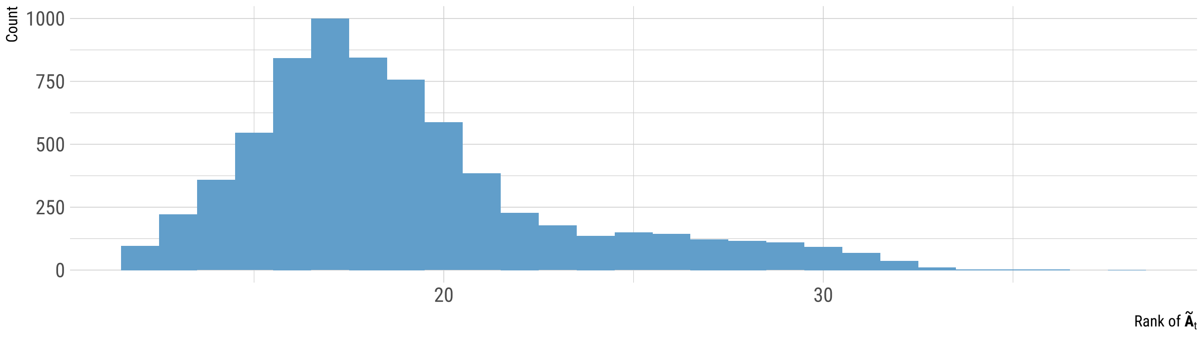
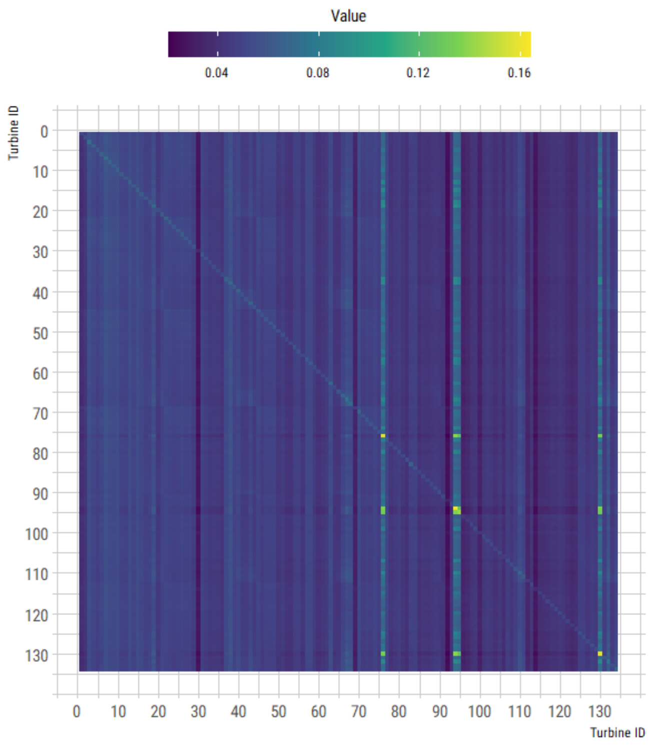
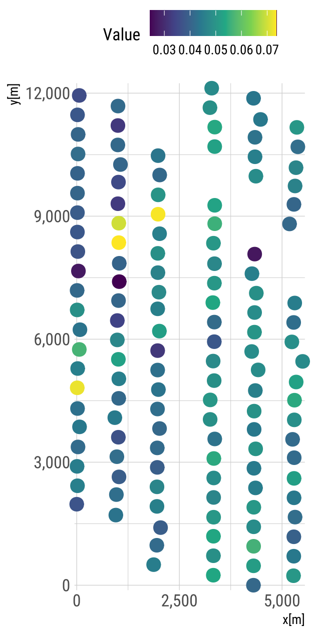


Figure 5.14: Distribution of the rank of the $\mathbf{A}_t$ resulting from every input in the test dataset.



(a) Matrix obtained by averaging the adjacency matrices that result from running the test dataset through the trained model.



(b) Centrality of the turbines in the wind farm, computed as the column average of a).

Figure 5.15: Visualization of the adaptive adjacency matrix obtained for the BDD Dataset.

Given the intricate nature of the dataset and the transformations applied by the model, understanding the central role of these turbines poses a challenge. We theorize that they may encounter unique conditions earlier than others, possibly due to their exposure to higher-altitude wind currents or being the first to be shut down according to the farm's operational rules. More thorough analysis and greater insight into the specifics of the wind farm are necessary to better understand its dynamics.

5.2.3. Ablation Study

To quantify the impact of each individual component on the overall performance of the model, we conducted an ablation study. The results, obtained using the SDWPF dataset, are presented in Table 5.8. The first row corresponds to the baseline model configuration, which integrates all the necessary components for optimal performance. Subsequent rows describe variations of the model, where the model's configuration, the adjacency matrix, and the input features are removed or modified to assess their individual impact on the error metrics. We consider the scenario with observation and forecast windows of length 12.

As observed in Table 5.8. the most critical feature of the AG-LSTM Network is the integration of future covariates (wind speed and direction). This aligns with the basic concepts of wind turbines (§2.1), given the strong physical relationship and the high correlation observed between these features and the power output. Fortunately, incorporating these covariates only slightly increases the number of trainable parameters by 0.6%, specifically within the decoder cell's internal parameters.

Integrating historical features (wind speed, wind direction, blade orientation, and internal temperature) into the encoder reduces the dMAE, while having minimal impact on the number of parameters. This reduction in error demonstrates the effectiveness of accounting for environmental conditions for each feature and synthesizing them through the AG-LSTM cell to model interactions. On the other hand, removing the static features (turbine position) affects the model's accuracy by 0.3kW, reflecting the inability of the model's decoder to learn from them.

The use of two pre-defined (static) adjacency matrices in the AG-LSTM Network was tested to assess the effect of the adaptive variation. In both cases, a self-loop was added. The first matrix contains the inter-turbine correlation of the target power series. We experimentally selected a threshold of 0.8, under which the entries were set to 0. This matrix configuration increased the dMAE by 12.1 kW, suggesting that while two turbines might exhibit similar general trends, their specific environmental conditions at a given time may not consistently aid in short-term forecasting.

The second static matrix considers a Gaussian kernel applied to the Euclidean distance between turbines, limited to those located within a range of 1,500 meters. This approach achieved a dMAE very close to that of the optimal configuration, with an increase of just +1.2 kW. We believe this is because neighbouring turbines provide information reflecting future short-term environmental conditions, such as wind patterns. Both cases reduce the number of learnable parameters by 24.8% associated with the input embeddings used to build $\tilde{\mathbf{A}}_{\mathbf{t}}$. The minor difference in dMAE between the distance-based and adaptive configurations highlights the effectiveness of both methods and suggests that turbine distance is relevant for the spread of information across the graph.

Finally, we replaced the decoder in our model with an MLP that considers future covariates, the encoder's output, and static variables for each future timestep to generate the forecasted output. This version without decoder cells achieves a dMAE of 65.4, using just over one-fourth of the parameters. The accuracy gap compared to the encoder-decoder versions of the AG-LSTM Network underscores the importance of both modules. This resonates with the results from Table 5.4 and Table 5.5, reiterating the necessity of the encoder-decoder configuration to transform and evolve the information from the input to the output.

Table 5.8: Settings for the proposed model and their corresponding parameter count and results. The reported results are obtained through the SDWPF dataset, using $O = 12$ and $H = 12$.

	Settings							
Configuration	$\tilde{\mathbf{A}}$	**X**	**Z**	$\mathbf{e}_s$	**Number of parameters**	**dMAE**	**SD**	**s-dMAE**
E-D	Adaptive	Y	Y	Y	62,153	60.3	7.2	16.2%
E-D	Adaptive	Y	N	Y	61,769	89.3	9.9	24.1%
E-D	Adaptive	N	Y	Y	61,825	67.7	11.2	18.2%
E-D	Adaptive	Y	Y	N	61,769	60.6	7.5	16.3%
E-D	Correlation-based	Y	Y	Y	46,716	72.4	11.2	19.5%
E-D	Distance-based	Y	Y	Y	46,716	61.5	6.2	16.8%
E	Adaptive	Y	Y	Y	46,909	65.4	7.1	17.6%

Configuration: E-D: encoder-decoder, E: encoder-only
$\tilde{\mathbf{A}}$**:** adjacency matrix
X: historical features (other than power)
Z: future covariates
$\mathbf{e}_S$**:** static features

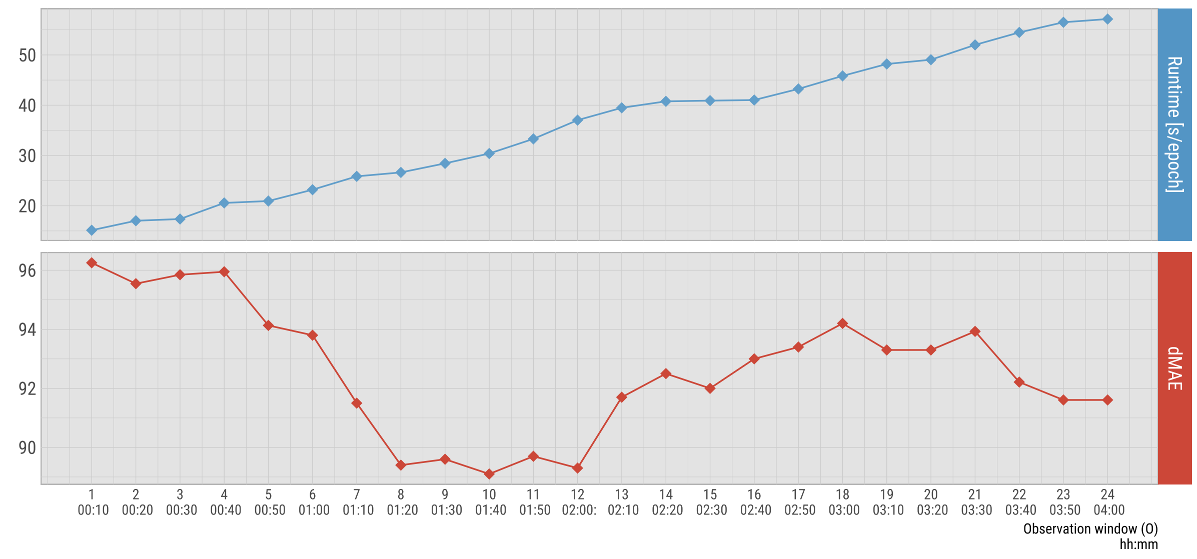


Figure 5.16: Execution time (top) and error (bottom) for observation windows ranging from 1 (10 minutes) to 24 (4 hours). The forecasts are made for $H = 12$ (2 hours ahead), using the SDWPF dataset.

5.2.4. Sensitivity Analysis

We investigate the effects of the length of the observation window on our model's performance when trained under the History-Driven setting for 12-step (2-hour) forecasts. We focus on the History-Driven setting, forecasting for 12 steps (2 hours) ahead with the SDWPF dataset. We log the dMAE and average time per epoch for model training with O ranging from 1 (10 minutes) up to 24 (4 hours). The experimental results are detailed in Figure 5.16.

Training time for the model increases linearly with the expansion of the observation window. Initially, at $O = 1$, the average training time per epoch is 15 seconds, which lengthens to 57 seconds at $O = 24$. The runtime for each epoch increases by roughly

1.8 seconds for every additional time-step included.

The error attained by the model exhibits a more complex behaviour. The highest dMAE, of 96.3 kW, occurs when training with just a single timestep. As more timesteps are included, the error generally decreases, with the lowest dMAE of 90 kW observed when $O = 10$. The error remains consistently low, around 90 kW, for O values between 8 and 12, suggesting that the most critical information for predicting the wind power time-series lies within these 12 timesteps, as supported by the ACF plot in Appendix A. However, as O exceeds 12 (2 hours), the error not only increases but also becomes less stable, indicating a decline in model performance with the inclusion of too much historical data.

Considering the above, we conclude that the optimal model results at $O = 10$, or 100 minutes. Considering our experimental setup, training this model for 150 epochs takes a maximum of 76 minutes and yields the lowest error overall.

5.3 | Discussion

Throughout this section, we have detailed a series of experiments to assess the performance of our model. We now present several observations made during these experiments, interpret their results, and discuss their direct implications within the context of WPF.

The datasets used for evaluation, as described in §5.1.1, were selected because they include multiple variables at the turbine level over long time periods, which is crucial for the recurrent and graph-based formulation of our model. However, there are some limitations associated with them, which we outline below:

- Firstly, the SDWPF dataset includes a significant proportion of invalid values, which we have interpolated during preprocessing because our model requires continuous data. Training the model on this interpolated data can negatively affect its performance, as it may learn the interpolation patterns instead of the actual, volatile power series. Future research could concentrate on evaluating the effects of missing values and imputation methods on our model, as well as on enhancing its robustness to invalid entries.
- Secondly, certain characteristics of both datasets, such as turbine position, day of the year, wind direction, and nacelle orientation, are presented as relative values. The absence of contextual information limits their analysis and exploitation and prevents the use of external forecast data as future covariates. As a workaround, we added noise to these variables to simulate a forecast, which yields somewhat unrealistic results. This approach should be taken into account when reviewing the outcomes, as the results might differ significantly in a real-life scenario. In more realistic settings, the quality and granularity of the forecast will undoubtedly play a crucial role in the accuracy of the predictions.
- Lastly, the SDWPF dataset only spans 245 days. Extending the dataset to include information for more than one year could allow us to incorporate seasonality into the model through features such as yearly progress, potentially enhancing its performance. However, it is important to note that our experiments with time

as an additional future covariate had a negative impact on model performance. This suggests that not all temporal features may yield beneficial results and their integration requires careful consideration.

Throughout the experiments, we observed a recurring pattern: the models generally perform better when provided with observation windows close to 10 time-steps. This is true for both the baseline models, as shown in Table Table 5.4, and for the proposed model, as detailed in §5.2.4. This finding aligns with the autocorrelation function of the wind power time-series, as described in Appendix A. It suggests that for optimal performance in short-term WPF, models require limited amounts of information, leading to shorter training and inference times.

Another observed trend across the models is the increase in dMAE and the spread of its distribution for larger forecasting windows. This phenomenon is well-documented in the literature and is not unique to our setting or models. As we examine the distribution of the error for our model (see §5.2), we notice that specific turbines tend to exhibit higher error rates. However, as our model approaches the wind farm as an integrated system, it processes the information from all turbines simultaneously, propagating information even among the atypical ones.

In §5.2.1, we qualitatively evaluate the forecasts generated by the AG-LSTM Network. While the result trends vary across turbines, and the model is not merely producing a lagged version of the input, we note that the output signal is relatively smooth. This smoothness indicates a limitation of the model in capturing the inherent volatility of the dataset. Although the exact cause of this behaviour is unclear, we hypothesize that it is related to the linear interpolation of missing values in the input data. We observe that integrating future covariates into the model mitigates this issue, reducing the uncertainty that the model exhibits about the future power values.

In §5.2.3, we observe that the AG-LSTM Network that uses dynamically generated adjacency matrices only slightly outperforms the one utilizing distance-based heuristics. This establishes both graphs as effective towards the end of our model, each with its own benefits. The good performance of the AG-LSTM cell suggests that our model effectively learns to map interactions within the wind farm for specific time-steps. The resulting adjacency matrices help map the relationships across turbines as time progresses, and help identify isolated turbines. The average of the matrices helps pinpointing key turbines throughout the wind farm. With additional domain knowledge, these insights could be used to optimize the wind farm operations. However, the embeddings generated by the model lack a direct physical representation, which poses a challenge.

§5.2.3 corroborates our design decisions, establishing the future covariates, the encoder-decoder configuration, and the integration of the GNN's key components towards its satisfactory performance. We finish the model's analysis with §5.2.4, where we determine the optimal trade-off between accuracy and time happens when the observation window equals 10 time-steps.

Overall, we conclude that the proposed method is effective for short-term WPF, achieving the best results among the selected baselines in the history-driven scenario. This success is attributed to the model's capability to learn temporal patterns in the input data and dynamically combine features across turbines. The main opportunity for improvement lies in the exploitation of future covariates, an area where other models have shown success.

6 Conclusion & Further Work

This chapter concludes the work carried out for this thesis. §6.1 provides an answer to the research question and an overview of the previous chapters, highlighting the insights obtained from them. §6.2 proposes directions for future research on using GNNs for WPF.

6.1 | Conclusion

This thesis focuses on the development and evaluation of the AG-LSTM Network, a GNN-enabled model for providing multi-step forecasts for wind power generation in wind farm turbines.

Chapter 1 introduces the context of the WPF task and motivates the need for developing effective, accurate, and fast models for this purpose. It starts by outlining the current state of wind power and arguing that accurate and reliable wind farm forecasts will become pivotal in the future energy landscape within a few years. This chapter posits the hypotheses that graphs are suitable for encoding the complex interactions between wind turbines and that GNNs can leverage this structure for accurate WPF. Finally, these hypotheses are translated into the research question:

RQ) How can GNNs efficiently utilize graph structures to capture turbine interactions in wind farms to generate accurate short-term wind power forecasts?

Chapter 2 presents the concepts on which the rest of the thesis builds. It provides an overview of wind turbines, explaining how and why they are arranged in wind farms. It introduces graphs and explains how GNNs operate on them. Furthermore, it introduces RNNs and details the AGCRN method, which forms the basis of the developed method.

A review of existing methods for WPF is presented in Chapter 3. A classification framework is introduced, and different methods within each category are detailed. Particular emphasis is placed on GNN-based methods, reviewing how these translate wind systems into the graph domain and how they leverage other types of networks with GNNs to model spatio-temporal relationships. It is noted that, although multiple models exist for WPF, none of them leverage various features to provide turbine-level, multi-step forecasts.

Chapter 4 builds upon the previous content and answers the research question by introducing and detailing the proposed AG-LSTM Network. An encoder-decoder architecture is considered as it can capture long-term dependencies and provides the flexibility to handle input and output sequences of different lengths. The encoder comprises a novel LSTM cell, which utilizes GNNs to combine the information from adjacent turbines. Notably, the adjacency matrix used by the model is dynamic, as it depends on the input information at every iteration of the model. During training, the cell learns the mapping of the input features, which is ultimately transformed into the adjacency matrix, reflecting the model's ability to learn and model the interactions among turbines given evolving conditions. The decoder of the proposed network integrates future covariates and static features, providing advantageous information for the forecast.

Finally, Chapter 5 presents an evaluation of the proposed method using the SDWPF dataset. By comparing the dMAE for 2-hour and 4-hour ahead forecasts obtained by our model to those from other state-of-the-art models, we demonstrate its superiority when future covariates are not considered. The AG-LSTM Network outperforms models with and without graph convolutions and even those integrating attention mechanisms. However, in scenarios that account for future-known variables, the model, while no longer the best, remains comparable to higher-parameter models.

In the same chapter, we provide a detailed output signal analysis. We conclude that our model does not merely lag or retain the input signal but instead learns the volatile behaviour of the time-series. Furthermore, we analyze the impact of future covariates on the output signal, noting that they transform it from a smooth trend line into a series that more closely resembles a power time-series.

Chapter 5 also includes an ablation study of the model, identifying the components and parameters responsible for its accuracy. We conclude that the integration of future covariates has the most significant impact, given the high correlation between wind speed and wind power. The study highlights the model's ability to encode past information, as evidenced by the significant drop in performance when historical features are ignored. This is further supported by the low error achieved by the encoder-decoder architecture compared to an encoder-only version. Using an adaptive adjacency matrix marginally reduces the model's error, indicating that while the model learns to map turbine interactions, distance-based adjacency matrices should not be disregarded as they do not require learning any parameters.

The experiments conducted throughout the thesis offer insights not only into the tested models but also into the dynamics of wind farms and the power series of turbines. We repeatedly observed that information from the last 10 time-steps is crucial for predicting future power production. Including data beyond this horizon can negatively impact short-term forecasts. Additionally, we found that the data from certain key turbines is utilized by most others at different time-steps, indicating their critical role in the wind farm. Therefore, ensuring the proper functioning of these key turbines should be prioritized to maximize the accuracy and reliability of WPF using models that consider turbine interactions.

The insights derived from this work, encompassing the problem setting, SCADA datasets, baseline models, and the proposed AG-LSTM architecture, will undoubtedly contribute to more accurate short-term WPF methods. These advancements will enable the stable and reliable operation of wind farms, which is crucial for seamlessly integrating wind power into the energy grid. As a result, this research will significantly propel the role of wind power in the emerging energy landscape, paving the way for a future where the energy needs are fulfilled through renewable sources. This thesis lays a solid foundation for continued innovation and progress in the field of WPF.

6.2 | Future Work

We conclude this work by providing directions for future research on using GNNs for WPF in wind farms.

Robustness to Missing Data

A significant problem with the SDWPF dataset is the large proportion of missing or invalid data from the SCADA system. In particular, 29.8% of the active power values were classified as anomalies and replaced through linear interpolation along the time dimension. This step was taken because the proposed model requires complete input data and temporal interpolation was chosen due to the simultaneous appearance of missing values across much of the wind farm. Two potential strategies can be explored.

The first is to adapt the proposed model to manage missing information by excluding nodes with missing values or learning to impute these values during training. The second is to investigate the effect of different interpolation and imputation techniques on the forecast accuracy of GNN-based models.

Future Covariates

Although the AG-LSTM Network achieves the lowest error using only historical data, it is outperformed when future covariates are considered. This highlights an opportunity to improve the integration of wind power and direction forecasts. While few existing WPF models integrate such features, current methods used for WPP can be explored to this end. One possible approach involves using a surrogate model to approximate the power curve, with parameters optimized based on each turbine's capacity. To our knowledge, no existing works take on this approach for WPF.

Improved Topology Learning

The proposed model comprises a data-driven approach to dynamically determine the adjacency matrix $\mathbf{A}_t$ for each iteration. This results in a matrix that resembles the interaction among the turbines. However, due to its dynamic nature, $\mathbf{A}_t$ results hard to analyze and interpret. Specifically, there is no clear way to understand the transformation that the input features undergo and their link to the farm's dynamics. The weight analysis in §5.2.2 shows that some critical turbines are used by others to generate power forecasts but does not identify what makes these turbines central. Further analysis of the wind farm's behaviour could help to interpret these results more effectively.

The ablation study in §5.2.3 shows a slight improvement in our model's performance when using the adaptive matrix compared to a constant, predefined $\mathbf{A}$. Even with a distance-based $\mathbf{A}$, the model still performs comparably to the baseline models. Future research could explore combining these matrices, and incorporating additional learnable parameters to weight them. A third matrix could also be considered based on physics-based heuristics, such as wind speed, direction, and turbine orientation.

References

[1] URL: https://www.nrel.gov/wind/floris.html.

[2] Hüseyin Akçay and Tansu Filik. "Short-term wind speed forecasting by spectral analysis from long-term observations with missing values". In: *Applied Energy* (2017). DOI: 10.1016/j.apenergy.2017.01.063. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85012000390&doi=10.1016%2fj.apenergy.2017.01.063&partnerID=40&md5=21d01a36288bcf02365ac212a7775bd0.

[3] S. Anagnostopoulos and M. D. Piggott. "Offshore wind farm wake modelling using deep feed forward neural networks for active yaw control and layout optimisation". In: vol. 2151. IOP Publishing Ltd, Jan. 2022. DOI: 10.1088/1742-6596/2151/1/012011.

[4] J. Scott Armstrong. "Standards and Practices for Forecasting". In: *Principles of Forecasting: A Handbook for Researchers and Practitioners*. Ed. by J. Scott Armstrong. Boston, MA: Springer US, 2001, pp. 679–732. ISBN: 978-0-306-47630-3. DOI: 10.1007/978-0-306-47630-3_31. URL: https://doi.org/10.1007/978-0-306-47630-3_31.

[5] European Wind Energy Association. *Wind Energy - The Facts: A Guide to the Technology, Economics and Future of Wind Power*. Earthscan, 2012. ISBN: 9781849773782. URL: https://books.google.nl/books?id=W0dre8-5FAYC.

[6] F. Azlan et al. "Review on optimisation methods of wind farm array under three classical wind condition problems". In: *Renewable and Sustainable Energy Reviews* 135 (2021), p. 110047. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2020.110047. URL: https://www.sciencedirect.com/science/article/pii/S1364032120303385.

[7] Lei Bai et al. *Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting*. 2020. arXiv: 2007.02842 [cs.LG].

[8] C. J. Bay et al. "Unlocking the Full Potential of Wake Steering: Implementation and Assessment of a Controls-Oriented Model". In: *Wind Energy Science Discussions* 2019 (2019), pp. 1–20. DOI: 10.5194/wes-2019-19. URL: https://wes.copernicus.org/preprints/wes-2019-19/.

[9] Y. Bengio, P. Simard, and P. Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166. DOI: 10.1109/72.279181.

[10] Lars Ødegaard Bentsen et al. "Probabilistic Wind Park Power Prediction using Bayesian Deep Learning and Generative Adversarial Networks". In: *Journal of physics* (2022). DOI: 10.1088/1742-6596/2362/1/012005.

[11] Lars Ødegaard Bentsen et al. "Spatio-temporal wind speed forecasting using graph networks and novel Transformer architectures". In: *Applied Energy* 333 (Mar. 2023). ISSN: 03062619. DOI: 10.1016/j.apenergy.2022.120565.

[12] Lars Ødegaard Bentsen et al. "Wind Park Power Prediction: Attention-Based Graph Networks and Deep Learning to Capture Wake Losses". In: *CoRR* abs/2201.03229 (2022). arXiv: 2201.03229. URL: https://arxiv.org/abs/2201.03229.

[13] Kanna Bhaskar and S. N. Singh. "AWNN-Assisted Wind Power Forecasting Using Feed-Forward Neural Network". In: *IEEE Transactions on Sustainable Energy* 3.2 (2012), pp. 306–315. DOI: 10.1109/TSTE.2011.2182215.

[14] Erasmo Cadenas and Wilfrido Rivera. "Short term wind speed forecasting in La Venta, Oaxaca, México, using artificial neural networks". In: *Renewable Energy* 34.1 (2009), pp. 274–278. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2008.03.014. URL: https://www.sciencedirect.com/science/article/pii/S0960148108001171.

[15] J.P.S. Catalão, H.M.I. Pousinho, and V.M.F. Mendes. "Short-term wind power forecasting in Portugal by neural networks and wavelet transform". In: *Renewable Energy* 36.4 (2011), pp. 1245–1251. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2010.09.016. URL: https://www.sciencedirect.com/science/article/pii/S0960148110004477.

[16] Wen-Yeau Chang. "A Literature Review of Wind Forecasting Methods". In: *Journal of Power and Energy Engineering* 02 (Jan. 2014), pp. 161–168. DOI: 10.4236/jpee.2014.24023.

[17] Si-An Chen et al. *TSMixer: An All-MLP Architecture for Time Series Forecasting.* 2023. arXiv: 2303.06053 [cs.LG].

[18] REN21 DATA and KNOWLEDGE TEAM. *REN21. 2023. Renewables 2023 Global Status Report Collection.* English. 2023.

[19] Gabriel Duarte et al. "How do loss functions impact the performance of graph neural networks?" In: Joinville, SC: SBIC, 2021. DOI: 10.21528/CBIC2021-161.

[20] Gregory Duthé et al. "Local flow and loads estimation on wake-affected wind turbines using graph neural networks and PyWake". In: *Journal of Physics: Conference Series* 2505.1 (2023), p. 012014. DOI: 10.1088/1742-6596/2505/1/012014. URL: https://dx.doi.org/10.1088/1742-6596/2505/1/012014.

[21] Ergin Erdem and Jing Shi. "ARMA based approaches for forecasting the tuple of wind speed and direction". In: *Applied Energy* 88.4 (2011), pp. 1405–1414. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2010.10.031. URL: https://www.sciencedirect.com/science/article/pii/S0306261910004332.

[22] PA Fleming et al. "Evaluating techniques for redirecting turbine wakes using SOWFA". English. In: *Renewable Energy* 70.October (2014). harvest Available online 12-03-2014, pp. 211–218. ISSN: 0960-1481. DOI: 10.1016/j.renene.2014.02.015.

[23] Paul Fleming et al. “Simulation comparison of wake mitigation control strategies for a two-turbine case”. In: *Wind Energy* 18 (Oct. 2014). DOI: 10.1002/we.1810.

[24] Ulrich Focken, Matthias Lange, and Hans-Peter (Igor) Waldl. “Previento - A Wind Power Prediction System with an Innovative Upscaling Algorithm”. In: (Jan. 2001).

[25] Sherry Garg and Rajalakshmi Krishnamurthi. “A survey of long short term memory and its associated models in sustainable wind energy predictive analytics”. In: *Artificial Intelligence Review* 56 (2023), pp. 1149–1198. URL: https://api.semanticscholar.org/CorpusID:260042073.

[26] Pieter Gebraad et al. “Maximization of the annual energy production of wind power plants by optimization of layout and yaw-based wake control”. In: *Wind Energy* 20.1 (2017), pp. 97–107. DOI: https://doi-org.tudelft.idm.oclc.org/10.1002/we.1993. eprint: https://onlinelibrary-wiley-com.tudelft.idm.oclc.org/doi/pdf/10.1002/we.1993. URL: https://onlinelibrary-wiley-com.tudelft.idm.oclc.org/doi/abs/10.1002/we.1993.

[27] Mingju Gong et al. “Short-term wind power forecasting model based on temporal convolutional network and informer”. In: *Energy* (2023). DOI: 10.1016/j.energy.2023.129171.

[28] J.M. González-Sopeña, V. Pakrashi, and B. Ghosh. “An overview of performance evaluation metrics for short-term statistical wind power forecasting”. In: *Renewable and Sustainable Energy Reviews* 138 (2021), p. 110515. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2020.110515. URL: https://www.sciencedirect.com/science/article/pii/S1364032120308005.

[29] Richard M. Gorman. “Intercomparison of Methods for the Temporal Interpolation of Synoptic Wind Fields”. In: *Journal of Atmospheric and Oceanic Technology* 26.4 (2009), pp. 828–837. DOI: 10.1175/2008JTECHO588.1. URL: https://journals.ametsoc.org/view/journals/atot/26/4/2008jtecho588_1.xml.

[30] Junhua Gu et al. “Dynamic Correlation Adjacency-Matrix-Based Graph Neural Networks for Traffic Flow Prediction”. In: *Sensors* 23.6 (2023). ISSN: 1424-8220. DOI: 10.3390/s23062897. URL: https://www.mdpi.com/1424-8220/23/6/2897.

[31] Yuqin He et al. “A robust spatio-temporal prediction approach for wind power generation based on spectral temporal graph neural network”. In: *IET Renewable Power Generation* 16 (12 Sept. 2022), pp. 2556–2565. ISSN: 17521424. DOI: 10.1049/rpg2.12449.

[32] José F. Herbert-Acero et al. “A Review of Methodological Approaches for the Design and Optimization of Wind Farms”. In: *Energies* 7.11 (2014), pp. 6930–7016. ISSN: 1996-1073. DOI: 10.3390/en7116930. URL: https://www.mdpi.com/1996-1073/7/11/6930.

[33] Rockey V Abram Hester, Matthew Travers, and Sebastian Scherer Julian Whitman. *Wind Farm Yaw Optimization and Real-Time Control using Graph Neural Networks*. 2021.

[34] Bri-Mathias Hodge et al. “Improved Wind Power Forecasting with ARIMA Models”. In: *Computer Aided Chemical Engineering* (2011). DOI: 10.1016/B978-0-444-54298-4.50136-7. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-79958804828&doi=10.1016%2fB978-0-444-54298-4.50136-7&partnerID=40&md5=fc81d10b009ca675ce972b166687b274.

[35] D. van der Hoek et al. “Experimental analysis of the effect of dynamic induction control on a wind turbine wake”. In: *Wind Energy Science* 7.3 (2022), pp. 1305–1320. DOI: 10.5194/wes-7-1305-2022. URL: https://wes.copernicus.org/articles/7/1305/2022/.

[36] L.C. Hollaway. “19 - Sustainable energy production: key material requirements”. In: *Advanced Fibre-Reinforced Polymer (FRP) Composites for Structural Applications*. Ed. by Jiping Bai. Woodhead Publishing Series in Civil and Structural Engineering. Woodhead Publishing, 2013, pp. 705–736. ISBN: 978-0-85709-418-6. DOI: https://doi.org/10.1533/9780857098641.4.705. URL: https://www.sciencedirect.com/science/article/pii/B9780857094186500193.

[37] Ying-Yi Hong, Ying-Yi Hong, and Christian Lian Paulo P. Rioflorido. “A hybrid deep learning-based neural network for 24-h ahead wind power forecasting”. In: *Applied Energy* (2019). DOI: 10.1016/j.apenergy.2019.05.044.

[38] Rob J Hyndman and Yeasmin Khandakar. “Automatic time series forecasting: the forecast package for R”. In: *Journal of Statistical Software* 26.3 (2008), pp. 1–22. DOI: 10.18637/jss.v027.i03.

[39] IEA. “World Energy Outlook 2023”. In: (2023). URL: https://www.iea.org/reports/world-energy-outlook-2023.

[40] Elvin Isufi. *Lecture Presentations for Machine Learning for Graph Data*. Feb. 2023.

[41] Elvin Isufi et al. *Graph Filters for Processing and Learning from Network Data*. 2021.

[42] N.O. Jensen. *A note on wind generator interaction*. English. Risø-M 2411. Risø National Laboratory, 1983. ISBN: 87-550-0971-9.

[43] Meiyu Jiang et al. *A novel automatic wind power prediction framework based on multi-time scale and temporal attention mechanisms*. 2023. arXiv: 2302.01222 [cs.LG].

[44] Angel Jimenez, Antonio Crespo, and Emilio Migoya. “Application of a LES technique to characterize the wake deflection of a wind turbine in yaw”. In: *Wind Energy* 13 (Sept. 2009), pp. 559–572. DOI: 10.1002/we.380.

[45] Yun Ju et al. “A Model Combining Convolutional Neural Network and LightGBM Algorithm for Ultra-Short-Term Wind Power Forecasting”. In: *IEEE Access* (2019). DOI: 10.1109/access.2019.2901920.

[46] Ali C. Kheirabadi and Ryozo Nagamune. “A quantitative review of wind farm control with the objective of wind farm power maximization”. In: *Journal of Wind Engineering and Industrial Aerodynamics* 192 (2019), pp. 45–73. ISSN: 0167-6105. DOI: https://doi.org/10.1016/j.jweia.2019.06.015. URL: https://www.sciencedirect.com/science/article/pii/S0167610519305240.

[47] Mahdi Khodayar and Jianhui Wang. “Spatio-temporal graph deep neural network for short-term wind speed forecasting”. In: *IEEE Transactions on Sustainable Energy* 10.2 (2018), pp. 670–681. DOI: 10.1109/TSTE.2018.2844102.

[48] Jennifer King et al. “Aerodynamics of Wake Steering”. In: *Handbook of Wind Energy Aerodynamics*. Cham: Springer International Publishing, 2022, pp. 1197–1221. ISBN: 978-3-030-31307-4. DOI: 10.1007/978-3-030-31307-4_60. URL: https://doi.org/10.1007/978-3-030-31307-4_60.

[49] Torben Knudsen, Thomas Bak, and Mikael Svenstrup. “Survey of wind farm control—power and fatigue optimization”. In: *Wind Energy* 18.8 (2015), pp. 1333–1351. DOI: https://doi.org/10.1002/we.1760. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/we.1760.

[50] Min-Seung Ko et al. “Deep Concatenated Residual Network With Bidirectional LSTM for One-Hour-Ahead Wind Power Forecasting”. In: *IEEE Transactions on Sustainable Energy* 12.2 (2021), pp. 1321–1335. DOI: 10.1109/TSTE.2020.3043884.

[51] Linsen Li, Qichen Sun, and Dongdong Geng. *Complementary Fusion of Deep Spatio-Temporal Network and Tree Model for Wind Power Forecasting*. 2022.

[52] Xuechao Liao, Zhenxing Liu, and Wanxiong Deng. “Short-term wind speed multistep combined forecasting model based on two-stage decomposition and LSTM”. In: *Wind Energy* 24.9 (2021), pp. 991–1012. DOI: https://doi-org.tudelft.idm.oclc.org/10.1002/we.2613. eprint: https://onlinelibrary-wiley-com.tudelft.idm.oclc.org/doi/pdf/10.1002/we.2613. URL: https://onlinelibrary-wiley-com.tudelft.idm.oclc.org/doi/abs/10.1002/we.2613.

[53] Bryan Lim et al. *Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting*. 2020. arXiv: 1912.09363 [stat.ML].

[54] Zachary C. Lipton, John Berkowitz, and Charles Elkan. *A Critical Review of Recurrent Neural Networks for Sequence Learning*. 2015. arXiv: 1506.00019 [cs.LG].

[55] Hui Liu, Xiwei Mi, and Yan-fei Li. “Wind speed forecasting method based on deep learning strategy using empirical wavelet transform, long short term memory neural network and Elman neural network”. In: *Energy Conversion and Management* 156 (Jan. 2018), pp. 498–514. DOI: 10.1016/j.enconman.2017.11.053.

[56] Hui Liu, Xiwei Mi, and Yanfei Li. “Smart multi-step deep learning model for wind speed forecasting based on variational mode decomposition, singular spectrum analysis, LSTM network and ELM”. In: *Energy Conversion and Management* 159 (2018), pp. 54–64. ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2018.01.010. URL: https://www.sciencedirect.com/science/article/pii/S0196890418300104.

[57] Hui Liu, Hong-qi Tian, and Yan-fei Li. “Comparison of two new ARIMA-ANN and ARIMA-Kalman hybrid methods for wind speed prediction”. In: *Applied Energy* 98 (2012), pp. 415–424. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2012.04.001. URL: https://www.sciencedirect.com/science/article/pii/S0306261912002875.

[58] Yu Liu et al. "Available power estimation of wind farms based on deep spatio-temporal neural networks". In: *Frontiers in Energy Research* 11 (2023). ISSN: 2296-598X. DOI: 10.3389/fenrg.2023.1032867. URL: https://www.frontiersin.org/articles/10.3389/fenrg.2023.1032867.

[59] Peng Lu et al. "Short-term wind power forecasting based on meteorological feature extraction and optimization strategy". In: *Renewable Energy* (2021). DOI: 10.1016/j.renene.2021.11.072.

[60] X.J. Luo and Lukumon O. Oyedele. "Forecasting building energy consumption: Adaptive long-short term memory neural networks driven by genetic algorithm". In: *Advanced Engineering Informatics* 50 (2021), p. 101357. ISSN: 1474-0346. DOI: https://doi.org/10.1016/j.aei.2021.101357. URL: https://www.sciencedirect.com/science/article/pii/S1474034621001105.

[61] Quoc Luu, Son Nguyen, and Uyen Phạm. "Time series prediction: A combination of Long Short-Term Memory and structural time series models". In: *Science & Technology Development Journal - Economics - Law and Management* 4 (Apr. 2020), pp. 500–515. DOI: 10.32508/stdjelm.v4i1.593.

[62] Ignacio Martí et al. "Wind power prediction in complex terrain: from the synoptic scale to the local scale". In: (Jan. 2004).

[63] Marcos S. Miranda and Rod W. Dunn. "One-hour-ahead wind speed prediction using a Bayesian methodology". In: 2006. DOI: 10.1109/pes.2006.1709479. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-35348877883&doi=10.1109%2fpes.2006.1709479&partnerID=40&md5=7fe53c956c546a36745e7f2cf478466e.

[64] N. Mittelmeier and M. Kühn. "Determination of optimal wind turbine alignment into the wind and detection of alignment changes with SCADA data". In: *Wind Energy Science* 3.1 (2018), pp. 395–408. DOI: 10.5194/wes-3-395-2018. URL: https://wes.copernicus.org/articles/3/395/2018/.

[65] Zhewen Niu et al. "Wind power forecasting using attention-based gated recurrent unit network". In: *Energy* 196 (2020), p. 117081. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2020.117081. URL: https://www.sciencedirect.com/science/article/pii/S0360544220301882.

[66] Christopher Olah. *Understanding LSTM Networks*. 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[67] Jinkyoo Park and Kincho H. Law. "Layout optimization for maximizing wind farm power production using sequential convex programming". In: *Applied Energy* 151 (2015), pp. 320–334. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2015.03.139. URL: https://www.sciencedirect.com/science/article/pii/S0306261915004560.

[68] Junyoung Park and Jinkyoo Park. "Physics-induced graph neural network: An application to wind-farm power estimation". In: *Energy* 187 (Nov. 2019). ISSN: 03605442. DOI: 10.1016/j.energy.2019.115883.

[69] Suraj Pawar et al. “Towards multi-fidelity deep learning of wind turbine wakes”. In: *Renewable Energy* 200 (Nov. 2022), pp. 867–879. ISSN: 18790682. DOI: 10.1016/j.renene.2022.10.013.

[70] Paweł Piotrowski et al. “Evaluation Metrics for Wind Power Forecasts: A Comprehensive Review and Statistical Analysis of Errors”. In: *Energies* 15.24 (2022). ISSN: 1996-1073. DOI: 10.3390/en15249657. URL: https://www.mdpi.com/1996-1073/15/24/9657.

[71] Charlie Plumley. *Penmanshiel Wind Farm Data*. Version 0.0.2. Zenodo, Feb. 2022. DOI: 10.5281/zenodo.5946808. URL: https://doi.org/10.5281/zenodo.5946808.

[72] Luana Ruiz, Fernando Gama, and Alejandro Ribeiro. “Gated Graph Recurrent Neural Networks”. In: *IEEE Transactions on Signal Processing* 68 (2020), pp. 6303–6318. ISSN: 1941-0476. DOI: 10.1109/tsp.2020.3033962. URL: http://dx.doi.org/10.1109/TSP.2020.3033962.

[73] Debashis Sahoo et al. “Comparative Analysis of Multi-Step Time-Series Forecasting for Network Load Dataset”. In: July 2020. DOI: 10.1109/ICCCNT49239.2020.9225449.

[74] Fathi M. Salem. “Recurrent Neural Networks (RNN)”. In: *Recurrent Neural Networks: From Simple to Gated Architectures*. Cham: Springer International Publishing, 2022, pp. 43–67. ISBN: 978-3-030-89929-5. DOI: 10.1007/978-3-030-89929-5_3. URL: https://doi.org/10.1007/978-3-030-89929-5_3.

[75] Farah Shahid et al. “A novel wavenets long short term memory paradigm for wind power prediction”. In: *Applied Energy* 269 (2020), p. 115098. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2020.115098. URL: https://www.sciencedirect.com/science/article/pii/S0306261920306103.

[76] Jing Shi, Xiuli Qu, and Songtao Zeng. “Short-term wind power generation forecasting: Direct versus indirect ARIMA-based approaches”. In: *International Journal of Green Energy* 8.1 (2011). Cited by: 53, pp. 100–112. DOI: 10.1080/15435075.2011.546755. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-79951847299&doi=10.1080%2f15435075.2011.546755&partnerID=40&md5=490d408368b7971fd5e9ad34bdffc316.

[77] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. “A Comparison of ARIMA and LSTM in Forecasting Time Series”. In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2018, pp. 1394–1401. DOI: 10.1109/ICMLA.2018.00227.

[78] Christopher A. Sims. “Macroeconomics and Reality”. In: *Econometrica* 48.1 (1980), pp. 1–48. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/1912017 (visited on 04/12/2024).

[79] Stanley Smith and Terry Sincich. “An Empirical Analysis of the Effect of Length of Forecast Horizon on Population Forecast Errors”. In: *Demography* 28 (June 1991), pp. 261–74. DOI: 10.2307/2061279.

[80] Tomasz Stańczyk and Siamak Mehrkanoon. *Deep Graph Convolutional Networks for Wind Speed Prediction*. 2021. arXiv: 2101.10041 [cs.LG].

[81] Xiao-Yu Tang et al. “Optimization of wind farm layout with optimum coordination of turbine cooperations”. In: *Computers & Industrial Engineering* 164 (2022), p. 107880. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2021.107880. URL: https://www.sciencedirect.com/science/article/pii/S0360835221007841.

[82] Jean-François Toubeau et al. “Deep Learning-Based Multivariate Probabilistic Forecasting for Short-Term Scheduling in Power Markets”. In: *IEEE Transactions on Power Systems* 34.2 (2019), pp. 1203–1215. DOI: 10.1109/TPWRS.2018.2870041.

[83] Ilham Tyass et al. “Wind Speed Prediction Based on Seasonal ARIMA model”. In: vol. 336. 2022. DOI: 10.1051/e3sconf/202233600034. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85130488599&doi=10.1051%2fe3sconf%2f202233600034&partnerID=40&md5=d105b33ff2751fc6808362f39587be8f.

[84] Soraida Aguilar Vargas et al. “Wind power generation: A review and a research agenda”. In: *Journal of Cleaner Production* 218 (2019), pp. 850–870. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2019.02.015. URL: https://www.sciencedirect.com/science/article/pii/S0959652619303944.

[85] Det Norske Veritas and DK Wind Energy Department. *Guidelines for design of wind turbines*. Contract ENS-51171/98-0036. 2001. ISBN: 9788755028708. URL: https://books.google.nl/books?id=XX5pSQAACAAJ.

[86] Lei Wang and He Yigang. “M2STAN: Multi-modal multi-task spatiotemporal attention network for multi-location ultra-short-term wind power multi-step predictions”. In: *Applied Energy* 324 (July 2022), p. 119672. DOI: 10.1016/j.apenergy.2022.119672.

[87] Binrong Wu, Lin Wang, and Yu-Rong Zeng. “Interpretable wind speed prediction with multivariate time series and temporal fusion transformers”. In: *Energy* 252 (2022), p. 123990. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2022.123990. URL: https://www.sciencedirect.com/science/article/pii/S0360544222008933.

[88] Qianyu Wu et al. “Ultra-short-term multi-step wind power forecasting based on CNN-LSTM”. In: *Iet Renewable Power Generation* (2021). DOI: 10.1049/rpg2.12085.

[89] Qu Xiaoyun et al. “Short-term prediction of wind power based on deep Long Short-Term Memory”. In: *2016 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*. 2016, pp. 1148–1152. DOI: 10.1109/APPEEC.2016.7779672.

[90] Chi Yan, Yang Pan, and Cristina L. Archer. “A general method to estimate wind farm power using artificial neural networks”. In: *Wind Energy* 22.11 (2019), pp. 1421–1432. DOI: https://doi.org/10.1002/we.2379. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/we.2379.

[91] Bing Yu, Haoteng Yin, and Zhanxing Zhu. "Spatio-temporal Graph Convolutional Neural Network: A Deep Learning Framework for Traffic Forecasting". In: *CoRR* abs/1709.04875 (2017). arXiv: 1709.04875. URL: http://arxiv.org/abs/1709.04875.

[92] Mei Yu et al. "Superposition Graph Neural Network for offshore wind power prediction". In: *Future Generation Computer Systems* (2020). DOI: 10.1016/j.future.2020.06.024.

[93] Yong Yu et al. "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures". In: *Neural Computation* 31.7 (2019), pp. 1235–1270. DOI: 10.1162/neco_a_01199.

[94] Jianjun Zhang et al. "Combined Wind Speed Prediction Model Considering the Spatio-Temporal Features of Wind Farm". In: Institute of Electrical and Electronics Engineers Inc., 2022, pp. 132–138. ISBN: 9781665466745. DOI: 10.1109/ICCCR54399.2022.9790115.

[95] Jinhua Zhang et al. "Short-term forecasting and uncertainty analysis of wind turbine power based on long short-term memory network and Gaussian mixture model". In: *Applied Energy* 241 (2019), pp. 229–244. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2019.03.044. URL: https://www.sciencedirect.com/science/article/pii/S0306261919304532.

[96] Hongjun Zhao et al. "Ultra-short-term Power Forecasting of Wind Farm Cluster Based on Spatio-temporal Graph Neural Network Pattern Prediction". In: *2022 IEEE Industry Applications Society Annual Meeting (IAS)*. 2022, pp. 1–25. DOI: 10.1109/IAS54023.2022.9939731.

[97] Yongning Zhao et al. "Correlation-Constrained and Sparsity-Controlled Vector Autoregressive Model for Spatio-Temporal Wind Power Forecasting". In: *IEEE Transactions on Power Systems* (2018). DOI: 10.1109/TPWRS.2018.2794450. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85041689621&doi=10.1109%2fTPWRS.2018.2794450&partnerID=40&md5=ea300f620fe4b58ffd43354b3eab3dd3.

[98] Hao Zhen et al. "A Hybrid Deep Learning Model and Comparison for Wind Power Forecasting Considering Temporal-Spatial Feature Extraction". In: *Sustainability* (2020). DOI: 10.3390/su12229490.

[99] Jingbo Zhou et al. *SDWPF: A Dataset for Spatial Dynamic Wind Power Forecasting Challenge at KDD Cup 2022*. 2022. arXiv: 2208.04360 [cs.LG].

[100] Jingwei Zuo et al. *Graph Convolutional Networks for Traffic Forecasting with Missing Values*. 2022. arXiv: 2212.06419 [cs.LG].

A SDWPF Dataset

Experimental Power Curve

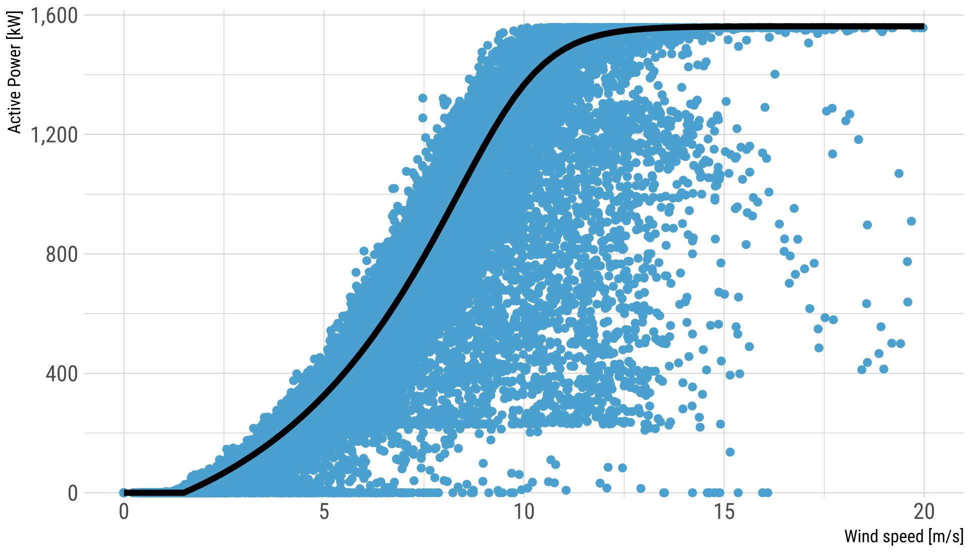


Figure A.1: Experimental power curve for all turbines across all time-steps.

Auto-correlation Function

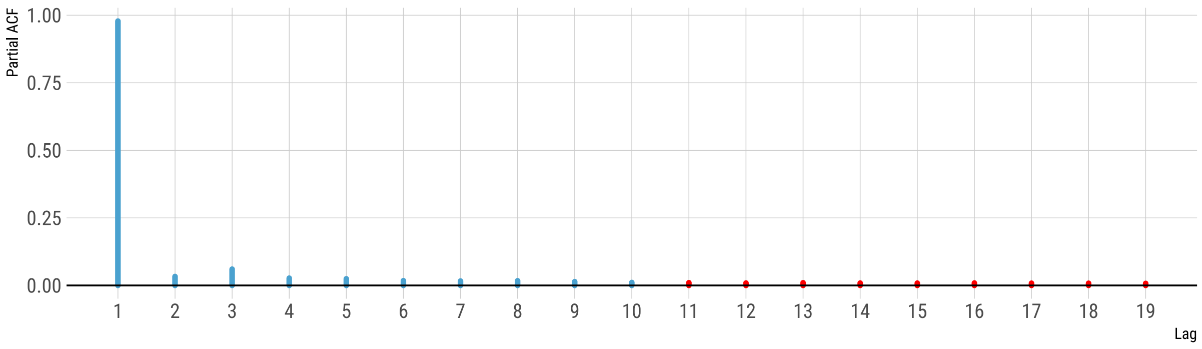


Figure A.2: Average absolute partial ACF for all turbines. Bars coloured blue exceed the significance threshold of 0.01, while those in red fall below this threshold.

Distribution of the Wind Direction

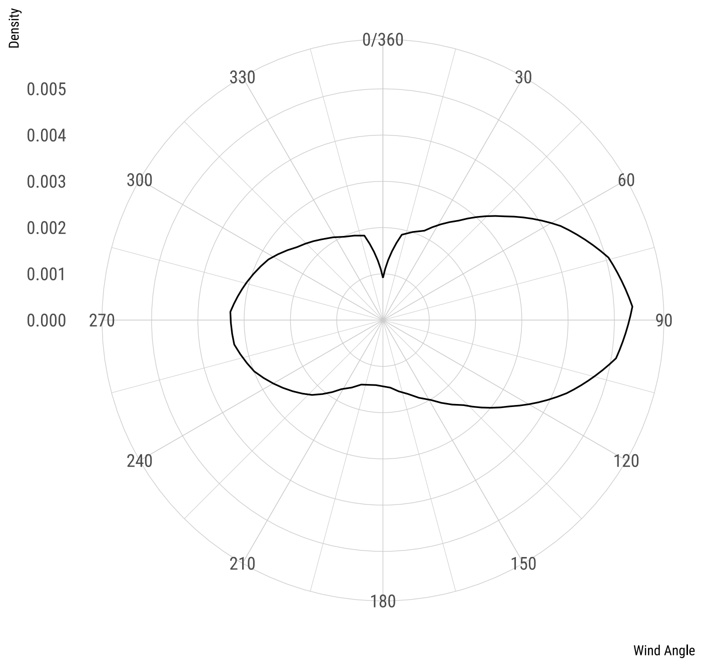


Figure A.3: Distribution of the wind direction across the entire dataset.

B Penmanshiel Dataset

Experimental Power Curve

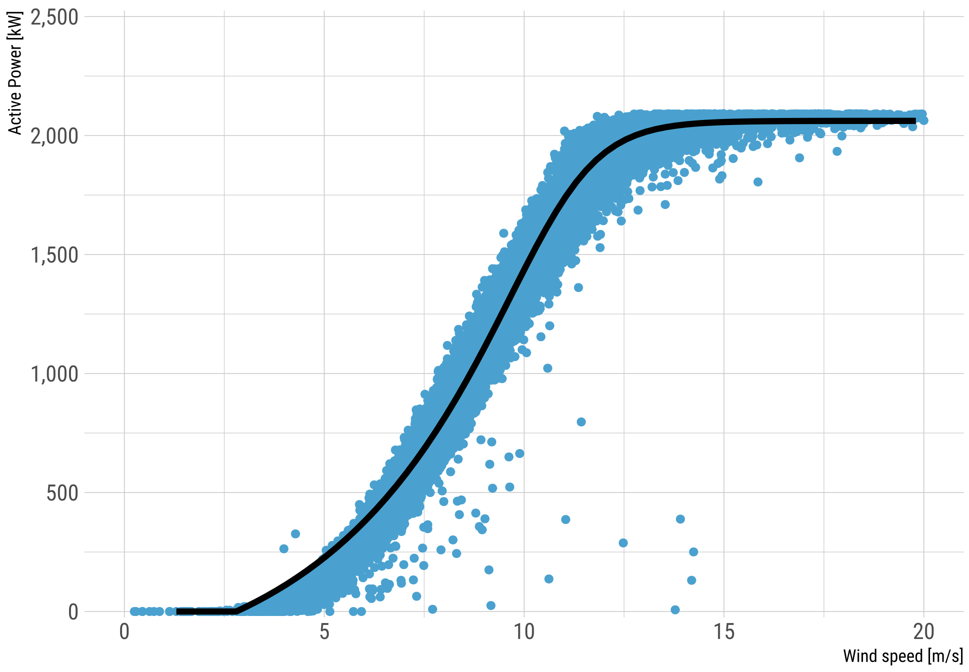


Figure B.1: Experimental power curve for all turbines across all time-steps.

Auto-correlation Function

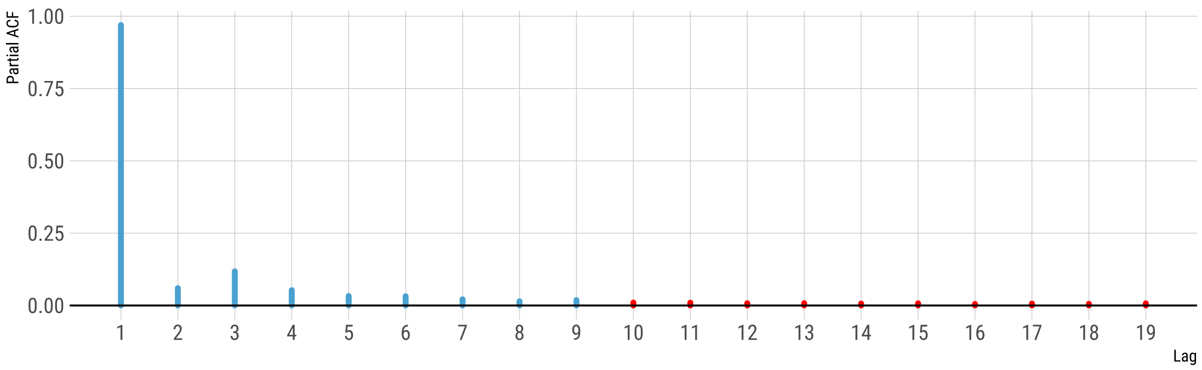


Figure B.2: The average absolute partial ACF for all turbines is shown. Bars colored blue exceed the significance threshold of 0.01, while red bars fall below this threshold. Although the time-series behavior is very similar to that of the SDWPF dataset (Figure B.3), the significance for the Penmanshiel dataset changes with a lag of 9 time-steps.

Distribution of the Wind Direction

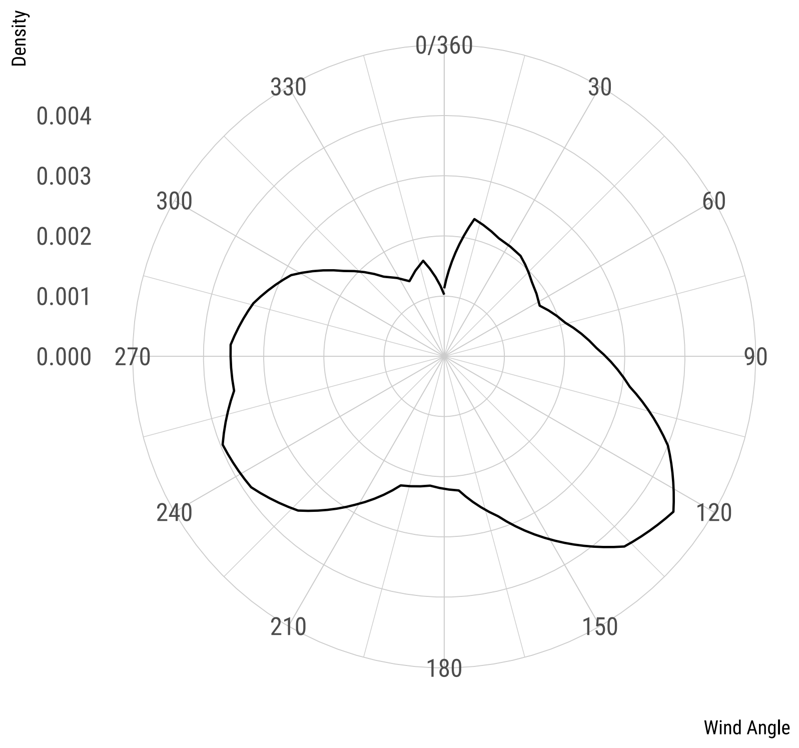


Figure B.3: Distribution of the wind direction across the entire dataset.