Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

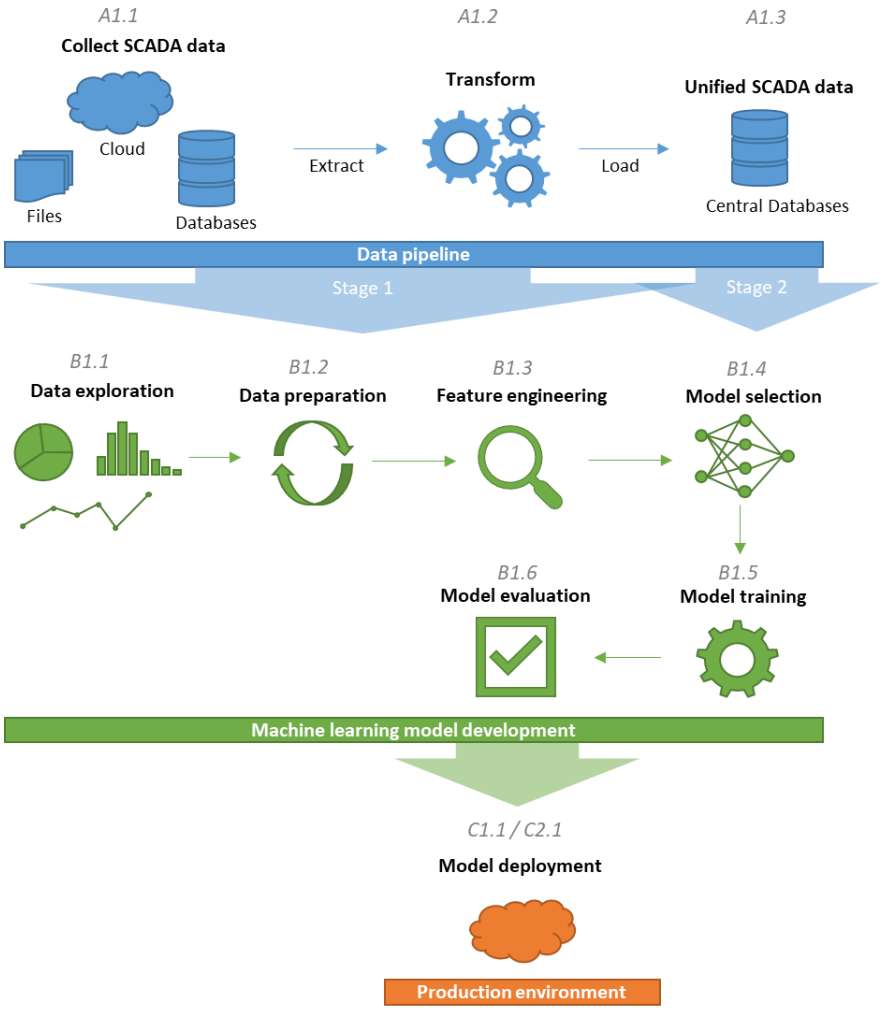
Federal Department of the Environment, Transport, Energy and Communications DETEC

Swiss Federal Office of Energy SFOE
Energy Research and Cleantech Division

Final report dated 01.12.2023

OpenIMPACT

Development of an open-source library for applying novel Machine Learning algorithms for optimising wind farm performance in complex terrain based on SCADA data



Source: ©OST 2023

Date: 01.12.2023

Location: Rapperswil

Publisher:
Swiss Federal Office of Energy SFOE
Energy Research and Cleantech
CH-3003 Bern
www.bfe.admin.ch

Co-financing:
WinJi AG
Badenerstrasse 808
8048 Zürich

Subsidy recipients:
OST – Ostschweizer Fachhochschule
Oberseestrasse 10
8640 Rapperswil

Authors:
Florian Hammer, OST, florian.hammer@ost.ch
Nora Helbig, OST, nora.helbig@ost.ch
Sarah Barber, OST, sarah.barber@ost.ch

SFOE project coordinators:
Katja Maus, katja.maus@bfe.admin.ch
Lionel Perret, lionel.perret@planair.ch

SFOE contract number: SI/502329-01

The authors bear the entire responsibility for the content of this report and for the conclusions drawn therefrom.

Zusammenfassung

Das Ziel dieser Arbeit war die Erstellung einer Open-Source-Bibliothek, **openimpact**, zur Modellierung von Windparks in komplexem Gelände unter Verwendung von Supervisory control and data acquisition (SCADA)-Daten. Drei Anwendungsfälle wurden untersucht: der Einfluss von Nachlaufinteraktionen und die Wirksamkeit von Turbinen-Upgrades (bereitgestellt von WinJi AG und Elektrizitätswerke des Kantons Zürich (EKZ)) sowie ein dritter Fall, basierend auf einem Open-Source-Datensatz, um die Fähigkeiten der Bibliothek zu demonstrieren.

Es wurde eine Datenpipeline für das Einlesen von Daten in ein Data Warehouse und deren Umwandlung in ein Standardformat nach IEC 61400-25 Konventionen erstellt. Dieser Prozess ist in **openimpact** für dokumentiert.

Für den ersten Anwendungsfall wurden ein Graphenmodell und ein Graph Neural Network (GNN) entwickelt, um Nachlaufinteraktionseffekte vorherzusagen. Das Graphenmodell verknüpfte die Leistungsdaten der Turbinen mit denen der benachbarten Turbinen, war aber auf ein Feature (Leistungsdaten) beschränkt. Deshalb wurde ein GNN entwickelt, welches weitere Features wie Windgeschwindigkeiten, Gierfehler und räumliche Grössen integriert. Anhand der GraphGym Bibliothek wurde das GNN Design entworfen und die Hyperparameter wurden mittels Bayes'sche Optimierung ermittelt. Das Modell konnte erfolgreich Nachlaufinteraktionen erfassen und Leistungskurven vorhersagen, unterschätzte jedoch die Gesamtenergieproduktion um 4%.

Der zweite Anwendungsfall befasste sich mit der Analyse eines Windparks mit nachgerüsteten Turbinen. Dabei wurde ein Problem mit unzuverlässigen Windgeschwindigkeitsmessungen nach dem Upgrade festgestellt. Deshalb wurde ein GNN-Modell erstellt, um Windgeschwindigkeiten für diese Turbinen zu schätzen, unter Verwendung von Daten benachbarter Turbinen. Die Genauigkeit des Modells wurde durch den Vergleich der Leistungskoeffizienten vor und nach dem Upgrade demonstriert, wobei eine deutliche Verbesserung festgestellt wurde. Weitere Validierungen sind jedoch in Zusammenarbeit mit dem Projektpartner erforderlich. Darüber hinaus wurde der positive Einfluss der Upgrades auf die Turbinenleistung bestätigt. Anhand der Daten, welche die fehlerhaften Windgeschwindigkeitsmessungen beinhalten, wurde eine Verbesserung der Turbinenleistung von über 13% festgestellt. Nach der Ersetzung der fehlerhaften Windgeschwindigkeiten mit denen des vom GNN geschätzten Windgeschwindigkeiten, zeigte sich eine realistischere Verbesserung in Übereinstimmung mit Literaturwerten von etwa 4%.

Der dritte Anwendungsfall diente zur Demonstration der **openimpact**-Bibliothek. In der Bibliothek, veröffentlich auf Github, sind eine Datenverarbeitungspipeline, Skripte für die Hyperparametereinstellung, GNN-Training sowie eine API basierend auf FastAPI für den Modell-Einsatz enthalten.

Insgesamt zeigten die entwickelten Modelle Potenzial zur Verbesserung in der Windparkmodellierung. Jedoch wurden auch einige Einschränkungen festgestellt, wie die Unterschätzung der Energieproduktion.

Summary

This work focused on creating an open-source library, **openimpact**, for modelling wind farm performance in complex terrain using Supervisory control and data acquisition (SCADA) data. Three use cases were explored to investigate the wake interaction effect, and theturbine upgrade efficacy (provided by WinJi AG and Elektrizitätswerke des Kantons Zürich (EKZ)), and a third use case was based on an open-source dataset to demonstrate the library's capabilities.

A data pipeline was established for ingesting data into a data warehouse and transforming it into a standard format according to IEC 61400-25 conventions. This process is documented in **openimpact**.

For the first use case, a graph model, and a Graph Neural Network (GNN) were developed to predict wake interaction effects. The graph model linked turbine power outputs with neighboring turbines but was limited to one feature (power output). The GNN, enhanced to include features such as wind speeds, yaw errors, and spatial values. The GNN design was determined based on a controlled random search with GraphGym and further hyperparameters were tuned using Bayesian optimisation. It successfully captured wake interactions and predicted power curves, though it underestimated total energy production by 4%. Further work on GNN design and hyperparamter optimization is needed.

The second use case involved analysing a wind farm with retrofitted turbines. It highlighted an issue with unreliable wind speed measurements post-upgrade. A GNN model was created to impute wind speeds for these turbines, using data from neighboring ones. The model's accuracy was demonstrated by comparing power coefficients before and after the upgrades, revealing a notable improvement. However, further validation is needed with the project partner. Moreover, the upgrades' positive impact on turbine performance was confirmed. Initial data, potentially affected by faulty wind speed measurements, suggested an over 13% improvement. However, when adjusted with the GNN imputed wind speeds, the improvement aligned more realistically with literature values, showing an increase of approximately 4%.

The third use case utilised the **openimpact** library for a standalone deployment approach. It includes a data processing pipeline, scripts for hyperparameter tuning, GNN training, and an API based on FastAPI for model deployment.

Overall, the library and the developed models showed promise in enhancing wind farm modelling, though some limitations were identified, such as the underestimation of energy production.

Main findings

1. Development of the **openimpact** library: An open-source library, **openimpact**, was created to for wind farm modelling.
2. Use Case 1 - Wake Interaction Effects: A graph model and a Graph Neural Network (GNN) were developed to predict wake-interaction effects in wind farms. The graph model successfully related turbine power outputs with those of neighbouring turbines, capturing the wake effects. However, it was limited to using only power output as a feature. The GNN, enhanced with additional features like wind speeds, yaw errors, and spatial data, was able to learn attention coefficients to determine the importance of neighbouring turbines in predicting wake interactions. Despite its effectiveness, the current GNN underestimated total energy production by approximately 4%. The very next step should be further model optimisation to improve on this.
3. Use Case 2 - Turbine Upgrade Efficacy: In a large wind farm where turbines were retrofitted with vortex generators and Gurney flaps, a GNN model was developed to predict wind speeds for the upgraded turbines, addressing issues with unreliable wind speed measurements due to missing anemometer recalibration. The model's accuracy was demonstrated by comparing power coefficients before and after the upgrades, revealing a notable improvement. Furthermore, the upgrades' positive impact on turbine performance was confirmed, showing an increase of approximately 3.8%.
4. Use Case 3 - Demonstrating Library Capabilities: The third use case involved the use of the **openimpact** library, demonstrating its capabilities with a data processing pipeline, hyperparameter tuning scripts, GNN training, and an API based on FastAPI for deploying the model.

Contents

Abbreviations

CNN	Convolutional neural network
ETL	Extraction, transformation and loading
GNN	Graph Neural Network
RNN	Recurrent neural network
SCADA	Supervisory control and data acquisition
SQL	Structured Query Language
WTG	Wind turbine generator
MLP	Multi-layer perceptron
AEP	Annual energy production

1 Introduction

1.1 Background information and current situation

Wind energy project planning and operation is particularly difficult in Switzerland due to its complex terrain and challenging weather conditions. For example, diurnal winds occur due to the interaction between thermal effects and the complex terrain [1], [2]. This leads to a reduced accuracy of wind modelling and performance assessment using conventional methods, such as power curve analysis using the standard method of binning from IEC 61400-12 [3]. This requires Swiss wind farm planners and operators to apply a range of different advanced modelling and analysis techniques to each site, making it difficult to transfer experience and methods between projects. Furthermore, the Swiss Energy Strategy 2050 sets the goal of reaching 4,300 GWh of wind energy by 2050, which amounts to approximately 1,000 new MW-scale wind turbines– about 40 per year. A large amount of these new installations will be in complex terrain or areas of challenging weather conditions.

The power generated by a wind turbine generator (WTG) is dependent on the atmospheric conditions, such as wind speed, air density, turbulence intensity and shear. The power output of WTGs is further affected by wakes of upstream WTGs, which is referred to as “wake interaction losses” [4]. Here, yaw misalignment, which refers to the difference between the wind direction and the nacelle position, can play a large role. Better understanding of these various effects is important to optimise the total power output of a wind farm, rather than just optimising single WTGs, leading to a higher Annual Energy production (AEP) and a reduction in the Levelized Cost of Electricity [5]. On the other hand, to increase the efficiency and with this the power generated of a WTG or to counteract the performance degradation that occurs over a turbine’s lifetime, a retrofit, also called upgrade, can be performed. Here two kinds of upgrade approaches are distinguished, namely active and passive upgrades. In the case of active upgrades, the control strategy of turbines is altered to gain an efficiency boost. Passive upgrades are devices installed on the turbine, most commonly the blades, such as vortex generators and/or gurney flaps [6]. Ding [6] noted that vortex generators could increase performance by 1-5%. Further exploration through flow simulations of vortex generators on turbine blades revealed power gains of 2.5% [7]. Astolfi et al. [8] found performance increases of up to 3.9% due to vortex generators and passive flow control devices. In order to be able to apply advanced modelling and analysis techniques for the operational decision-making process, large amounts of data are paramount. In the digital era, data is increasingly becoming a valuable asset for creating and maintaining competitive advantage. The standard data obtained from all wind turbine generators by owner/operators is the 10-minute averaged Supervisory Control and Data Acquisition (SCADA) data, which includes the wind speed and direction, the power production, the nacelle position, the rotational speed and the temperature of the generator, and more. There are many possibilities to enhance this data with additional, higher-frequency measurements (e.g. Condition Monitoring Systems for detecting drivetrain vibrations), but these solutions are expensive, difficult to install or intrusive [9].

Therefore, many recent studies attempt to exploit SCADA data for performance improvement as far as possible. Machine learning models have been shown to have high potential for doing this [9]–[16], by increasing the accuracy of power curve predictions and hence helping owner/operators to improve their control strategies by closely monitoring the actual compared to the potential power output. For example, [12] compared different modelling techniques and found that their model reduces the power curve prediction error by up to 45% compared to the standard method of binning [3]. In comparison to other models, such as the bivariate kernel model, they managed to improve their predictions through the choice of model and through further considering atmospheric conditions in more detail. Additionally, in the area of turbine upgrade quantification accurate power curve models are needed as well.

On the wind farm level, one common approach to reducing the impact of wake interaction losses is wake steering. Under yaw misalignment conditions, the wake behind the WTG is deflected to a certain degree. This technique is used for steering wakes away from downstream turbines, which can lead to an increase in overall power production of a wind farm [5]. Wake effects can be modelled in three different ways, namely: physics-based formulations, Computational Fluid Dynamics simulations and data-driven models. The advantage of

data-driven models is their efficiency and the incorporation of in-field data of an actual wind farm [17]. More recent data-driven approaches that model wake effects are graph neural networks, which represent the wind farm as a graph, where each WTG is a node, which is connected to other nodes by edges. Edges contain weighting factors, indicating the strength of the connection, or in our case, interaction between neighbouring WTGs.

There is a high potential for developing these methods further for improving the performance of entire wind farms, considering the terrain, the weather and the wake interactions between wind turbines. These models could help owner/operators adjust the control parameters and set points of individual wind turbines optimally, hence improving performance and increasing revenues, especially in hilly and mountainous terrain.

1.2 Purpose of the project

The purpose of this project is to develop an open-source library for applying novel machine learning solutions for optimising wind farm performance in complex terrain based on SCADA data. We focus on complex terrain in order to serve the needs of the Swiss market as well as Swiss product and service providers. Our models will consider complex terrain, weather, and wake interaction effects in power predictions. An improved understanding of these phenomena will allow us to develop an open-source library that contains algorithms and methods applicable to wind farms at complex sites. Investigations into the transferability of trained models to other wind turbines within the wind farm will help reduce the computational power required.

1.3 Objectives

The objectives of this project are to:

1. Create a data pipeline, which includes one specific implementation (use case 1) as well as an open-source framework that allows users to build their own pipeline.
2. Investigate data-driven methods for reducing wake losses in operation (use case 1).
3. Investigate data-driven methods for quantifying the effect of performance upgrades (use case 2).
4. Publish an open-source library that is easy to use for owner/operators and includes generalised methods developed from the results of Objectives 2 and 3 as well as an example based on an existing open-source dataset (use case 3).
5. Create a production environment specific to use case 1.

The three use cases are described below.

Use case 1: Wake Effects within a Wind Farm (WinJi AG)

For this use case, a dataset of a wind farm that suffers from performance losses due to wakes was provided by WinJi AG. An example of WTGs in a wind farm being influenced by wakes is shown in Figure 1. The specific goal of this use case is to develop a data-driven model that takes wind farm wake losses into account and predicts their influence on AEP per turbine and per wind farm. The model could then be used, for example, to test various configurations, such as nacelle positions, and the gained information could then be used for adjusting control strategies. Before this project, WinJi AG had only carried out some brief investigations, mostly regarding basic wake effect estimations based on layout and planned losses.

The provided dataset comprises nacelle anemometry parameters, production parameters, technical parameters as well as flags indicating abnormal operation periods (curtailments, faults etc.) of a wind farm containing 50 WTGs for a period of two years. All parameters are 10 minute-averaged and anonymised. The data is stored in a csv file, accumulating to around 1.5 GB size per year for the entire wind farm. Additional goals for this use case specified by the partner are:

- Improvement of the current process of choosing and optimising the right model for a given problem.
- Deploying solutions into a production environment.

An important point to note is that the provided data is confidential. For this reason, we decided, in agreement with the project partner, to only present the results for a cluster of six out of the 50 WTGs in the wind farm within this report.

Figure 1. Horns Rev 2 wind farm (Source: [18])

Use case 2: Turbine Upgrades within a Wind Farm (EKZ)

This use case deals with the incorporation of vortex generators and Gurney flaps on WTG blades in a wind farm that alters the performance of the whole WTG. The goal is to explore methods for identifying and quantifying the expected net effect on the turbine performance of those upgrades.

SCADA data of a wind farm containing 20 WTGs for the period from 2013 until the present day was provided by Elektrizitätswerke des Kantons Zürich (EKZ). The dataset can be split into three parts:

- A period where all 20 WTGs operate without turbine upgrades
- A period where 4 out of 20 WTGs operate with turbine upgrades
- A period where all 20 WTGs operate with turbine upgrades

The large amount of data for these three distinct periods will greatly help to distinguish the effects of turbine degradation and turbine upgrades, building a model to predict the effect of turbine upgrades and eventually test the quality and accuracy of the predictions.

The data was provided in the form of csv files for each turbine. The accumulated size of the dataset is 5 GB per year for the whole wind farm. Additionally, the exact locations and dates of the installation of the turbine upgrades was provided.

Figure 2. Gurney flap (left, Source: [19]) and vortex generators1 (right).

Use case 3: Open-source data (Cubico)

An open-source dataset of a wind farm in the UK was chosen for use case 3. The data was published by Cubico on Zenodo[1] [18] and contains, among other things,

- Static data including turbine coordinates and turbine details (rated power, rotor diameter, hub height, etc.)
- 10-minute SCADA data from the 6 Senvion MM92's at Kelmarsh wind farm, grouped by year from 2016 to mid-2021

This dataset has been used already for two challenges in our WeDoWind wind energy ecosystem[2]. The idea is to showcase the models as part of the developed open-source library. This will also allow to use the models easily and readily for further challenges on WeDoWind, and thus, continue developing the methodology within this work.

1.4 Deliverables

The deliverables of this project are:

- A new **data pipeline**, which includes one specific implementation (use case 1) as well as an open-source framework that allows users to build their own pipeline.
- A published **open-source library** that is easy to use for owner/operators and includes generalised methods for reducing wake losses in operation (use case 1) and quantifying the effect of performance upgrades (use case 2), as well as an example based on an existing open-source dataset (use case 3).
- A **production environment** specific to use case 1.

[1] https://zenodo.org/records/5841834
[2] https://www.wedowind.ch/

2 Procedures and methodology

In this section, the general methodology is first presented, followed by a description of the machine learning approach used.

2.1 General methodology

The general methodology applied in this work is summarised in Figure 3.

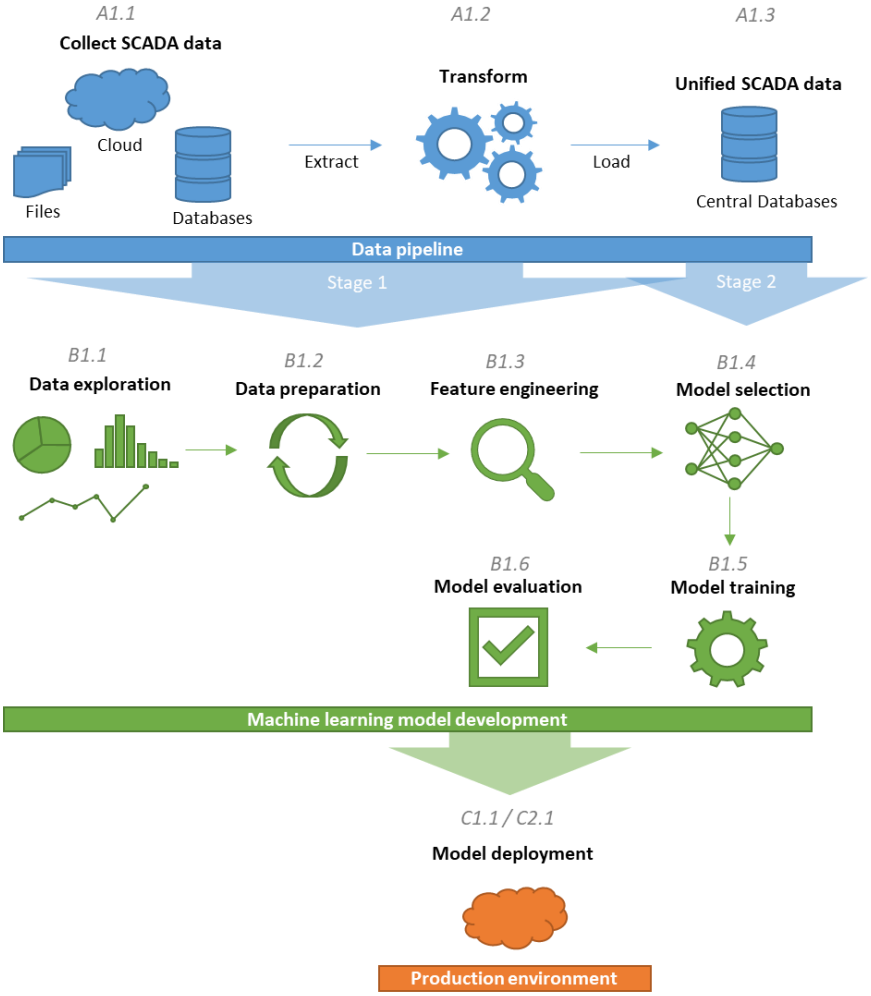


Figure 3. Concept of this project

The **data pipeline** (A) was built with the following modules:

- (A1.1.) Collection of SCADA data: this includes simple text files, databases, and cloud storage. Part of this data was provided by our two partners, who also developed concrete use cases to be solved. The other part was collected from literature (Zenodo[3]).
- (A1.2) Data extraction: the gathered raw data was extracted and transformed into a unified data set,
- (A1.3) Data loading: the unified data set was loaded into a central database.

[3] https://zenodo.org/records/8252025

The open-source python library (B), called **openimpact**[4], for building a data-driven dynamical system of a wind farm. A wind farm is a complex system where individual WTGs and the wind farm as a whole interact with the turbulent atmospheric boundary layer as well as with other WTGs in the wind farm due to wake effects. The state of the atmospheric boundary layer is further influenced and determined by meso- and macroscale weather phenomena, thermal effects, the topology of the surface [19]. Therefore, there are some very interesting and complex problems to be solved on the wind farm and the turbine level to optimise operation, such as:

1. Prediction problems
    a. Prediction of wind farm power production under certain conditions
    b. Prediction of turbine interaction losses
    c. Short-term wind power forecasting
2. Design and optimisation problems
    a. Wind park layout
    b. Turbine upgrade efficacy
    c. Passive wake steering
3. Control problems
    a. Active wake steering (feedback control), which requires high frequency data as opposed to passive wake steering
    b. Full state estimation for linear control methods
4. Understanding a system
    a. Wind farm wind flow patterns
    b. Influence of terrain on wind farm and WTG flows

In order to tackle these problems a wind farm can be viewed as a dynamical system (**Fehler! Verweisquelle konnte nicht gefunden werden.**), represented by equation

$$\frac{d}{dt}\boldsymbol{x}(t) = f(\boldsymbol{x}(t), t, \boldsymbol{u}; \beta)$$

***x**(t)* describes the state of the system. In case of a wind farm this could be the wind speed, the wind direction, the temperature on both the individual turbine and the wind farm level. The states of a system change over time, *t*. The turbines in a wind farm are not just subjected to the atmospheric boundary layer and the wakes of other turbines but can be actively controlled by the wind turbine controller, which adjusts the turbine pitch, yaw, rotor speed etc. depending on the prevailing conditions. These control variables, $\boldsymbol{u}$, need to be accounted for in the dynamical system. Lastly, a dynamical system contains parameters, β, which are fixed values that cannot be controlled, such as the turbine height, the topology of the surface and the wind farm layout. However, some of these parameters can be changed to some extent, such as altering the turbine aerodynamics through turbine upgrades, certain biases imposed on WTG yaw angles for passive wake steering etc.

[4] https://github.com/weid-ost/openimpact

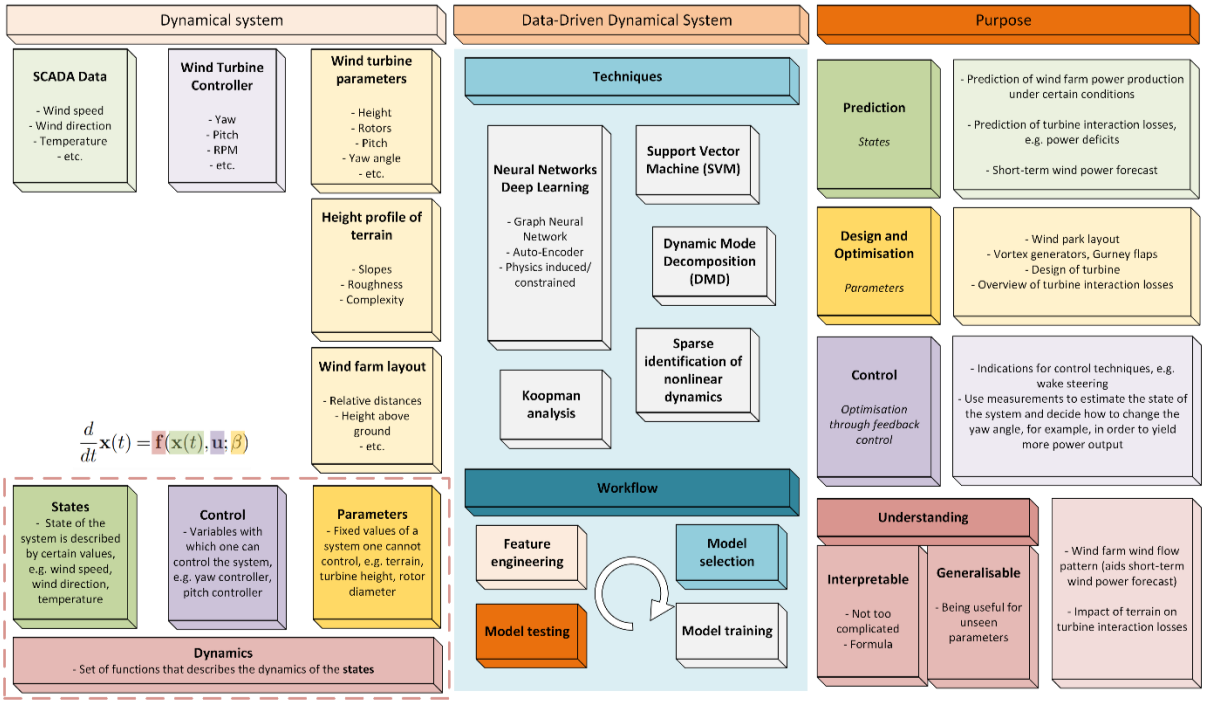


Figure 4. A wind farm viewed as a data-driven dynamical system

There are a variety of techniques and approaches to choose from in order to build a model that represents the data-driven dynamical system. This is highly dependent on the problem to be solved. Some of these methods are

- Dynamic Mode Decomposition [20]
- Neural Networks [21]
- Deep Learning [22]
- Graph Neural Networks [23], [24]
- Koopman operator theory [25], [26]

The **openimpact** library was built by the following steps:

- (B1.1) Data exploration: in-depth analysis of the available data was carried out in order to gain insights into the specific flow patterns of each wind farm. This helped to set a clear path for the successive steps.
- (B1.2) Data preparation: on the basis of the analysis, the data was prepared, i.e. cleaned, transformed and scaled.
- (B1.3) Feature engineering: we selected the most relevant features, such as wind speed, yaw error, spatial values, that maximise the model performance and at the same time do not compromise generalisability and interpretability.
- (B1.4) Model selection: we used a reduced amount of data and trained a variety of different graph neural network (GNN) architectures based on a controlled random search. The architecture with the best score was chosen for further steps. Moreover, for power curve modelling we decided to use boosted trees, as these have shown to perform well in previous work. See Section 2.2 for further details.

- (B1.5) Model training: the selected boosted tree and GNN architecture were then further optimised based on Bayesian optimisation. These models were then trained on the full datasets for later use.
- (B1.6) Model evaluation: Before the trained model can be used for further analysis or in a production environment, the final accuracy was tested based on a withheld test dataset, which has not been used in any of the previous steps.

The concept of a possible **production environment** (C) was developed for the deployment of the data pipeline and trained models:

- (C1.1) Model deployment: many machine learning and deep learning frameworks have their own model deployment functionalities. For the deployment, the trained model parameters or weights are stored in a central and accessible location, e.g. company server. An application running on the server is responsible for loading the weights and providing an API for users or other applications to send and receive data from the model.

2.2 Machine learning approach

For this project, we choose the Graph Neural Network approach. The approach and the reason for choosing it is presented in more detail in the following section.

2.2.1 Graph Neural Networks (GNN)

A graph is a data structure that allows representation of a wide variety of different and complex systems that consist of different objects and their paired connections. Common systems that can be represented by a graph amongst others are social networks, telecommunication networks, molecules, and protein structures [27]. This pattern of connecting objects that influence each other was the reason for choosing the graph data structure to also represent a wind farm for modelling wake interactions.

A graph, also called a network, is constituted of a set nodes ***v*** and edges ***e***, see Figure 5. Within a wind farm, the nodes are the individual turbines, which can be described by the available SCADA measurements or other variables depending on the task and goal at hand. The nodes are connected by edges based on user defined criteria. Within a wind farm these criteria might be the relative distance and angle between turbines. The closer the turbines to each other, the stronger the influence on the aerodynamic performance on the respective turbines. This influence can be represented in the edges by adding, for example, simple weight factors, physical equations, overlap measures based on wind direction, distances, and relative angles.

In order to determine or “learn” the strength of these connections between turbines based on measurement data, graph representation learning is employed [27]–[30]. The purpose of graph representation learning is to encode or map the structural information of graphs into a lower or higher dimensional feature space that represents that graph. These lower or higher dimensional features can then be used for downstream machine learning tasks. This can be done with techniques that fall into the following categories. Graph signal processing-based methods, matrix factorisation methods, random walk-based methods and deep learning based methods [28]. One of the deep learning-based methods are GNNs, which combine recursive neural networks (RNNs) and Markov Chains for the use on graph structures and allow for node-level applications [31], [32].

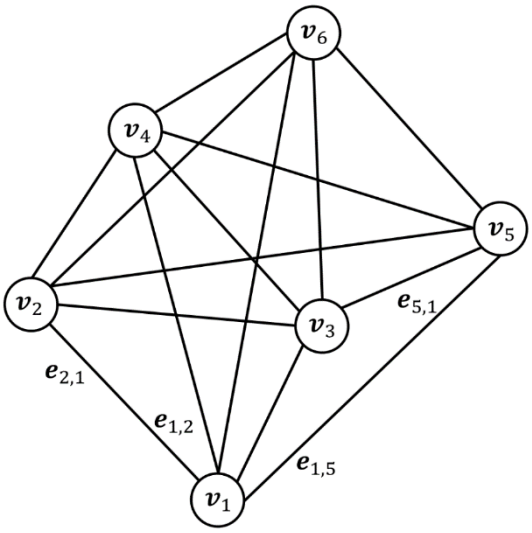


Figure 5. Graph comprised of nodes v and edges e (Source: [22])

For the use cases within this work, GNNs can be used to predict the power output of turbines in a whole wind farm, by considering the various interactions between adjacent turbines, which influence each other due to wakes. The advantage of GNNs compared to other common machine learning architectures like Support Vector Machines, Random Forest Trees and more is that relations between neighbouring turbines can be easily considered based on the graph of the wind farm [33]. In recent years GNNs have been applied for wind speed forecasts and predictions [34], [35], power predictions [23], [36], [37] and interaction loss estimations [23], [37], [38] in wind farms.

The very first task when setting up a GNN is to represent the wind farm in an appropriate graph structure. Owing to the time-varying dynamics of a wind farm, i.e., changing wind speeds and directions, for each measurement point a new graph must be generated, breaking or creating new links between turbines. The large list of generated graphs can then be used for the GNN learning step. The specific generation of the list and the learning step depend on the used GNN framework. At this point we would like to introduce some of the most common GNN frameworks and justify the choice of framework for this project. This will also help understand the various choices made while building a GNN, which we will expand on in later sections.

To create and train a GNN of a wind farm a GNN library written in Python was used. There are currently three popular packages:

- PyTorch Geometric
- Deep Graph Library (DGL)
- Graph Nets library

For the purpose of this project the **PyTorch Geometric** was chosen, which is based on the popular PyTorch library, an open-source machine learning framework. It is easy to use and is a very common choice in academia. One of the biggest advantages compared to the other two options is the availability of the latest models and methods developed by researchers all over the world. In PyTorch Geometric the container that holds the generated graphs of the wind farm is called Dataset, which is a Python class.

You et al. [39] identify a GNN design space and its evaluation methodology for a specific task. The GNN design space consists of three components, the intra-layer, the inter-layer and the learning configuration, see Figure 6.

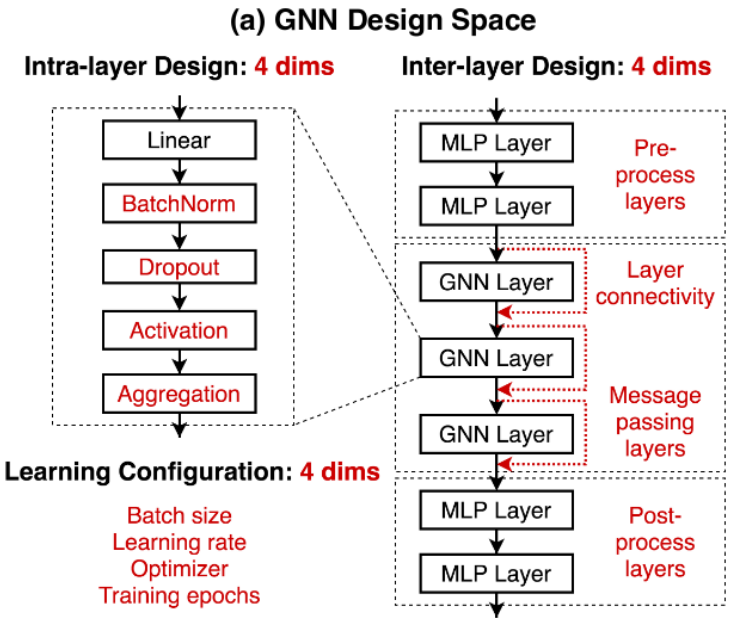


Figure 6. The GNN Design Space by [39]

Intra-layer

Also referred to as GNN layer and is most commonly the subject and focus in GNN research. You (2021) define the GNN layer by

$$\mathbf{h}_v^{(k+1)} = \text{AGG}\left(\left\{\text{ACT}\left(\text{DROPOUT}\left(\text{BN}(\mathbf{W}^{(k)}\mathbf{h}_u^{(k)} + \mathbf{b}^{(k)})\right)\right), u \in \mathcal{N}(v)\right\}\right)$$

with the weight matrix, **W**, the feature vector, h_u,for node, u. Some options for the various functions are given in Table 1.

Table 1. Various options for the GNN layer function

Batch Normalization (BN)	Dropout	Activation (ACT)	Aggregation (AGG)
True, False	0.3, 0.6	ReLU, PReLU, SWISH	MEAN, MAX, SUM

Other popular definitions can be found in Hamilton [27] and Battaglia et al. [40].

However, the most convenient and clear definition that also helps to showcase how GNNs can be applied to windfarms has been found to be by Veličković [41], [42]

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} c_{vu}\psi(\mathbf{x}_v)\right)$$

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} a(\mathbf{x}_u, \mathbf{x}_v)\psi(\mathbf{x}_v)\right)$$

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\right)$$

Here, Φ and Ψ are neural networks. The three definitions show the least to most potent/complex formulation of a GNN layer.

The first definition has resemblance to convolutional neural networks, and hence was termed convolutional GNN, which has fixed edge weights, c_{uv}. Examples can be found in [43], [44].

The second definition was termed attentional GNNs and uses the concept of self-attention, which is used to determine attention coefficients, $a(x_u, x_v)$, which place a certain amount of emphasis on each neighbour's features. The higher the attention coefficients, the more important the neighbouring node. Examples of attentional GNNs are GAT [45] and GATv2 [46].

The last definition is also referred to as message-passing GNN. In fact, the first two definitions are special cases of message-passing GNNs. Veličković [47] even goes so far as to claim that all GNNs developed to date are a variant of message-passing GNNs.

GNN layer research is an active field of research and new models are being developed on a regular basis. At the time of writing, the PyTorch Geometric website[5] is listing 66 different available GNN layers. This makes the choice of finding the right layer time consuming and difficult. Within this work we therefore decided to use the GAT and GATv2 layer, as those have been used already for wind farm modelling [23] and showed promising results. By fixing the GNN layer, this tremendously reduces the searchable design space configurations, and the focus was set to find the best inter-layer design and learning configuration.

Inter-layer

The inter-layer defines how many different intra-layers are used and how they are connected to each other. For example, it is possible to have some multi-layer perceptron (MLP) layers before and after several GNN layers. Another important aspect is how the outputs of one layer is used as the input for the next layer, especially when using the outputs from an MLP as inputs for a GNN.

Learning configuration

As is the case for conventional neural networks, choices have to be made in terms of batch sizes, learning rate, weight decay, the number of training epochs and the optimiser for loss minimisation.

Based on these considerations, You et al. [39] developed a methodology for determining the various parameters for a given task. This resulted in a Python library, called `GraphGym`, which was later integrated into PyTorch Geometric, and hence, can be conveniently used for our use case as well.

[5] https://pytorch-geometric.readthedocs.io/en/latest/cheatsheet/gnn_cheatsheet.html

3 Results and discussion

In this chapter, the results related to the five objectives from Section 1.3 are presented.

3.1 Data pipeline

In data science and machine learning projects a large part of the time and effort spent is on data handling and preparation [48]. A streamlined, automated, and scalable data pipeline greatly reduces the work required by data scientist, analysts and engineers that are trying to get insights from the data and build models. Therefore, part of this project is to develop a framework for an open-source data pipeline that can be deployed anywhere, by anyone.

The developed framework is shown in Figure 7. The first step is Data Ingestion, i.e., the process of importing all available data from a large variation of data sources into a data warehouse. For the ingestion a framework such as Apache Nifi can be used and that can be more user friendly. Scripting is another option, which requires more experienced users, but also offer a plethora of options for automatisation and optimisation. For use case 1 the scripting route was developed, based on the data processing module SparkSQL, whereas for use case 2 Apache Nifi was the project partner's preferred option.

All data is stored in a Spark Data Warehouse and processed based on Apache Spark, an open-source data processing engine that is highly scalable. SQL queries by a client, e.g., data analyst or another software, are executed via the Spark Thrift Server. Within this project the client is "dbt", a framework for data transformation, which allows the transformation of data directly in the warehouse. The result are specific datasets for each use case, that can then be used to gain insights and build machine learning models.

The data transformation with dbt is split into various atomic transformation steps and a final merging into one large dataset. Each transformation step can be created based on SQL or Python. A documentation option allows for the visualisation of every single step and its connection to the other steps in a directed acyclic graph, as shown in Figure 8 for use case 1.

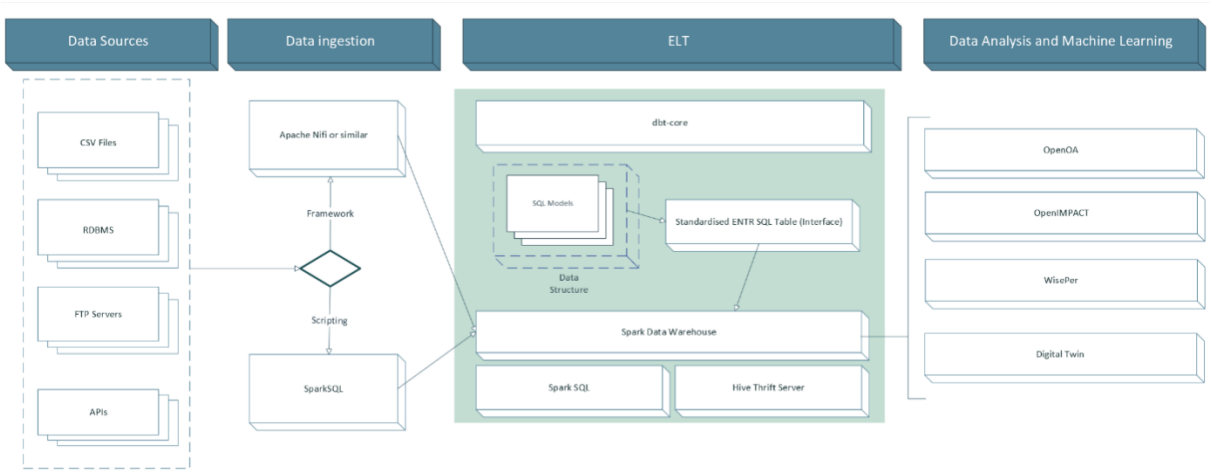


Figure 7. Data pipeline framework

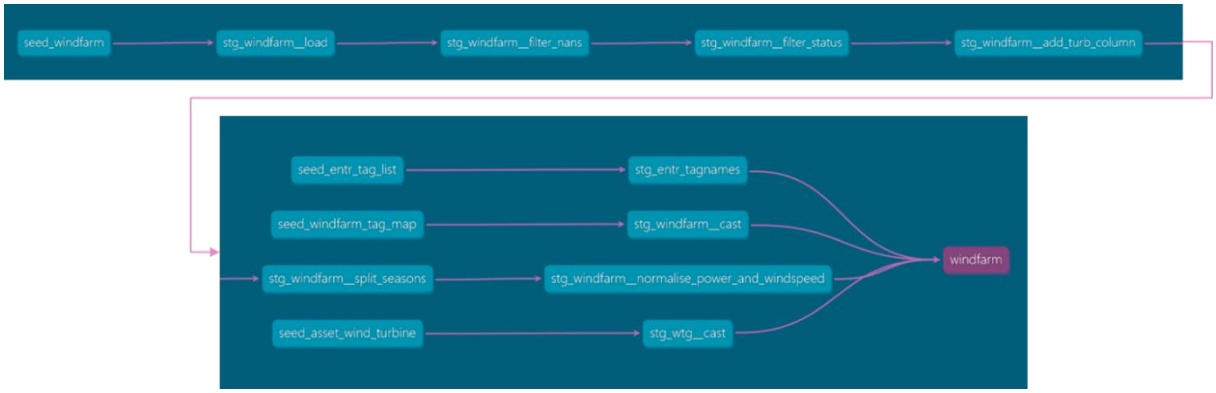


Figure 8. dbt's directed acyclic graph showing the relations between the various data transformation steps for use case 1.

A standard “dbt” project structure is depicted in Figure 9. The data processing and transformation steps are handled by models, which are further split into the categories “marts” and “staging”. The staging models build the core for the data cleaning, type casting, name mapping and filtering process. The marts models bring the processed datasets together to coherent and intelligible data structures serving specific research questions or business use cases that researchers and data analysts are trying to tackle.

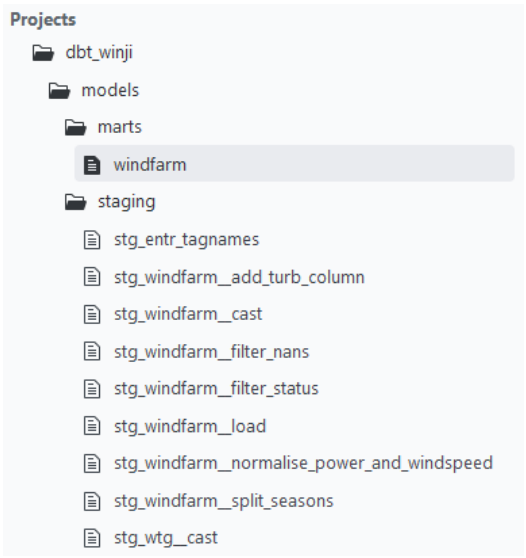


Figure 9. Standard dbt project structure

Another very important aspect of the data pipeline is the transformation of raw data and its naming conventions into a standardised format with standard and widely agreed upon terminology. This is also one of the main goals of IEA Wind Task 43 [49]. For this reason, we work together in this project with the ENTR Alliance and National Renewable Energy Laboratory (NREL), who have developed a standardised data storage format and naming convention, which is based on the standard IEC 61400-25. These standardised datasets then serve as the basis for building tools and models by data analysts and model developers. Within this project the standardised datasets are created by the “windfarm” mart model, see Figure 9, with the structure shown in Figure 10. The data is stored in the long data format, which is more convenient and flexible in terms of data storage itself, when, e.g., new features are introduced or in case of large amounts of missing data within a specific data source.

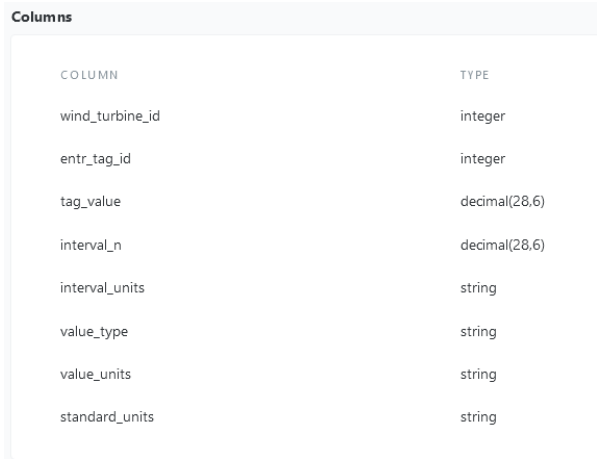


Figure 10. Standardised dataset created by the "windfarm" mart model.

Furthermore, a detailed setup guide for the framework was developed as part of this project and can be found online[6]. Code examples for the data pipeline based on use case 3 can be found in the OpenIMPACT repository[7].

3.2 Data-driven methods for predicting wake interactions (use case 1)

In this use case we focus on a wind farm that suffers from performance losses due to wakes. For this, we will first explain the data pre-processing and cleaning step. After that we show the results of our exploratory data analysis and the choice of important features for the model. Then, the results of a simple graph model will be presented. Lastly, we show how the GNN was optimised, trained and evaluated in terms of wake-interaction effects.

An important point to note is that the provided data is confidential. For this reason, we decided, in agreement with the project partner, to only present the results for a cluster of six out of the 50 WTGs in the wind farm.

3.2.1 Data pre-processing / cleaning

For the data provided in use case 1, we applied extensive pre-processing to remove data outlier in two steps. For all turbines, data was removed according to the following criteria:

Step 1:

- Periods with shutdowns and throttled performance, which were flagged by the project partner.
- With power less than 10% of its rated power [50]
- With wind speed larger than 1.5 times the wind speed at 85 % of rated power [3] (IEC 61400-12 Section "Data correction")
- With power outside cut-in and cut-out wind speed

Step 2:

- With power being outside 10 to 90 % quantile range per wind speed bin (0.5 % bins)

[6] https://weid-ost.github.io/openimpact/getting_started.html
[7] https://github.com/weid-ost/openimpact

For the example turbine shown in Figure 11, we removed about 54% during step 1 and about 44% more during step 2. In total about 74% of the original turbine data was removed during steps 1 and 2.

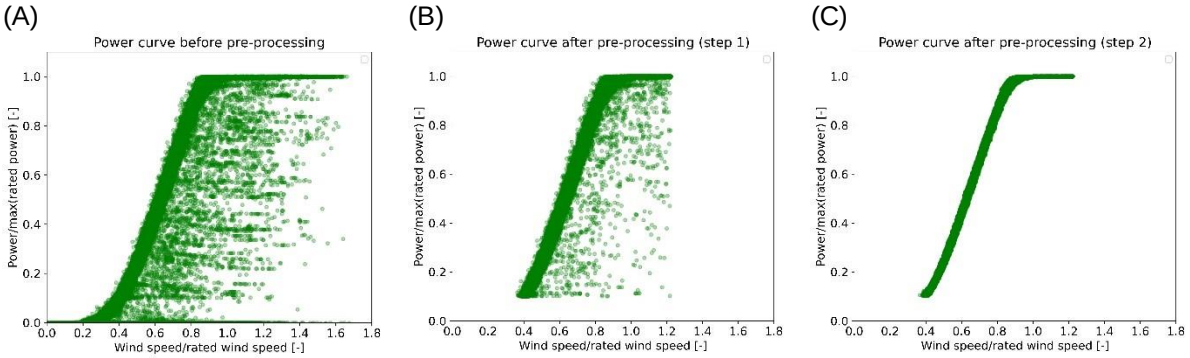


Figure 11. The impact of data pre-processing is shown for one turbine of the use case 1: (A): with outliers, (B) without outliers (step 1) and (C) without outliers (step 2).

3.2.2 Data analysis

Wind roses

We analysed predominant wind speed and direction by partitioning the year in its four seasons and by selecting a WTG from the wind farm for which we assume it is less influenced by the others (highest elevation of the wind farm). Following this procedure wind comes predominantly from directions between 120° to 180° degree, i.e., Southeast to Southerly wind directions with maximum wind speed values up to 1.3 m/s (Figure 12). During spring and summer, winds came more from Southeast (180° to 150°, see Figure 12(B) and Figure 12(C)) while during autumn and winter winds came from a wider wind interval (180° to 120°, see Figure 12(A) and Figure 12(D)). Wind speeds were larger in spring and summer than in autumn and winter, though largest in spring and lowest in autumn.

Wake effect

According to Ding [6] wind turbine wake effects can be best understood through a Δpower-wind direction plot between neighbouring wind turbines (with Δpower being power from a turbine one – power from a turbine two). The larger Δpower the larger the wake effect for the pair of wind turbines from that wind direction. Figure 13 shows Δpower as a function of wind direction and Δwind speed for one pair of wind turbines of the wind farm. For this specific neighbouring wind turbines, wake effects clearly exist when the wind comes from Southeast to Southwest. Another, though weaker, wake effect appears when the wind comes from Northeast. The larger the wind speed differences of the turbines (Δwind speed) the larger Δpower.

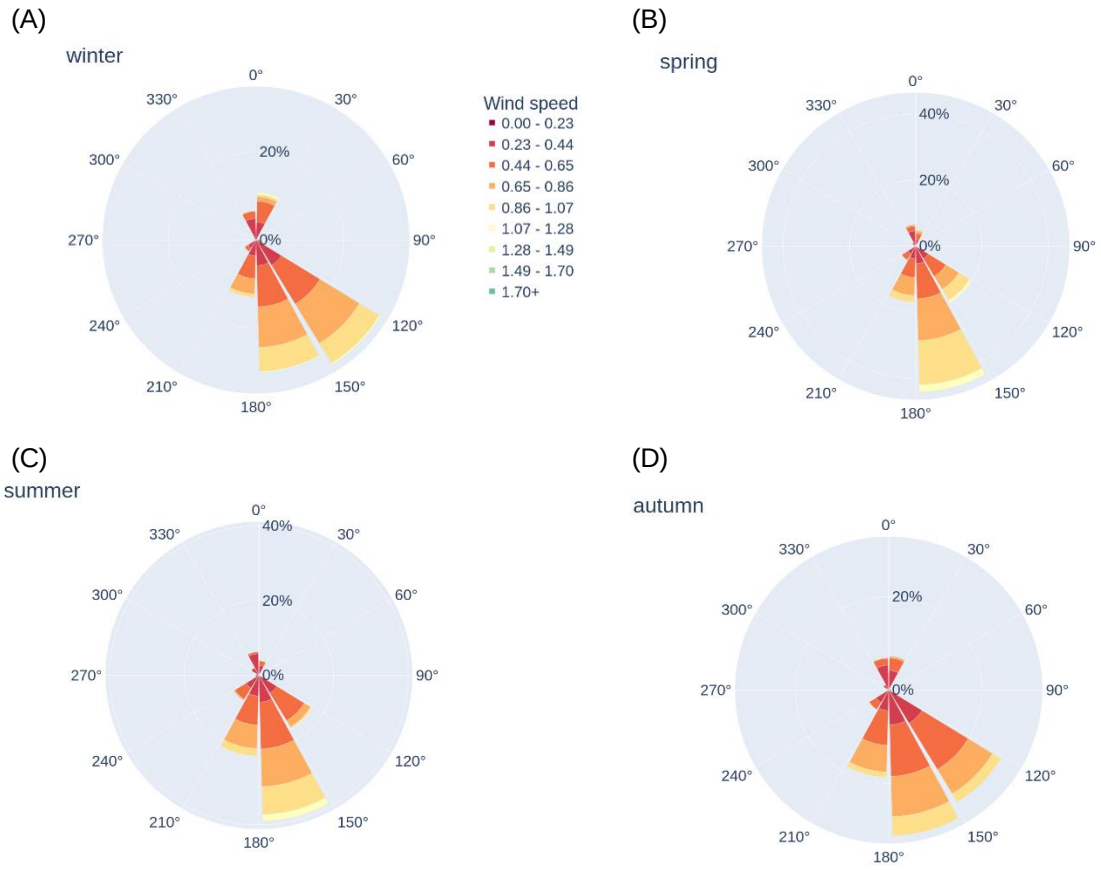


Figure 12. Seasonal wind roses derived for the highest WTG in the wind farm (WTG 13 is the turbine to the lower right in Figure 18: (A): for winter, (B) for spring, (C) for summer and (D) for autumn.

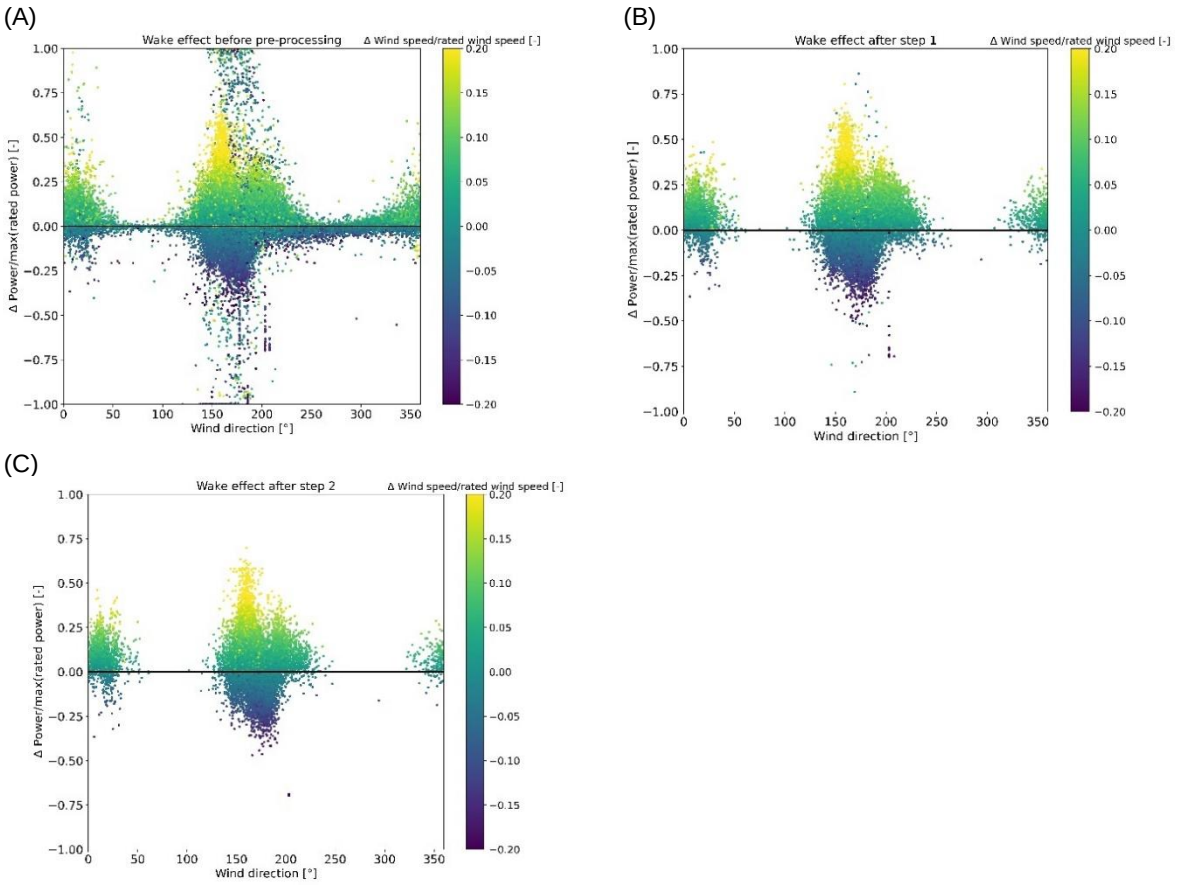


Figure 13. The wake impact for one pair of neighboring wind turbines is shown. (A) to (C) show the results when data pre-processing is included: (A): with outliers, (B) without outliers following step 1 and (C) without outliers following step 2.

In addition to demonstrating the wind turbine wake effect, with Figure 13 we also show the impact that data pre-processing had. The data set for one pair of neighbouring wind turbines cleaned from outliers as in Figure 12(C) allows to see the wake impact more clearly than in Figure 12(A) where outliers disturb the wind turbine data set. Note that the data sets shown in Figure 12(A) to (C) were pre-processed as for the power curves shown in Figure 10.

Terrain

A map of terrain elevations for the wind farm, the digital elevation model (DEM), was extracted from the free Shuttle Radar Topography Mission (SRTM) data set in 30 m horizontal resolution[2]. The given vertical accuracy worldwide is between 2 to 9 m. We used the EarthExplorer[3] to download the DEM for the target geographical region from the SRTM data set.

Figure 14 shows a cut-out of the reprojected SRTM DEM with the wind farm turbines marked by crosses. The higher terrain elevations Southeast to East are likely to have an impact on the overall wind flow patterns in the wind farm given that the predominant wind direction is Southeast, and the mountains are less than 1 km away. Among all wind turbine locations, i.e., the entire wind farm, terrain elevation differences reach up to 161 m with a standard deviation of 57 m. Terrain slope angles at wind turbine locations vary between 0 ° to 13 °.

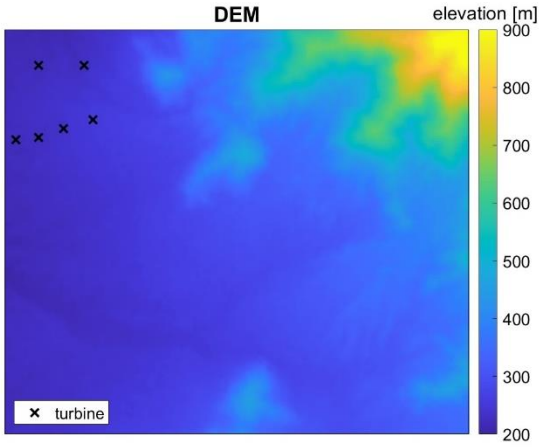


Figure 14. Some of the wind turbines (black crosses) of the wind farm (cutout) are shown on the digital elevation model (DEM) in 30 m horizontal resolution. The DEM was extracted from the free SRTM data set.

3.2.3 Feature engineering

In this section we present the features that will be used for the graph and GNN models. These are

- Free stream conditions, derived from measurement data
- Yaw misalignment, which is directly measured
- Distance and azimuthal matrices, derived from longitudes and latitudes
- Terrain parameters, derived from a digital elevation model

Free stream conditions

So-called free stream conditions describe the meteorological conditions as measured by a weather station (met mast) close by but undisturbed by surrounded wind turbines, terrain, settlements, or forest. We need these free stream conditions for wind speed and wind direction to compute the turbine wake effects within the wind farm, i.e., for the GNN models. Since such an undisturbed measurement was not available here, we must derive these conditions from the most undisturbed wind turbine in the wind farm. Here, we select such an undisturbed wind turbine from DEM characteristics of all turbines in the wind farm. Note that we add the turbine height to the terrain elevation to consider this additional height above the ground.

To find the turbine location with approximate free stream conditions we exploit the Sx parameter [48]. The Sx parameter is commonly used in studies dealing with complex topography. A negative Sx parameter indicates that a grid cell is “exposed” relative to the surrounding terrain. A positive Sx indicates that a grid cell is “sheltered” by the surrounding terrain. The smaller/larger the Sx parameter the more exposure/sheltering prevails.

We derived the Sx parameter for all grid cells (and thus including the turbines) by looking in a fixed direction, e.g., the predominant exposed wind direction of a wind farm (directional Sx) and we further derived spatial mean Sx parameter for all grid cells. For the spatial mean Sx we first calculated Sx in all angular intervals of 30° around each grid cell and then averaged the resulting angular values. The maximum horizontal distance for evaluating all Sx parameters was set to 1000 m.

In Figure 15 we show the same cut-out of the wind farm as presented for the DEM in Figure 14 but this time for the Sx parameters of all grid cells: (A) the spatial mean exposure index Sx and (B) Sx derived for the predominant wind direction of the wind farm (here southeast 165°±15°). The larger positive Sx

values behind each turbine (Figure 15(B)) indicate the spatial range up to which a wake calculated from the directional Sx parameter shows an influence.

We also show the most exposed turbines according to the corresponding Sx values in Figure 15 (circles). The most exposed turbine marks the turbine location with approximate free stream conditions. For demonstration and confidentiality reasons we derived the most exposed turbines from the Sx parameters of the 6 wind turbines shown in Figure 15 and not from all turbines of the entire wind farm. Predominant wind directions for the turbine with approximate free stream conditions according to spatial mean Sx parameters (Figure 15(A)) were from Southwest to Southeast (Figure 16). Free stream wind directions are thus more balanced around South directions than for the highest WTG where winds came predominantly from South to Southeast (Figure 11). Spatial mean Sx-value variations varied for the 6 turbine locations shown Figure 15 and the 12 angular intervals between 0.3° and 0.5° and between 0.6° and 1.1° for all turbines in the wind park. Spatial mean Sx values for the 6 turbines reach values around -52°, i.e., much larger values than their surroundings (cf. colorbar scale in Figure 15(A)).

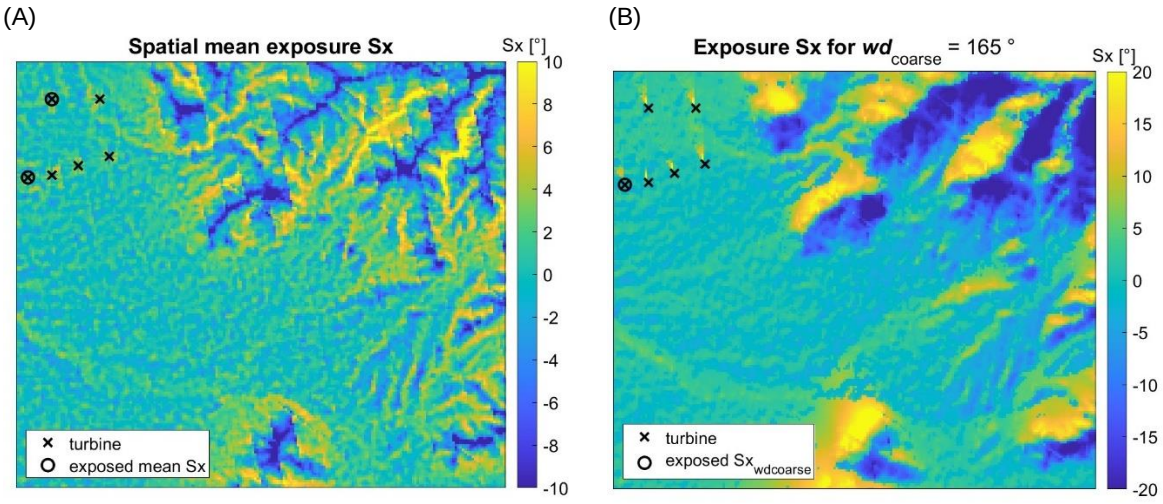


Figure 15. Exposure angle Sx for a cutout of the wind farm in 30 m horizontal resolution, calculated on the SRTM-DEM. (A): Spatial mean exposure angle Sx and (B): Directional Sx values in a fixed wind direction (here southeast 165°±15°). Locations of turbines are indicated with black crosses. The circles indicate the most exposed turbine according to the corresponding Sx exposure angles.

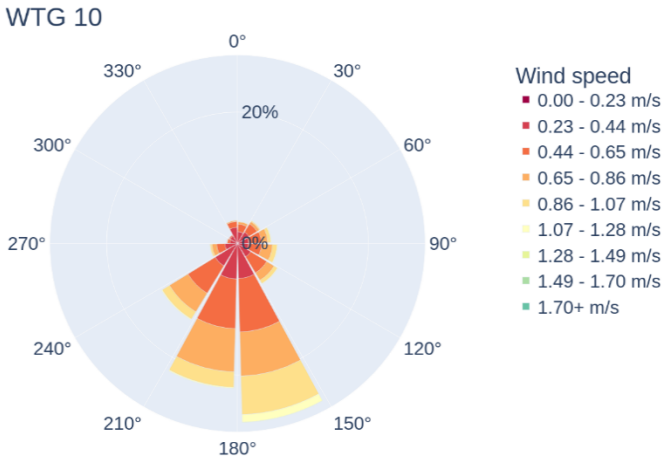


Figure 16. Wind rose of WTG 10 for the entire year. WTG 10 is the most exposed turbine based on the spatial mean Sx parameters of the six turbines shown in Figure 15.

Yaw misalignment

In order for a WTG to extract as much energy out of the incoming wind it is paramount that the rotor surface and the wind direction are aligned, i.e., the yaw misalignment or yaw error is at a minimum. It is therefore important to include this feature when modelling power curves for WTG to account for this effect [51].

There is another important reason to consider yaw misalignment as a feature, as already detailed in section 1.1. The direction of the wake behind a WTG is influenced by the yaw misalignment. Hence, setting a bias to the default yaw angle of a specific WTG, and thus steering the wake into a slightly different direction, has the potential to increase the possible power output of downstream turbines and in turn results in a positive net power increase of the whole wind farm.

Distance and azimuthal matrix

The layout of the wind farm can be translated into a graph data structure by calculating the relative distances and angles between each turbine pair in a wind farm. Given that wind farms span many kilometres in size, the curvature of the earth must be accounted for in the distance calculation. For this the Haversine function was used, which takes the position of each WTG in the longitude and latitude format as inputs. The resulting distance matrix is shown in Figure 17(A).
For the relative angles the azimuth angle formed for each turbine pair was determined, with 0° pointing in the north direction and 90° pointing towards east. The resulting azimuthal matrix is shown in Figure 17(B).

Both features are either used directly for the GNN model or to derive new features, such as the measure of overlap between a turbine pair, where the distance, the azimuth as well as the wind direction are used. This last step is still work in progress and will be presented in more detail in the final report of this project.

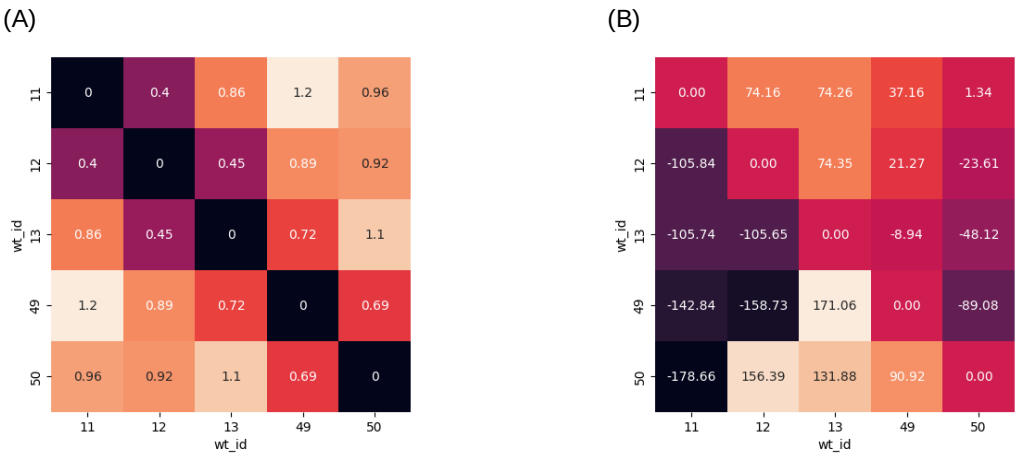


Figure 17. Distance matrix (A) and azimuthal matrix (B) used for translating the wind farm layout into a graph data structure.

Terrain parameters

Hilly or mountainous terrain alters the wind flow. Since the wind farm in this use case is close by to mountainous terrain (cf. Figure 14), we include terrain parameters as features for the weights in the distance/azimuthal matrix. One option is the Sx parameter which nicely explains exposure and sheltering with regards to terrain, forest, and wind turbines in the surroundings (cf. Figure 18). The Sx parameter can be used with (Figure 18(A)) or without considering predominant wind directions of a wind farm (Figure 18(B)).

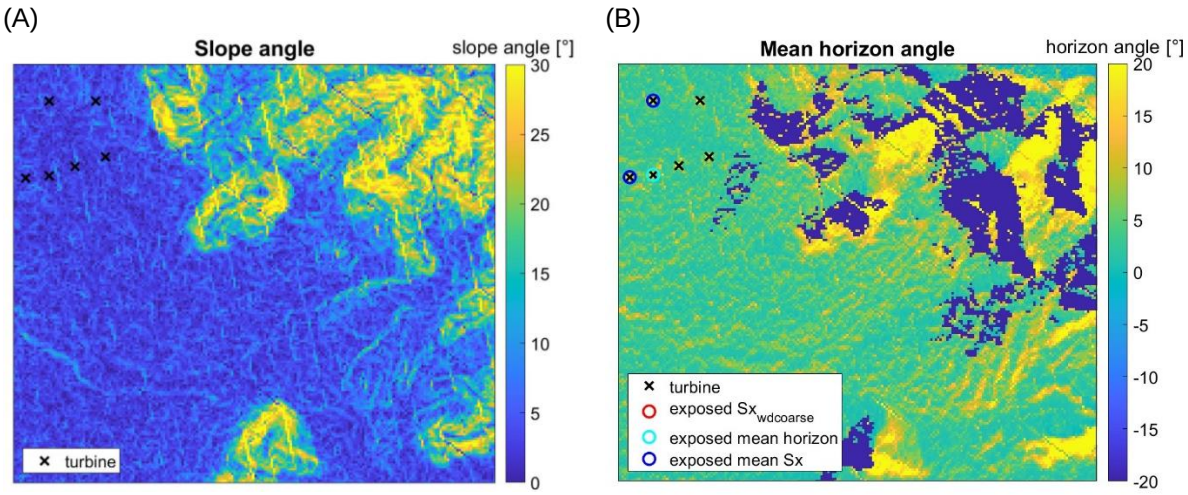


Figure 18. (A): Spatial slope angles and (B): Mean terrain horizon angles. The colored circles indicate the most exposed turbine according to the corresponding approach. The red and blue circles overlap for the lower left turbine.

3.2.4 Graph Model of WTG cluster

A first and simplified graph model was built and trained that serves as a precursor to the final GNN. The graph model and its results were presented at the Wake Conference 2023 in Uppsala, Sweden, as well as published in the Journal of Physics: Conference Series, with the title “Graph machine learning for predicting wake interaction losses based on SCADA data“. In what follows is a short description of the approach and some results. The interested reader is encouraged to consult the paper for further details.

The very first step was the translation of the wind farm layout into a graph data structure, which will finally also be used for the GNN. This was done based on the *distance and azimuthal matrices* together with the python library NetworkX[8]. The final bi-directional graph is shown in Figure 19.

In the next step an "ideal" power curve model according to Figure 20 was built. The ideal power curve model predicts the power output for a wake free WTG and uses the *free stream wind speed and direction*, as well as the *yaw misalignment* of each WTG as features. The XGBoost[9] python library was used for the model creation.

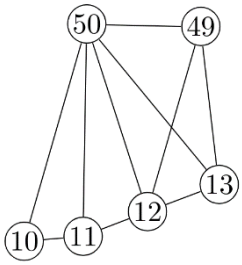


Figure 19. Network or graph of the five WTG cluster of the 50 WTG wind farm.

Figure 20. "Ideal" power curve model based on the free stream wind speed wsg, the free stream direction wdg and the yaw misalignment n_{di} of each WTG.

In a next step the difference between the "ideal" power, $\boldsymbol{p_i}$, predicted by the power curve model and the measured power output, $\boldsymbol{p_{m,i}}$, for each measurement point was calculated. The goal was then to represent these power differences in the graph through weighting factors and attribute them to interaction losses between a pair of turbines. The weighting factors are determined by solving the following minimisation problem

$$min\,||P - \langle 1\,|G|\,p\rangle||_2^2 \qquad \forall t \in T$$

with the vector $\boldsymbol{P}$ containing the measured power outputs, $p_{m,i}$, of each turbine for one point in time, t, the vector, $\boldsymbol{p}$, containing the predicted "ideal" power, p_i, and the adjacency matrix, $\boldsymbol{G}$, which represents the graph data structure. An example of one possible adjacency matrix, $\boldsymbol{G}$, is

[8] https://networkx.org/

[9] https://xgboost.readthedocs.io/en/stable/

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

where each row and each column belong to one WTG and indicating whether two turbines are connected or not. An initial value of one denotes that a turbine pair is connected in the graph and a value of zero denotes no connection. By solving the minimisation problem, the initial values of one are replaced by weighting factors, which indicate how strongly one turbine, or rather the potential wake of a turbine, affected the power output of a downstream turbine. The python library cvxpy[6] was used for solving the minimisation problem.

This process is illustrated in Figure 21. The result was optimised adjacency matrices, $\boldsymbol{G_k}$, for each measurement time, *t*.

Figure 21. The resulting optimised adjacency matrices.

Lastly, an adjacency matrix model (Figure 22) was built that takes the free stream conditions and the yaw misalignment of each WTG as an input and the optimised adjacency matrices as a target. This model, built with the XGBoost library, predicts adjacency matrices for given free stream and yaw error conditions, which in turn represent the interaction losses between turbine pairs in terms of weighting factors in the predicted adjacency matrices.

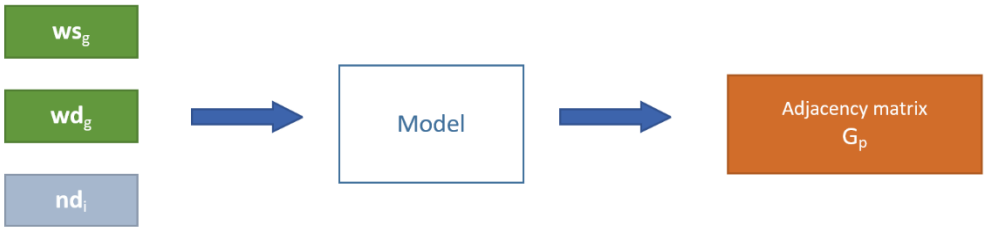


Figure 22. The adjacency matrix model based on the free stream conditions and the yaw misalignment.

The model was then tested to see if it could capture wake interaction losses. For this the test data points that lie within the wake region of WTG 13, as seen by WTG 49, were flagged. All test data points within a wind direction range from 160 - 180° were categorised as ”in wake”, which is roughly ±10° of the angle of alignment between WTG 13 and WTG 49.

The box plots in Figure 23 show the median value, the quartiles as well as outliers of the power output over bins of wind speeds for the measurements (top), the predicted ideal power (middle) and the graph model (bottom). For the data points in the wake region, distinct differences in the measured power output for WTG 49 were observed, which is in line with other studies, e.g. [4], [52]. It can also be seen that the graph model is able to capture, partially, these differences as well. In summary, a simplistic machine learning approach based on graph structured data was able to capture wake interaction losses.

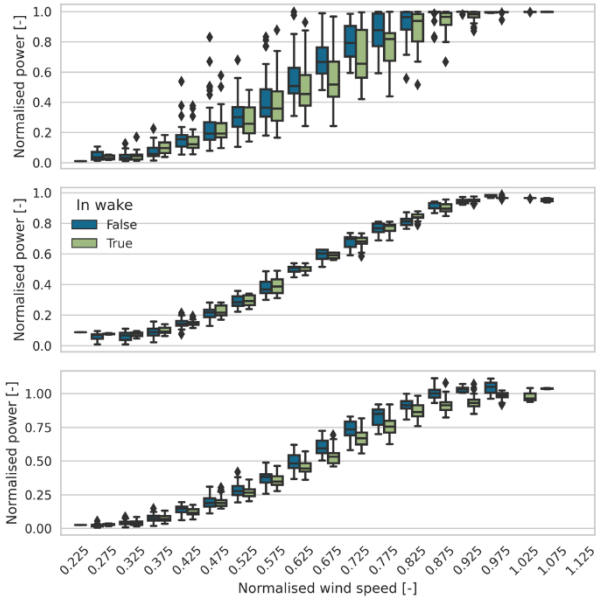


Figure 23. Power curves of WTG 49 based on the measurements (top), the “ideal” power curve model (middle) and the graph model predictions (bottom) for the wake and non-waked regions.

3.2.5 Final GNN

The main drawbacks of the simplified graph model in the previous section are threefold:

1. Free-stream conditions need to be determined in case no met mast data is available.
2. A power curve model needs to be trained first and the predicted power output is then corrected based on the optimised graph. The graph itself only encodes how the power output of one turbine might have an influence on the power output of a neighbouring turbine. It is not possible to use additional features such as environmental conditions, turbine positions and other spatial information.
3. Scalability: Given the constraints in the in the last two points, the approach is not scalable, i.e., using data from much larger wind farms.

These points, however, can be addressed using GNNs instead. In the following, the approach used within work for modelling wake interaction effects between neighbouring turbine is presented.

Before one can use the measurement data to train said GNN, several steps are needed.

1. Cleaning and filtering the raw measurement data
2. Data exploration
3. Feature engineering
4. Choosing features and targets

5. Creating a PyTorch Geometric dataset
    a. Translating the turbine position into graph nodes
    b. Connecting neighbouring turbines by edges depending on their relative distance and alignment
    c. Attaching features and targets to their respective nodes and edges
6. Splitting the dataset into train, validation, and test sets
7. Choosing a GNN design: GNN and MLP layers as well as their connection
8. Determining learning parameters.
9. Training, validating, and repeating the above points.
10. Testing

Points 1 – 4 are the same to the graph model approach. The first difference is the need for a PyTorch Geometric dataset, where the turbine positions need to be translated into nodes, which are then connected by edges depending on their relative distance and alignment. After that each node receives a feature vector and a target vector. The feature vector contains the u and v component of the wind speed, calculated by

$$u = \text{wind speed} \cdot cos(\text{wind direction})$$

$$v = \text{wind speed} \cdot sin(\text{wind direction})$$

and were additionally normalised to have a mean around zero and a standard deviation of one. Furthermore, the yaw error as well as the turbine coordinates were used as features. The GATConv layers also allow for edge features. Here the x and y component of the relative distance between turbines were used.

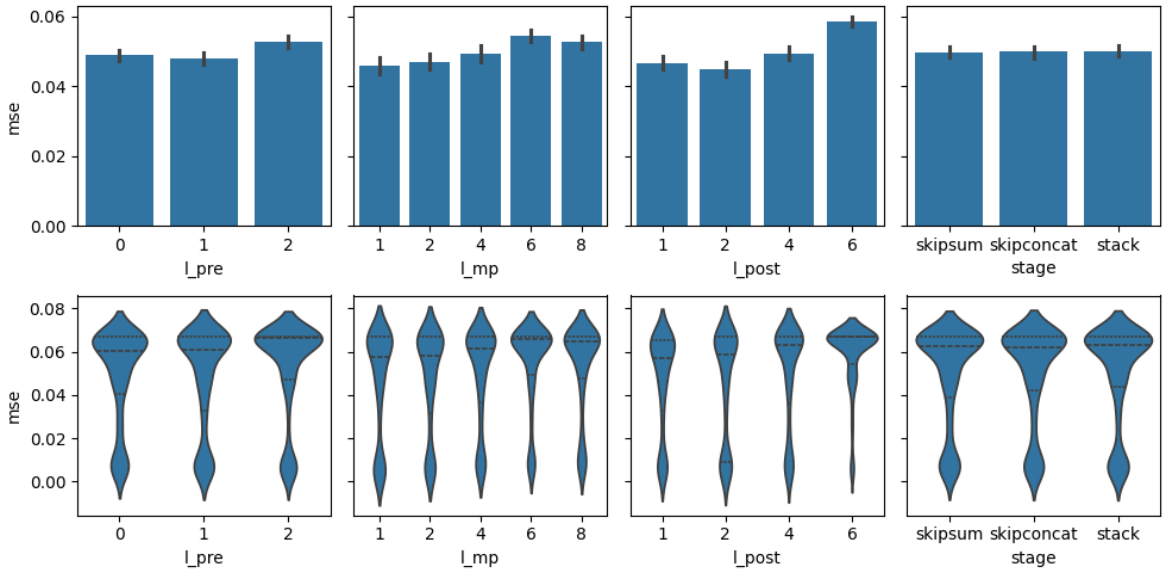
The result is a number of n graphs, one for each measurement point. The edges of all graphs were then additionally filtered depending on the wind direction. If the alignment between two turbines is much different to the underlying wind direction, the edge is destroyed, otherwise is kept. All graphs are then stored into a single PyTorch Geometric dataset. The dataset serves as a convenience container, which can be easily handled for further processing, e.g., splitting the graphs into train and test splits.

The next step was the choice of GNN design. As mentioned in Section X, the GATConv and GATv2Conv layers were chosen due to their recent use for wind farm modelling. In order to find a suitable design, the `GraphGym` library in PyTorch Geometric was used. Nine different design parameters were considered

1. Activation function (act): PrELU, ReLU and Swish
2. Attention heads (att_heads): Key feature of the GAT layer, for stabilising the learning process
3. Inner dimensions (dim_inner): The number of neurons for each layer in the MLP
4. Batch normalization (batchnorm): If true, at each layer in the GNN the inputs to the next layer are normalised. Used to improve speed, performance and stability
5. Batch size (batch_size): The size of the batch for one forward pass through the whole GNN
6. Layer connectivity (stage): Determines how the different layer are connected. "Stack" means that the output of one layer is directly used as the input for the following layer. Skip-sum and skip-cat are skip connections that allow for information transfer from earlier layers to later layers by adding or concatenating the outputs from an earlier layer to the outputs of a later layer, respectively.
7. Pre-processing layers (l_pre): The number of MLP layers before using GNN layers.

8. GNN layers (l_mp): The number of GNN layers.
9. Post-processing layers (l_post): The number of MLP layers after the GNN layers.

This resulted in over 315'000 different parameters combinations. As it is not feasible to test this amount of different combinations, `GraphGym` perform a controlled random search [39], which reduced the number of combinations to train to around 1'900. We were able to run these cases on our High Performance Cluster (HPC). 24 cores were used in parallel and the total run time for all cases was around 5 days. The results of this run are summarised in Figure 24, which shows the mean squared error (top row) and the distribution of the mean squared error (bottom row). What is particularly striking is the positive effect of having batch normalisation. Furthermore, wider MLPs perform better, whereas there is no clear difference for different numbers of attention heads. The combination resulting in the lowest mean squared error is shown in Table 2.



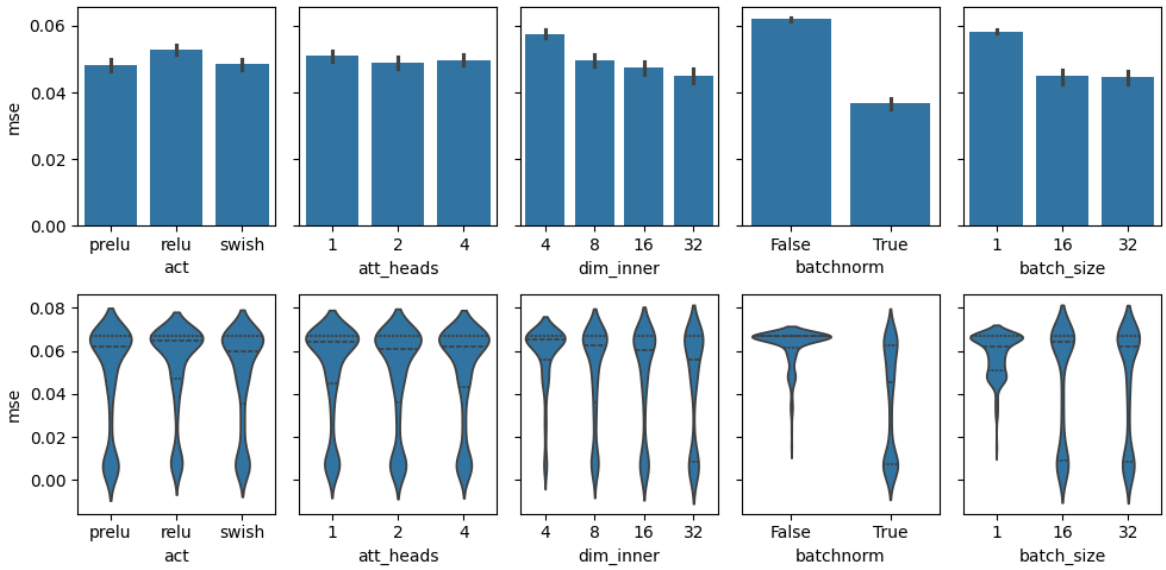


Figure 24. Mean squared errors and distributions of mean squared errors for the different design space dimensions.

Table 2: Parameters resulting in the lowest mean squared error out of 1'900 different combinations.

l_pre	l_mp	l_post	stage	act	att_heads	dim_inner	batchnorm	batch_size
2	1	2	stack	prelu	4	32	True	32

After this step, an additional hyperparameter tuning step was conducted based on Bayesian optimisation (REF). The following parameters were optimised

1. Training epochs (max_epochs)
2. Learning rate (lr)
3. Weight decay
4. Inner dimensions (dim_inner): Number of neurons per MLP layer. As a downward trend for higher dimensions was observed in the previous step, this parameter was further optimised.
5. Number of layers in the MLP (num_layers): Previously, a fixed number of layers was used.
6. Attention heads (att_heads): As this is a key feature of the GAT layer, but yielded no conclusive results in the previous step, it was further optimized.
7. Feature dimensions (h_dim): The output dimensions for the GAT layer
8. Batch size (batch_size): Inconclusive results in the previous step. Batch size also has a large impact on the training speed. For this reason, it was further optimised.

For the optimisation the Python library `scikit-optimize` was used.

In total 100 optimisation steps were carried out, resulting in the convergence plots as shown in Figure 25. The plot shows the minimum of the objective function after n optimisation steps. The objective function is a model of the loss function of the GNN. As can be seen, beyond 45 optimisation steps, no further improvement in loss reduction is yielded. Table 3 shows the parameters resulting in the minimum loss.

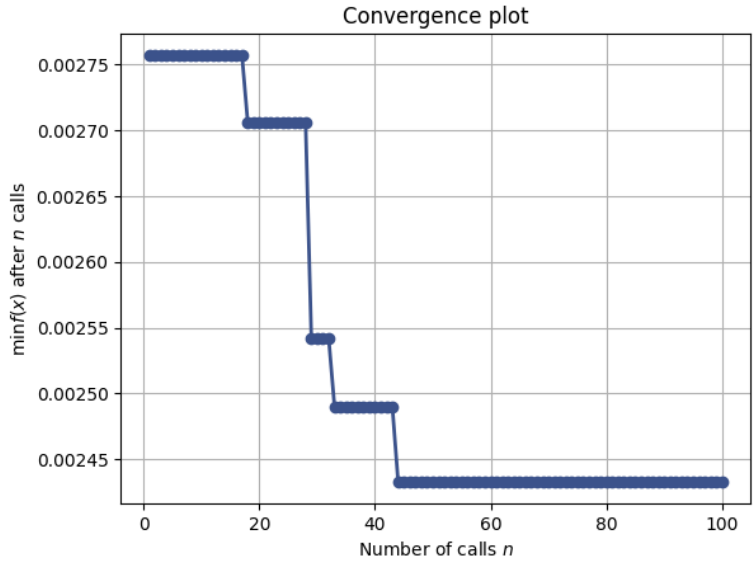


Figure 25. Convergence plot of the minimum of the objective function versus the number of tuning steps.

Table 3. Results of the hyperparameter tuning with the Bayesian optimisation approach

max_epochs	lr	weight_decay	dim_inner	num_layers	att_heads	h_dim	batch_size
800	2.4e-3	1e-5	312	4	1	22	512

Finally, the GNN can be used, and the produced results analysed. As for the graph model, six wind turbines out of the 50 WTG wind farm were used due to confidentiality considerations. Again, only wind direction between 140° to 210° are considered here for show-casing purposes. The GNN, however, was trained for all wind directions and can hence be used for all wind directions. Figure 26 shows the graphs of the wind farm cluster of six WTGs for different wind directions at a given wind speed. Here the above-mentioned process of edge elimination can be observed. As the wind direction angle and the turbine alignment angles diverge from each other, edges are removed from the graph and vice versa. This was done to incorporate some prior engineering knowledge into the model. Wake interaction effects can be ruled out when the wind direction and turbine alignments are not in line with each other.

The numbers at the graph edges are the learned attention weights. The higher the value, the more important the receiver node (downstream turbine) deems the features of the sender node (upstream turbine). An important point to note is that the attention coefficients do not necessarily tell us much about the strength of the wake effect itself but gives us an intuition about whether the model is able to capture significant relationships in the data itself [23]. This can be especially observed when looking at WTGs 13 and 49, as the wind direction changes from 150° to 170° more and more importance is put on the upstream WTG 13 from the perspective of WTG 49.

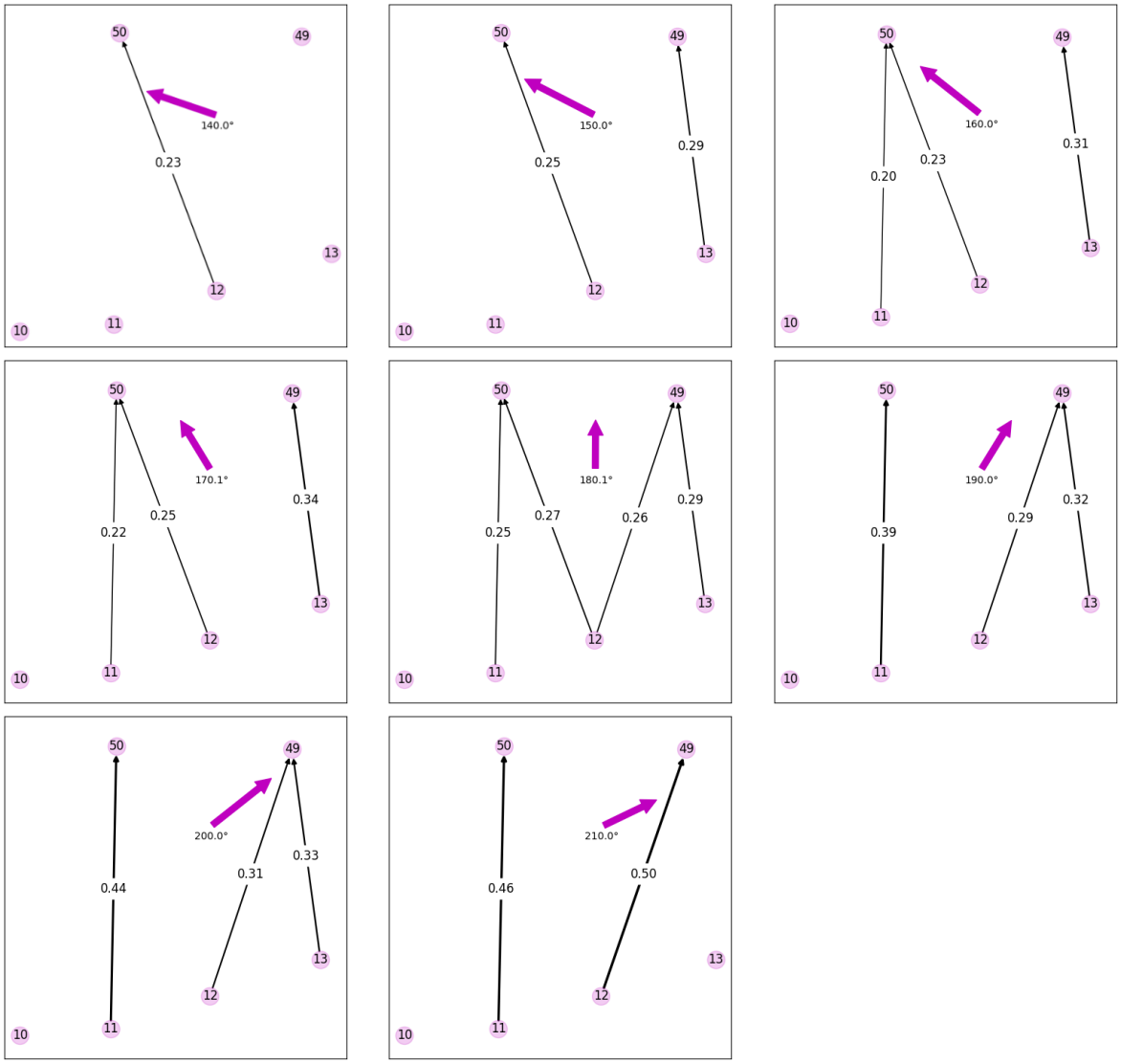


Figure 26. Graph structure of the turbine cluster for different wind directions. The attention coefficients for each egde are given.

We singled out WTG 13 and 49 for further investigation with the GNN model. Figure 27 shows the normalised power difference between both turbines for various wind direction bins of size 10°. The vertical lines denote one standard deviation of power differences for each wind direction bin. The green vertical line shows the turbine alignment angle. The GNN model can successfully predict the power differences for wind directions below 195°. Possible reasons for the differences for wind directions above 195° are currently not known and need further investigation.

For WTG 49 box plots of the power curves for the waked and non-waked case (top) and the respective power differences (bottom) with respect to WTG 13 are shown in Figure 28. Similarly, to the graph model analysis, data points were flagged as "in wake" and "out of wake". On a qualitative level, the GNN model is able to reproduce the behaviour seen in the measurement data. When WTG 49 is waked, it produces slightly less power and the power differences between WTG 13 and WTG 49 noticeably increase.

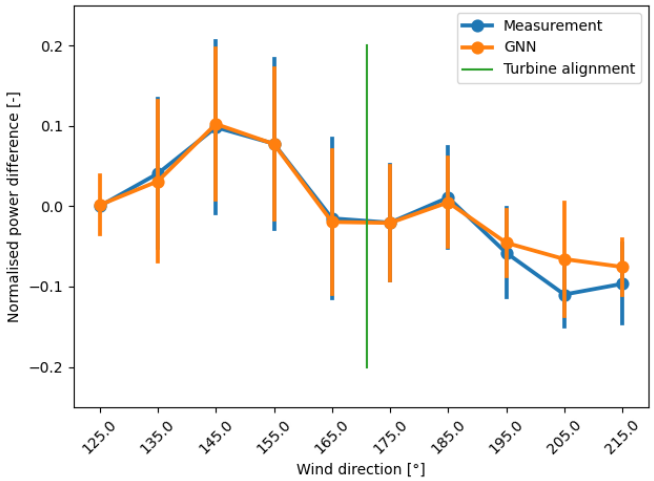


Figure 27. Normalised power difference between WTG 49 and upstream WTG 13 for different wind directions. The green line denotes the turbine alignment angle between both WTGs.

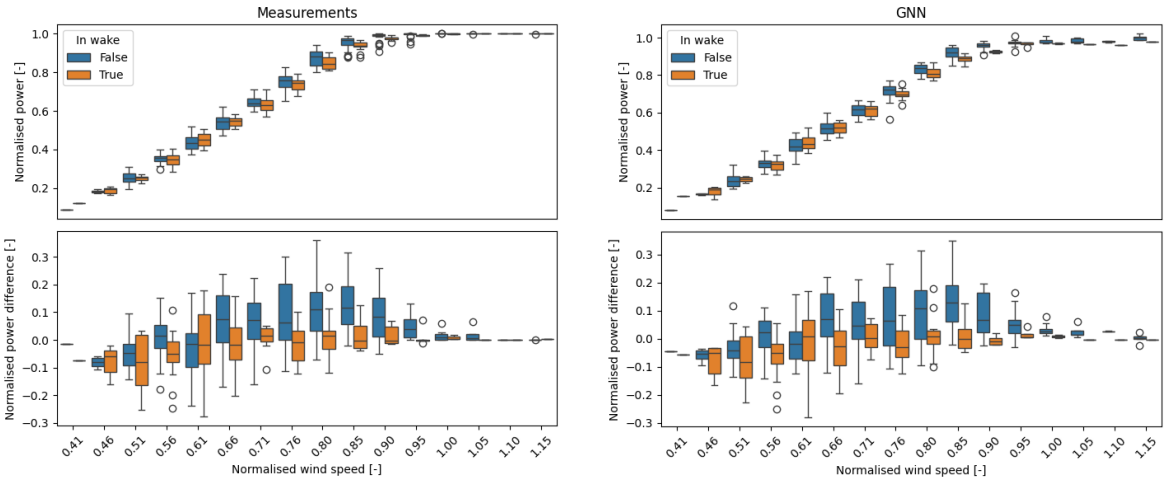


Figure 28. Power curves of WTG 49 for the waked and non-waked case as well as the normalised power differences between WTG 49 and upstream WTG 13.

Looking more closely at the power curve differences between the measurements and the GNN predictions, see Figure 29, the GNN noticeably underpredicts the produced power for a given wind speed for the waked and the non-waked case. To quantify the differences, the normalised energy production for each wind speed bin was calculated for the given time period in the dataset, see Figure 30 (top). For wind speeds around the rated power, the produced energy in the observed period was underpredicted by almost 1%. The absolute error of energy production is shown in Figure 30 (bottom). This resulted in an underestimation of the total energy for the given period of 3.68%. This could be due to two main reasons. First, the amount of data and the provided features in the data are not sufficient for the complexity of the model and second, further optimisation and tuning in terms of GNN design and hyperparameters are needed.

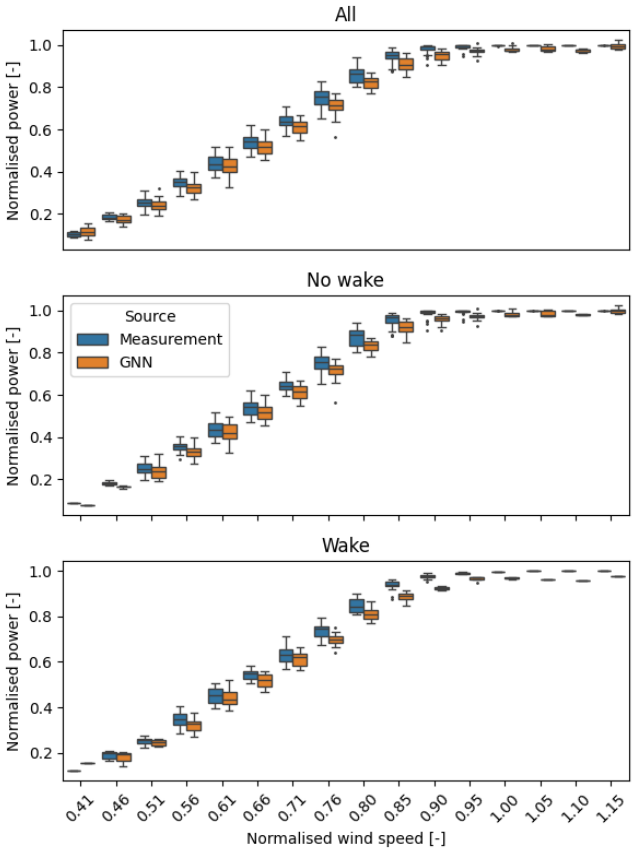


Figure 29. Power curves of WTG 49 based on the GNN predictions and the measurements for all data points (top), the data points outside the wake region (middle) and for the data points in the wake region (bottom).

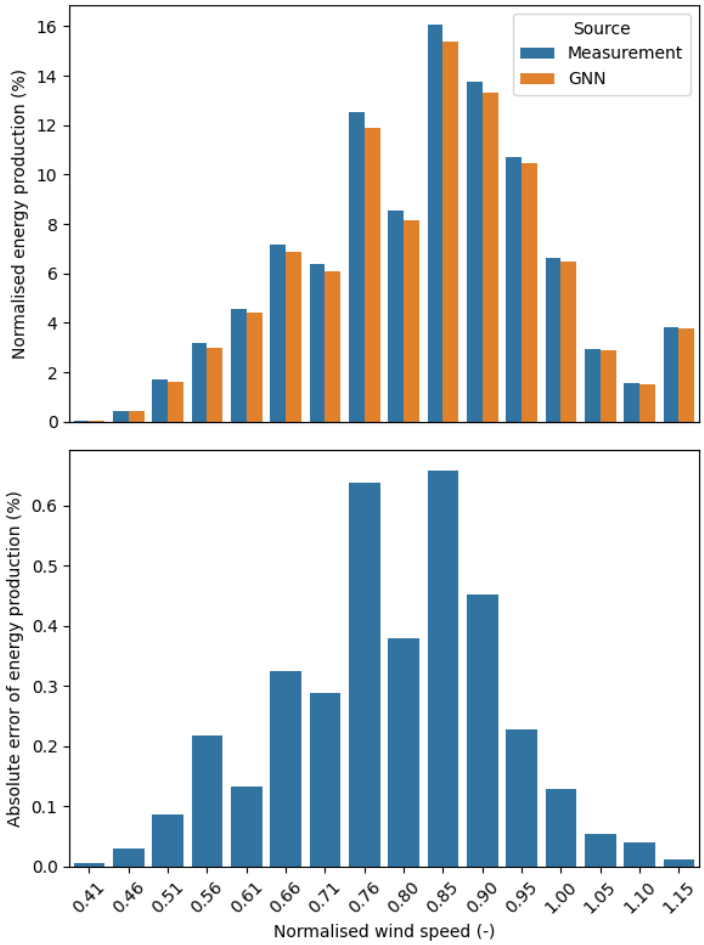


Figure 30. Normalised energy production of the GNN predictions and measurements (top) and absolute error of energy production for the GNN prediction (bottom)

In this use case we have shown the process of developing a graph model and GNN for predicting wake-interaction effects. The simple graph model was able to capture the effect of wakes in the given dataset by relating the turbine power outputs by the power outputs of its connected neighbour turbines. However, the approach is limited in that only one feature – here the power output – can be incorporated to find those relationships.

To leverage important features such as wind speeds, yaw errors and spatial values, such as coordinates, relative distances and angles, a GNN was developed. It was shown how to search for GNN designs and how to optimise GNN hyperparameters through Bayesian optimisation. The GNN model was then able to learn attention coefficients that represent the amount of importance one turbine puts on a neighbouring turbine, while aggregating spatial information (turbine coordinates, relative distances and angles) and features (wind speeds and yaw errors). We showed that the GNN model was able to capture wake interaction effects present in the dataset and predict power curves that show waked and non-waked behaviour. However, the current model underestimated the total energy production by about 4% compared to the measurements. The very next step and focus should be on further model optimisation to improve on this.

Furthermore, GNNs allow for complex architectures and have an immense design space, rendering the development, training, optimisation and interpretation of such models arduous. It is also known that the more complex a model gets, the more data is required for training, referred to as the curse of dimensionality [53]. Hence, high quality data for long periods of time are needed to improve the prediction accuracy of such models. The provided dataset was limited to two years and was lacking important features such as turbulence intensity, which was shown to have a considerable effect on power predictions [54]. Therefore, in future work the development of a GNN could be accompanied by simulation data.

3.3 Data-driven methods for quantifying the effect of performance upgrades (use case 2)

In this use case we dealt with upgrade identification and quantification for WTGs that were retrofitted with vortex generators and Gurney flaps on their blades. Data from 16 turbines in a wind farm for a period from 2012 to 2022 was available.

Firstly, we will show the data pre-processing steps that were carried out, followed by an exploratory analysis. The analysis revealed that after the turbine upgrades were installed, the wind measurements on these turbines were unreliable. Hence, we developed a GNN for data imputation, predicting corrected wind speeds based on neighbouring turbines.

After that we show the results of the turbine upgrade identification and quantification and compare the cases with and without the corrected wind speeds.

3.3.1 Data pre-processing / cleaning

For the data provided in use case 2, we conducted extensive pre-processing to eliminate outliers, with slight differences from how we performed the pre-processing for use case 1. In step 1, we did not have flagged periods from the project partner, and in step 2, we had to adjust the power quantile range used. Subsequently, data from all turbines was removed based on the following criteria:

Step 1:

- With power less than 10% of its rated power [46]
- With wind speed larger than 1.5 times the wind speed at 85 % of rated power [3] *(IEC 61400-12 Section "Data correction")*
- With power outside cut-in and cut-out wind speed

Step 2:

- With power being outside 0.5 to 99.5 % quantile range per wind speed bin (0.5 % bins)

For the example turbine shown in Figure 31, we removed about 47% during step 1 and about 7% more during step 2. In total about 47% of the original turbine data was removed during steps 1 and 2.

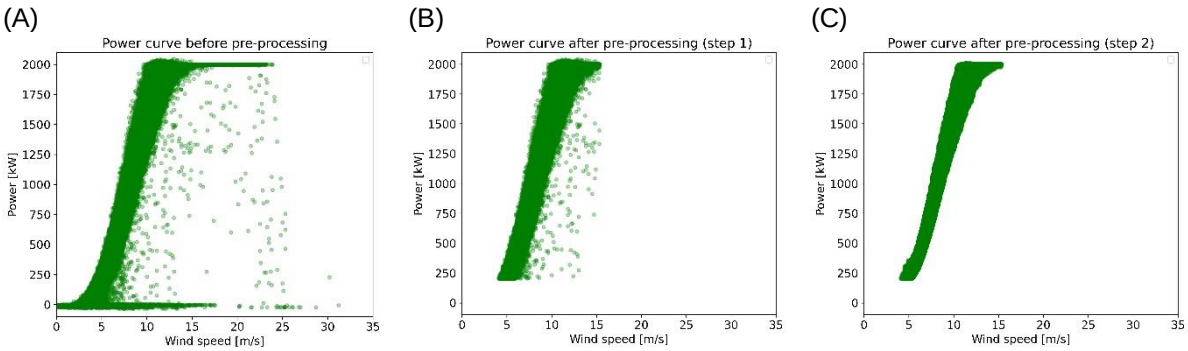


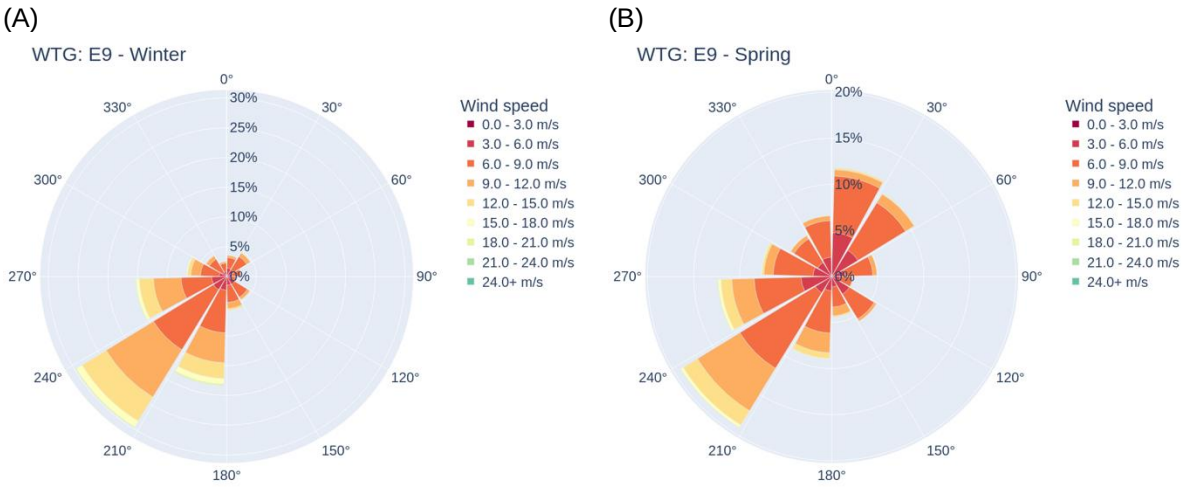
Figure 31. The impact of data pre-processing is shown for one turbine of the use case 2: (A): with outliers, (B) without outliers (step 1) and (C) without outliers (step 2).

The reason for adapting the quantile range (step 2 during pre-processing) in use case 2 is the presence of erroneous wind speed data resulting from uncorrected wind speed measurements once a turbine upgrade was installed. To address this issue, we introduced an additional data processing step in which we corrected the erroneous wind speed data using information from surrounding wind turbines. This correction was achieved using a GNN and is described in Section 3.3.4.

3.3.2 Data analysis

Wind roses

For use case 2, we once again analysed the predominant wind speed and direction by partitioning the year into its four seasons and selecting a WTG from the wind farm that we assume is less influenced by the others due to its highest elevation within the wind farm. Following this procedure, the wind predominantly comes from directions between 210° to 240°, i.e., Southwest wind directions, with maximum wind speed values up to 18 m/s (Figure 32). During winter and autumn, the wind blows predominantly from the South to the West, whereas in spring, winds also come from the Northeast, and in summer, from the North and Northwest. Wind speeds were highest in winter and lowest in summer. In spring and autumn, wind speeds are similar, reaching values between winter and summer wind speeds.



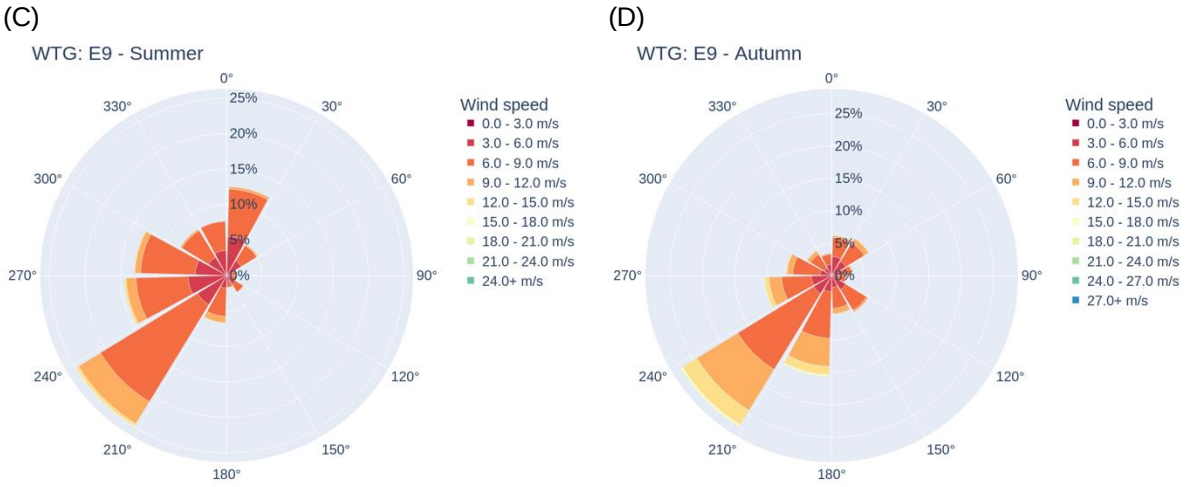


Figure 32. Seasonal wind roses derived for the highest WTG in the wind farm (WTG 9 is the turbine in the third row, second from the left in Figure 33(A): (A): for winter, (B) for spring, (C) for summer and (D) for autumn.

Terrain

A map of terrain elevations for the wind farm, the digital elevation model (DEM), was again extracted from the SRTM data set in 30 m horizontal resolution[2].

Figure 33(A) shows a cut-out of the reprojected SRTM DEM with the wind farm turbines marked by crosses. The higher terrain elevations Southwest to West are likely to have an impact on the overall wind flow patterns in the wind farm given that the predominant wind direction is Southwest, and the hills are close by. Among all wind turbine locations terrain elevation differences reach up to 32 m with a standard deviation of 10 m. Terrain slope angles at wind turbine locations vary between 0 ° to 4 ° (Figure 33(B)).

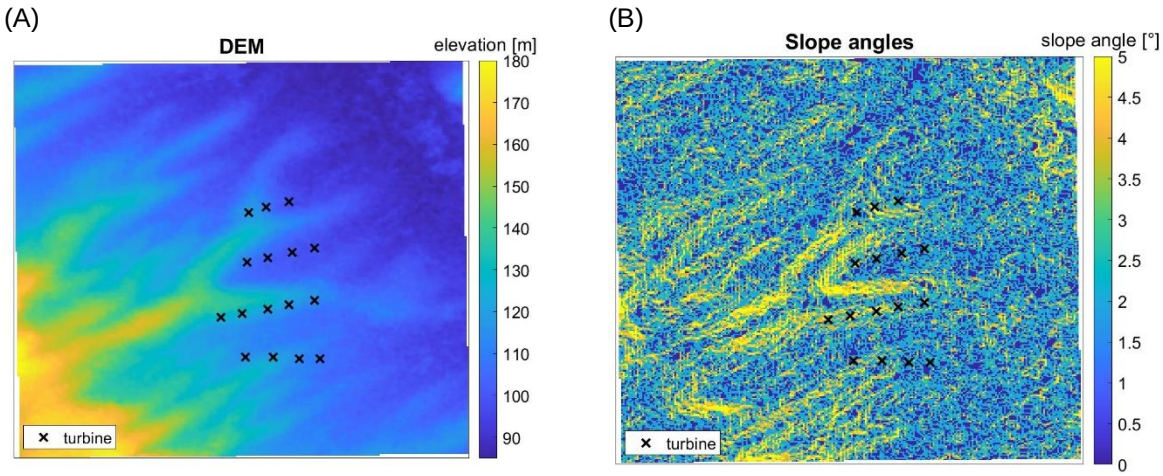


Figure 33. Wind turbines (black crosses) of the wind farm (cutout) are shown on the digital elevation model (DEM) in 30 m horizontal resolution. The DEM was extracted from the free SRTM data set.

Free stream conditions

We do not need exposure indices for use case 2 as we used it for use case 1, since for use case 2 we were not focusing on wakes but on the impact of gurney flaps and vortex generators. However, we also analysed terrain characteristics for use case 2 to verify potential topographic influences on the wind flow. In Figure 25(A), we show the spatial mean Sx exposure indices, derived as described for use case 1, but for 9 angular intervals of 40°.The most exposed turbine is indicated with a circle, potentially marking the turbine location with approximate free stream conditions. Predominant wind directions for the turbine with approximate free stream conditions according to spatial mean Sx parameters (Figure 34) were from 210° (Southwest) to 270° (West) (Figure 34(B)). Free stream wind directions are slightly more oriented toward western directions than for the highest WTG, where winds predominantly come from the southwest (Figure 32 vs. Figure 34(B)). Spatial mean Sx-value variations varied at turbine locations for the 9 angular intervals between 0.3° and 0.7° for the 16 turbines shown Figure 34(A). While overall terrain heights were much lower for use case 2 than use case 1, the similar or even larger spatial Sx variations for use case 2 indicate that absolute terrain heights are not necessarily indicators for the sheltering or exposure of wind turbines. This agrees with that the mean over all spatial mean Sx-values at the turbines of -50.13° is only slightly larger than the one for the six turbines in use case 1 of -52.1°.

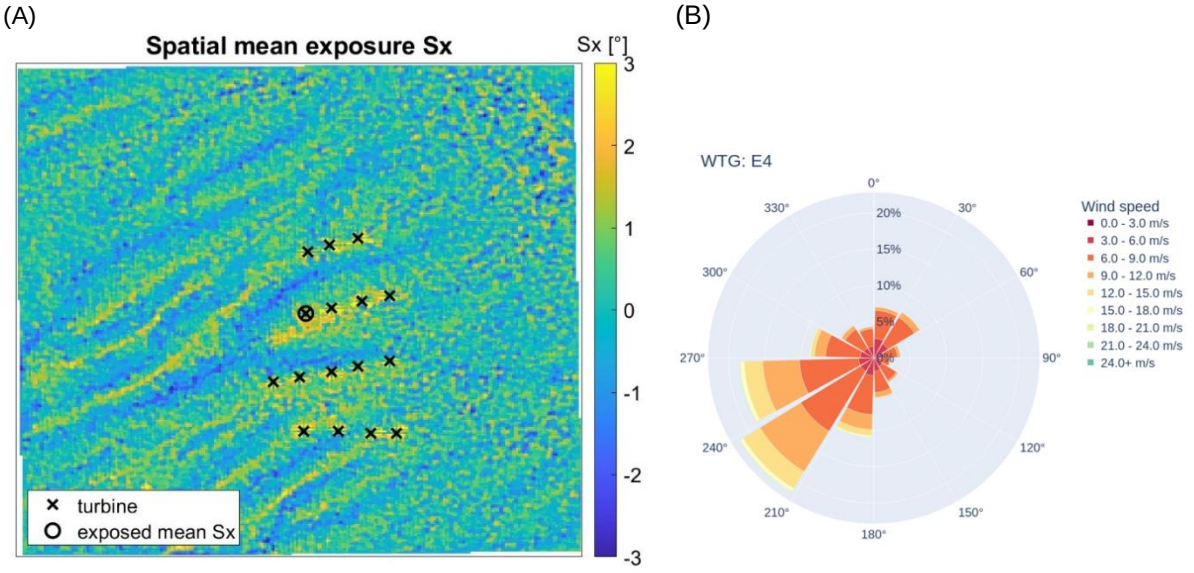


Figure 34. (A): Spatial mean exposure angle Sx for the wind farm in 30 m horizontal resolution, calculated on the SRTM-DEM (use case 2). Locations of turbines are indicated with black crosses. The circle indicates the most exposed turbine according to the spatial mean Sx exposure angles. (B) Wind rose of WTG E4 for the entire year. WTG E4 is the most exposed turbine as shown in (A).

3.3.3 Turbine upgrade quantification

Paramount for determining an accurate value for a turbine performance increase due to upgrades is an accurate turbine performance assessment. The performance is assessed before and after the upgrade and then compared. This can be done, for example, by covariate matching [6], [55], where the main goal is to find long enough periods before and after the upgrade where the wind conditions are very similar and the difference in power is compared.

Another approach is the analysis of power curves based on the standard IEC method of binning [56]. However, a major drawback is that it falls short in terms of power predictions [54], as further

environmental conditions cannot be taken into account due to the nature of the IEC model. It has also been shown in the literature that additional environmental conditions such as turbulence intensity, density, wind direction and shear factor have a noticeable effect on the power prediction accuracy [51], [54], [57].

Hence, the power curve is commonly modelled through a data-driven approach, where a model for a long period before the upgrade, to capture the yearly, monthly, and daily seasonality effects is trained. One can then use this power curve model to predict the power output for a period after the upgrade and compare it to the actual power output. The difference is then attributed to the upgrade effect. This method has been used with success in the literature [6], [8], [12] and was therefore used for this work as well.

The first step in this approach is the definition of three time periods, see Figure 35. A period for training the power curve model, "train", that spans a time range long enough to capture the seasonality of wind conditions within a year. Simultaneously, it is crucial to ensure this period is free from any issues related to turbine degradation. After that period, spanning a time range before (pre) and after (post) the turbine upgrade are defined. Lee et al. [12] chose a month for both periods, Astolfi et al. [8] chose six months.

Figure 35. The three periods for the turbine quantification process. Data from the train period is used to build a power curve model. Power outputs are then predicted for a period before (pre) and after (post) the turbine upgrade.

A power curve model is then trained on the "train" dataset and tested on the "pre" and "post" datasets by calculating the error residuals based on equation:

$$y(\boldsymbol{x}) - \hat{y}(\boldsymbol{x})$$

where $y(\boldsymbol{x})$ and $\hat{y}(\boldsymbol{x})$ are the measured and predicted power for a point $\boldsymbol{x}$**.** This will result in a residual distribution for the pre and post datasets. The main idea is that, if the turbine upgraded resulted in a change in performance, the predicted power values and the resulting error residuals for the post period are statistically different to the ones obtained for the pre period. To check for the statistical difference and its significance, Lee et al. [12] and Astolfi et al. [8] calculated the *t*-statistic and its *p*-value. Within this work we chose to use the permutation test to determine the statistic and its *p*-value.

After successfully determining that the turbine upgrade had a significant effect on the measured data, the performance difference needs to be quantified. This can be done by summing the differences between the measured, $y(\boldsymbol{x})$, and predicted, $\hat{y}(\boldsymbol{x})$, power values is calculated by:

$$\mathrm{DIFF}(x) = \frac{\sum_{x \in D_{test}} \left(\mathrm{y}(x) - \hat{\mathrm{y}}(x)\right)}{\sum_{x \in D_{test}} \mathrm{y}(x)} \cdot 100\%$$

for both the pre and the post upgrade periods [6], [8], [12]. After that the difference between these two differences gives the value for the actual performance upgrade, defined by:

$$\mathrm{DIFF} = \mathrm{DIFF}_{post}(\boldsymbol{x}) - \mathrm{DIFF}_{pre}(\boldsymbol{x})$$

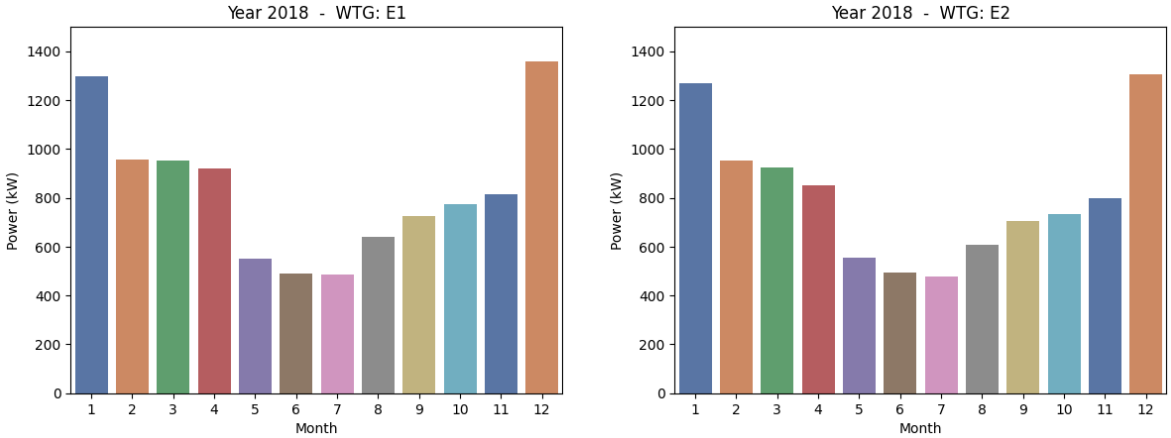
However, one important problem remains. As mentioned above, the turbine power is susceptible to various environmental conditions. It is not possible to measure and capture all of them for use in a power curve model. Hence, some environmental conditions, not captured by the power curve model, for the "pre" and "post" periods might differ too much and can cause, at least partially, a difference in power outputs for the two periods. For this reason the use of a control turbine is advised [6], [8], [12], where no upgrades were installed and do the exact same analysis as describe above. This will yield in a power difference, $\mathrm{DIFF}_{control}$, between the pre and post upgrade periods. This difference is then explained by the existence of difference in environmental conditions that where not captured by the power curve model. The final quantification of the turbine performance due to upgrades is then calculated by taking this difference into account as well:

$$DIFF_{upgrade} = \mathrm{DIFF} - \mathrm{DIFF}_{control}$$

An important point to note, however, is that the ground truth is not knowable when only using measurement data. Training and comparing a list of different models to get an overall idea of the effect of turbine upgrades on the performance of wind turbines is suggested [6], [8], [12].

Period splitting

As described above, the data is split into three periods for the turbine upgrade quantification process. To better understand the wind conditions and the resulting power outputs, and to choose appropriate time periods, for the "pre" and "post" datasets, the monthly power outputs and wind speeds for both WTGs E1 and E2 for the year 2018 are shown in Figure 36. For the months before August, in which the upgrades were installed for E1, lower wind speeds and much lower power outputs can be seen. After August the power outputs increase noticeably, with an extreme jump in December. The wind farm site seems especially high yielding in winter and spring, likely due to the higher wind speeds in winter (Figure 36). Based on this, the training period was chosen from April 2017 until April 2018 to capture the observed extremes within a year. For the "pre" period data from the months of May, June, July and half of August, before the upgrade, were used. For the "post" period data from half of August, after the upgrade, September, October and November were taken.



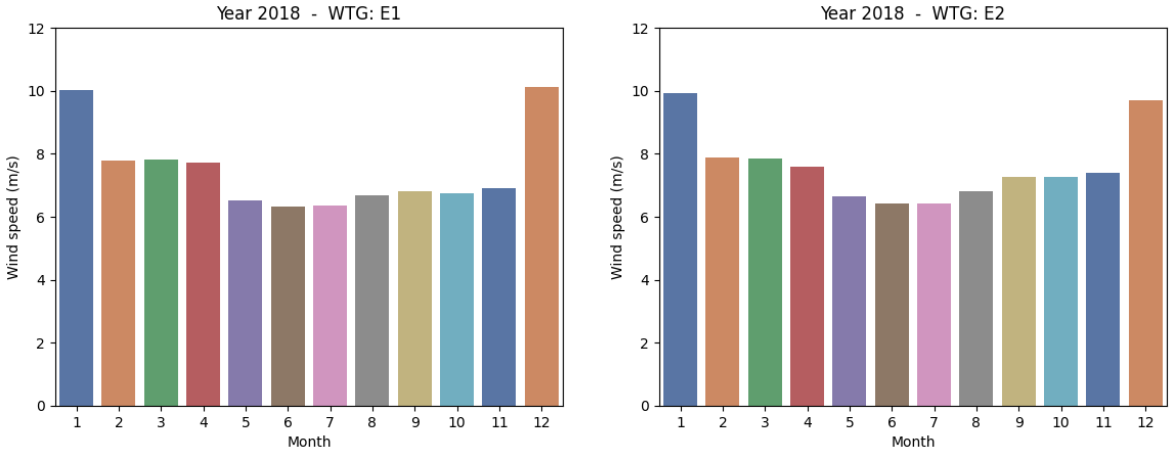


Figure 36. The monthly averaged power (top) and wind speeds (bottom) for WTGs E1 and E2 for the year 2018.

Turbine upgrades

The wind farm has a total count of 16 wind turbines, for each of which measurements from 2012 until 2022 are available. In August 2018 six turbines were retrofitted by carrying out a software upgrade – the purpose of which is not known to us – and the installation of vortex generators and gurney flaps. Four turbines only received the software upgrade, whereas the remaining six turbines were left unchanged for control. A short summary is given in Table 4.

Table 4. Overview of the installed upgrades on the WTGs. The Control group has not received any upgrade.

Control	Software Upgrade	Software + Gurney Flap + Vortex Generators
E2, E4, E7, E11, E14, E16	E3, E5, E8, E10	E1, E6, E9, E12, E13, E15

In order to see if there is a noticeable effect of turbine degradation and the performed upgrades on the turbine power, the power curves for WTGs E1 (full upgrade) and E2 (control) for each year are shown in Figure 37. As can be seen, for WTG E1 the power curves for the year from 2013 to 2015 are noticeably shifted towards the right, resulting in less power for the same wind speed. This is not observed for the control WTG E2, which shows a much narrower spread between all power curves. A marked performance jump can be seen for the years 2016, with the trend following for the years onward of 2018, where the upgrade occurred. Figure 38 shows the AEP values for each year for both turbines, calculated by multiplying the power curves from Figure 37 with the wind speed frequency distribution from the same year. Overall, WTG E1 produces on average 4% more energy compared to E2; however, the yearly trend is very similar. This contrasts with the power curve plots, which showed much larger differences, and demonstrates the importance of considering wind speed frequency distributions in performance quantification.

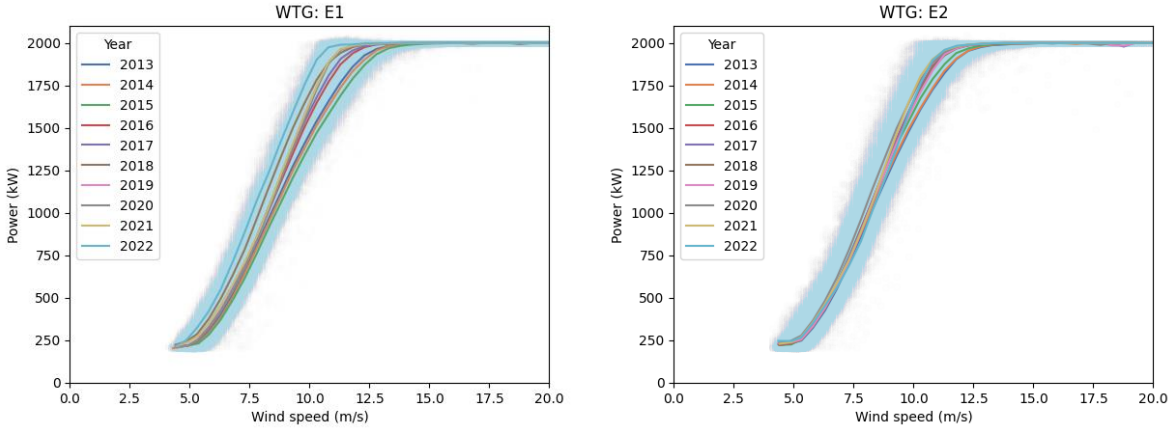


Figure 37. Power curves for WTGs E1 (full upgrade) and E2 (control) for each year.

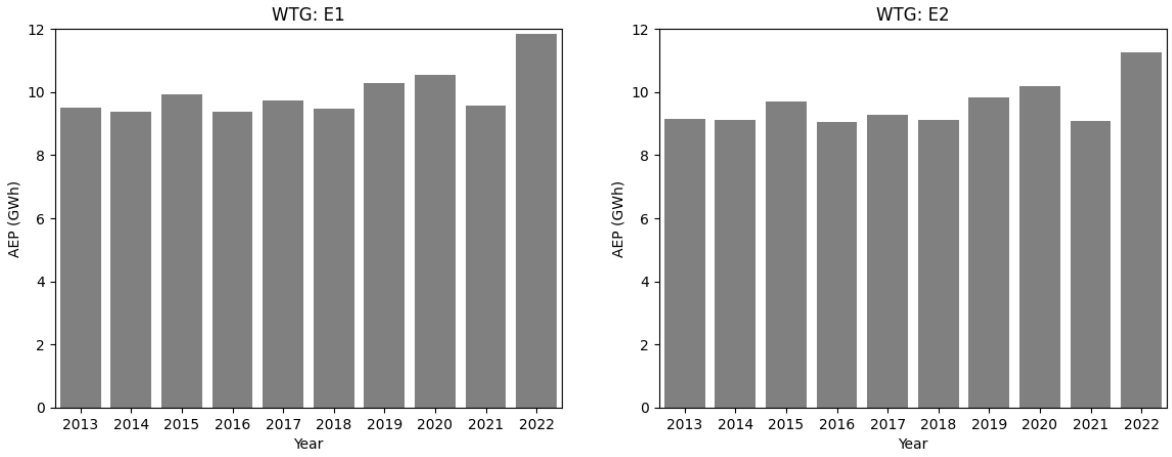


Figure 38. Annual energy production over the years for WTG E1 and E2

Another measure to quantify turbine performance is the power coefficient, C_p, defined as

$$C_p = \frac{P}{0.5\rho\pi R^2 v^3}$$

with the measured power, P, the density, ρ, the rotor radius, R, and the absolute wind speed, v. Figure 39 shows the power coefficient versus the wind speed for two fully upgraded turbines, E1 and E9, as well as two control turbines, E2 and E8, for the years from 2016 to 2019. No change is observed for the two control turbines. On the contrary, marked difference in the case of the upgraded turbines can be seen. A large difference for WTG E1 occurs in 2018 as well as for WTG E9 in 2019, and to some extend in 2018 as well. Vortex generators and Gurney flap upgrades change the geometry of the rotor blades and hence alter the aerodynamic properties of the turbine. This may then affect how the turbine interacts with the incoming wind and a recalibration of the anemometers might be necessary to ensure accurate readings. One reason for the observed behaviour might therefore be a missing sensor recalibration. The same was observed by Astolfi et al. [8], with the conclusion that no anemometer recalibration was performed after the installation of the turbine upgrades, rendering the measured wind speeds from this point onwards unreliable.

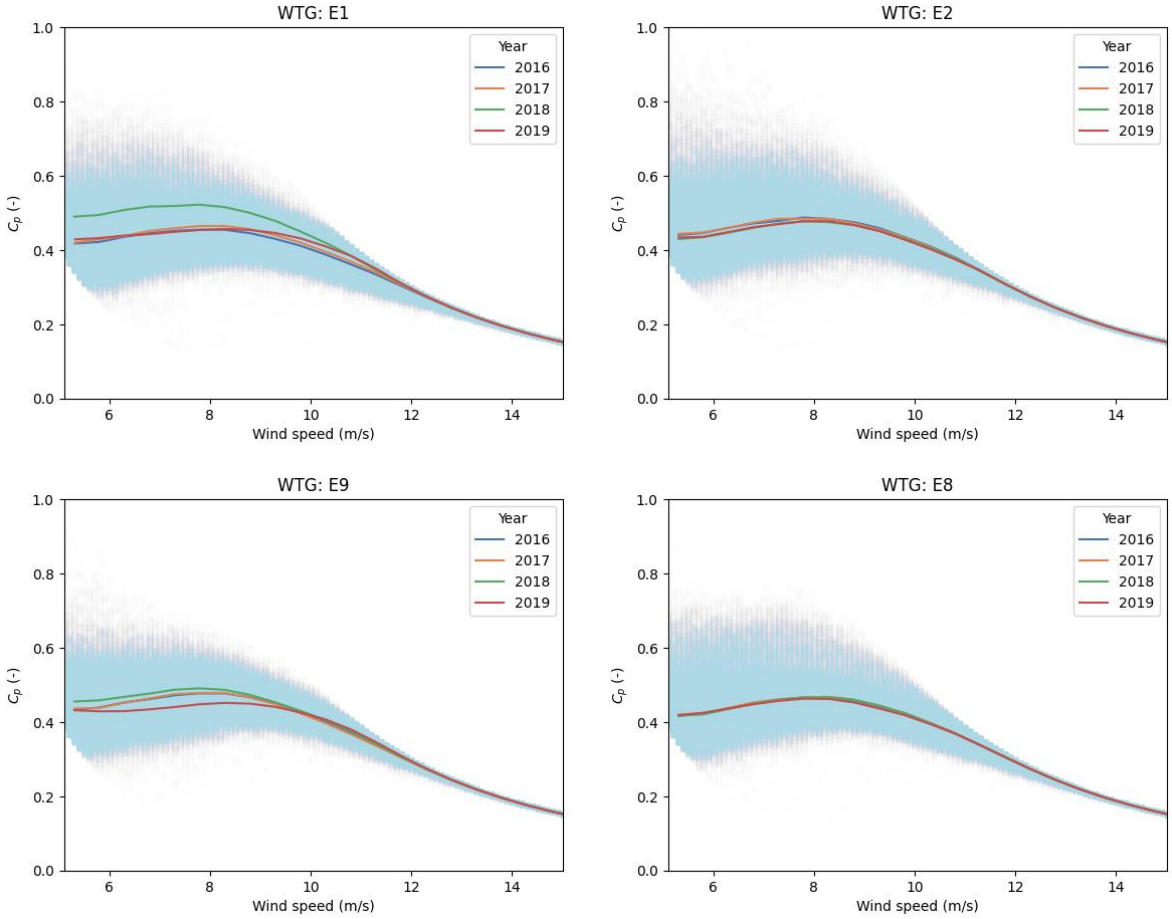


Figure 39. Power coefficients versus wind speeds for two fully upgraded turbines, E1 and E9 (left), as well as two control turbines, E2 and E8 (right), for the years from 2016 to 2019.

To overcome this issue, Astolfi et al. [8] built a neural network model, predicting the power output for the upgraded turbines based on power measurements on neighbouring control turbines. The upgrade quantification was then carried out based on these corrected power output values. This process is also known under the name data imputation. Assuming, based on the power coefficient plots above, that the anemometers were not properly recalibrated, we imputed the wind speeds as well. In the following section, we will present our data imputation approach using GNNs.

3.3.4 Data imputation with GNNs

The unreliable wind speed measurements for the turbines where the upgrades are installed heavily affect the data in the “post” period for the upgrade quantification. For this reason, we decided to impute the data for these periods. Data imputation originally comes from statistics and deals with the replacement of missing or faulty data with replacement values. In the context of machine learning, this task can be formulated as a prediction problem. Conventional machine learning methods such as decision trees, k-nearest neighbours and support vector machines as well as deep learning methods such as recurrent neural networks, generative models and autoencoders have been used for data imputation [58]. Very recently, GNNs were also used for data imputation, stressing the importance of taking global and local relationships in a dataset into account [58], [59]. In the following we briefly describe our approach for data imputation with the GNN model introduced for use case 1.

Based on the problem setting of imputing faulty wind speeds, the targets for the GNN are given already. Instead of the turbine power, we want to predict the wind speeds and wind directions. A natural choice is to predict the u and v component of the wind speed vector. The edge features are the same as for use case 1, where the x and y components of the relative distance between neighbouring turbines were used. Here, no edges were eliminated, and a fully connected graph was used in order to capture the global and local relationships in the data [58]. The feature vectors for each node were chosen to also be the u and v component of the wind speed vectors as well as the turbine coordinates. However, in this case adjustments were made for the upgraded turbines. In the training dataset, where the wind speeds are assumed reliable for all turbines, the wind speeds for the to be upgraded turbines were fixed to zero and the targets to the measured u and v components. The goal for the GNN was then to learn the relationships between these u and v components with the information of the neighbouring WTGs.

For this dataset another round of hyperparameter tuning through Bayesian optimisation was performed. The length of the training period ranged from April 2017 up to August 2018. A turbine cluster of seven turbines was used to test and demonstrate the GNN for data imputation. Turbines E1 to E7 were transformed into a PyTorch Geometric dataset. Turbines E1 and E6 are fully upgraded turbines, whereas the rest are used as control turbines. The dataset was then further split into a training (80%) and validation set (20%) with a number of graphs of around 155'000 and 38'000, respectively.

Figure 40 shows the measured versus the predicted wind speed for the validation dataset for control WTG E2 and upgraded WTG E1. The prediction of wind speed for the control WTG works very well, having a correlation coefficient of 0.998. The wind speed prediction for WTG E1 show a larger spread but still have a very high correlation coefficient of 0.966.

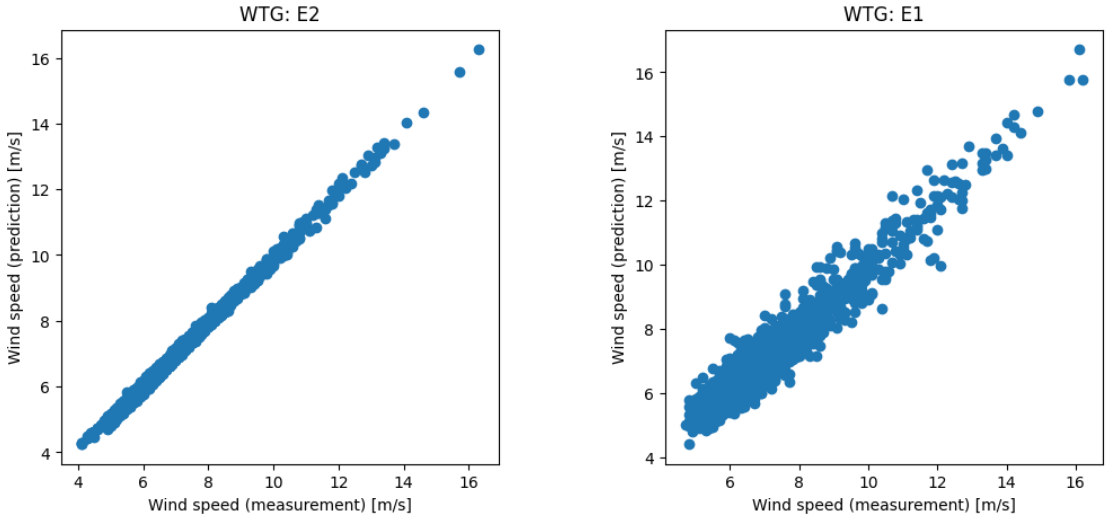


Figure 40. The measured versus the predicted wind speed for the validation dataset for control WTG E2 (left) and upgraded WTG E1 (right)

The fully connected graph with attention coefficients as well as the attention coefficient matrix for a given wind direction and wind speed are shown in Figure 41. In this case, the wind is coming from the south-west, which is also the main wind direction. The attention coefficients, when looking at the x-axis of the attention coefficient matrix, show how much importance is put on the information of the neighbouring turbines. As can be seen, WTG E1 puts large emphasis on control turbine E2 and WTG E6 on control

turbine E7. An additional advantage of the GNN results is that they also allow for choosing a sensitive control turbine for the turbine upgrade quantification process.

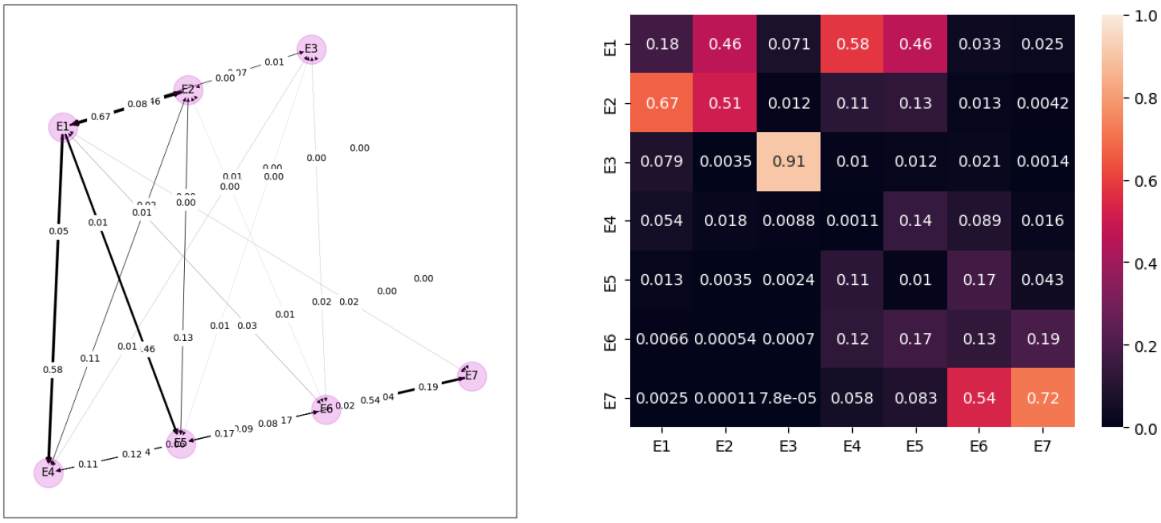


Figure 41. The fully connected graph with attention coefficients (left) and the corresponding attention coefficient matrix (right) for a given wind direction and wind speed.

Finally, the wind speeds of the upgraded turbines in the "post" upgrade dataset, ranging from August to November 2018, were replaced by the GNN predictions. To evaluate how well the imputation process worked, the C_p values versus wind speeds for WTG E1 are plotted in Figure 42, for both the "pre" and the "post" datasets without data imputation (left) and with data imputation (right). Without data imputation the C_p curve even exceeds the theoretical maximum according to Betz, further showing the unreliability of the measured wind speeds after the upgrades were installed. On the contrary, the power coefficient curve with the imputed wind speeds seems more reasonable and approaches the based on the "pre" dataset. In the following section the influence on the turbine upgrade quantification results for the cases with and without imputed wind speeds are examined.

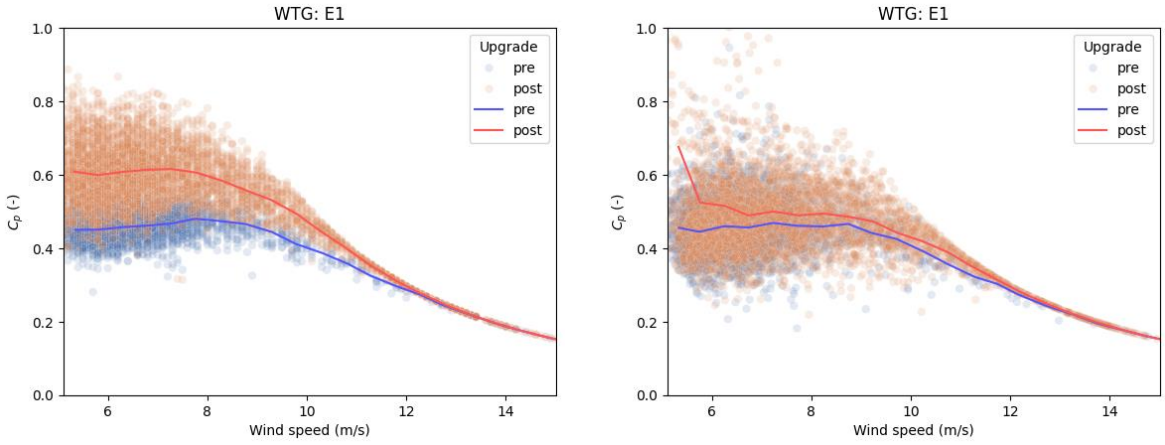


Figure 42. Power coefficient versus wind speeds for WTG E1 for the "pre" and the "post" datasets without data imputation (left) and with data imputation (right).

3.3.5 Turbine upgrade quantification

In this last section the turbine upgrade quantification results are presented for turbine E1. Based on the proximity to E1 and the attention coefficients of the GNN model, turbine E2 was used as control. To briefly recap, the following period split was done for the quantification:

- Train period: From April 2017 to April 2018
- Pre period: From May 2018 to the first half auf August 2018, just before the upgrade
- Post period: From the second half of August, just after the upgrade, to November 2018

This was done for the cases with and without wind speed imputation to see the impact of unreliable measurements. In total two different power curve models were trained:

1. For E1 on the training data with around 110'000 data points
2. For E2 on the training data with around 110'000 data points

The XGBoost library was used to train boosted trees, as these have shown to perform well for power curve modelling [54].

Table 5 shows the mean squared error (MSE) for both models on the train, test, pre and post datasets. As can be seen, the MSE for all sets are similar, showing that the models are able to generalise reasonably well. The only large deviation can be seen for the post dataset for turbine E1. This shows that the model is not able to accurately predict the power output of E1 for this period, indicating that the turbine upgrade altered the turbine behaviour significantly. For the post dataset with the GNN corrected wind speeds, the error slightly reduces, showing that part of this error was only due to the unreliable wind speed measurements and not the effect of the upgrade itself. For the control WTG E2 no decrease in prediction accuracy is observed. Keep in mind that the error scores for the train, test and pre datasets is not supposed to change for E1 between the uncorrected and the corrected case, as only the wind speeds in the post dataset were imputed.

Table 5. Mean squared error for both models on the train, test, pre and post datasets. The MSE for the post dataset with the GNN corrected wind speeds (right column).

	Uncorrected wind speed				GNN corrected wind speed
	Train	*Test*	*Pre*	*Post*	*Post*
E1	37.2	37.4	33.4	**126.9**	**117.0**
E2	34.3	35.1	37.4	34.9	34.9

We now move on to check whether the observed large differences in mean squared error for E1 are statistically significant. For this the residuals for the pre and post datasets were calculated, and a permutation test was performed to derive the statistical value and the p-value. The results for the two turbines for the uncorrected and corrected wind speeds are depicted in Table 6. Commonly, a p-value of less than 0.05 is used to determine whether an observed difference between distributions is statistically significant or not. As can be seen, for both cases the large difference in mean squared error are significant, which further confirms an effect of the turbine upgrade. For the control turbine E2 a statistically significant difference was observed as well, however, small. As mentioned above, this might be due to some underlying differences in the environmental conditions that were not captured in the data itself.

Table 6. The statistical value and the p-value for the two turbines for the uncorrected and corrected wind speed cases.

	Uncorrected wind speed		GNN corrected wind speed	
	Statistic	*p-value*	*Statistic*	*p-value*
E1	-93.5	0.0002	-79.6	0.0002
E2	2.6	0.0002	2.6	0.0002

Having established the fact that the turbine upgrades had a significant effect on the power output of turbine E1, we now need to quantify the strength of the effect. The results are shown in Table 7. A difference, DIFF, of around 16.9% between the pre and post power outputs for WTG E1 were observed, meaning that the turbine produces 16.9% more power on average. For the control turbine an increase of 3.7% was found. This might partly be explained by looking at Figure 39. The pre dataset contains data from the months with very low power outputs and the post dataset contains data from months with noticeably higher power outputs. One way that might alleviate and shed light on this issue is to test various different pre and post periods, which should be addressed in future work.

The final upgrade quantification for turbine E1 is around 13.2%, which is very significant and uncommon, when looking at the literature. This value greatly reduces when using the dataset with the corrected wind speeds, with which a final quantification of around 3.8% was yielded. This makes sense, given that the C_p curve for E1 exceeded the theoretical maximum due to the faulty wind measurements, leading to an over-estimation of the effect of the turbine upgrade.

Table 7. The performance difference, DIFF, between the pre and the post dataset power predictions and the final upgrade quantification for the cases with and without data imputation.

	Uncorrected wind speed		GNN corrected wind speed	
	DIFF	*Quantification*	*DIFF*	*Quantification*
E1	16.9%	13.2%	7.4%	3.8%
E2	3.7%	-	3.7%	-

The approach and results presented in this chapter are not limited to small turbine clusters but can be used for the full wind farm as well. However, for the sake of brevity and simplicity, only a subset of the data and results were shown to give a reasonable amount of intuition of how and why this approach works as well as how developer might use it for their own work.

In this use case we looked at a large wind farm, where turbines were retrofitted with vortex generators and Gurney flaps. Data analysis revealed that the wind speed of the upgraded turbines might be unreliable due to missing recalibration of the anemometers after the installation. We therefore developed a GNN model that was able to predict wind speed values for the upgraded turbines based on neighbouring turbines. The capability of this prediction was shown by comparing the power coefficients for a period before and after the upgrade. A clear improvement was observed. However, it needs to be further checked with the project partner whether there might be other or additional reasons for this behaviour.

After that we were able to identify a positive effect of the turbine upgrades on the turbine performances. The effect was quantified, showing an improvement of more than 13% based on the dataset with the

faulty wind speed measurements. In comparison, when using the GNN corrected wind speeds the performance improvement was around 3.8%, which was more in line with values found in the literature.

3.4 Open-source library with example (use case 3)

For use case 3 we decided to use an open-source dataset [18]. The majority of the time was spent during this project looking at use cases 1 and 2. However, as part of this work is the development and publishing of an open-source library, the open-source dataset was used to showcase the library as an example. Within the repository of the **openimpact** library, published on Github[10], an "example" folder was created, containing scripts for the data cleaning and preparation process, the creation of a PyTorch Geometric dataset, hyperparameter tuning via Bayesian optimisation as well as the GNN training process. The creation of these examples also constituted the main work of this use case.

The example folder contains a TOML[11] configuration file, see Figure 43, that is used to define the used datasets as well as for selecting the features for the GNN. In case the data contains column names with special signs or whitespaces, the names are conveniently remapped to simpler ones.

```
[csv]
encoding = "utf8"
sep = ","
header = 0

[index]
name-from-source = "# Date and time"
time-zone-from-source = "UTC"
name = "datetime"
unit = "ns"
time-zone = "UTC"

[[columns]]
name-from-source = "Wind direction (°)"
name = "wind_direction"

[[columns]]
name-from-source = "Nacelle position (°)"
name = "nacelle_direction"

[[columns]]
name-from-source = "Wind speed (m/s)"
name = "wind_speed"

[[columns]]
name-from-source = "Power (kW)"
name = "power"

[[columns]]
name-from-source = "Wind turbine ID"
name = "wt_id"

[dataset]
name = "kelmarsh_test"
data = "featured_data.csv"
static = "Kelmarsh_WT_static.csv"
```

Figure 43. TOML file for use case 3 (Kelmarsh wind farm)

To start the data processing and the creation of the PyTorch Geometric dataset, the file `pipeline_kelmarsh.py` must be executed. Here, the configuration file is set, and various pre-processing steps are executed. Finally, the class `KelmarshDataset` is instantiated, transforming the pre-processed column data into graphs. By executing `train_kelmarsh.py`, the GNN is trained. The train function creates a checkpoint folder, automatically saving the trained GNN. The training can then be picked up from the latest checkpoint at a later time or used for deployment.

[10] https://github.com/weid-ost/openimpact
[11] https://toml.io/en/

An important point to note is that the current structure described here is subject to change in the future, as the **openimpact** library is under active development.

3.5 Production environment

The focus of this project was on the development of data-driven models and data pipelines for three use cases. To be able to use these models and pipelines for a production environment, the various building blocks need to be deployed. For the deployment three possible options were determined, which are shown in Figure 44:

1. Standalone
2. Docker image
3. ENTR distribution

In the “Standalone” deployment strategy the trained data-driven models, the data ingestion scripts, the dbt environment and the data warehouse are all separately deployed and connected on a chosen infrastructure, e.g., a server. For simpler use cases, e.g., where lower amounts of data are used or no continues stream of new data is needed, the ingestion scripts as well as the dbt environment and the warehouse could be simply replaced by processing scripts that can be manually triggered.

For the “Docker image” strategy the whole data pipeline would be bundled in a Docker image. A Docker container can then be easily and readily deployed anywhere.

The last strategy allows to bundle the data pipeline and the OpenIMPACT model in an ENTR distribution, which is being developed by the ENTR Alliance and NREL. The main advantages are easy deployment anywhere as well as data standards and interfaces used in the wind energy industry. A downside might be the dependence on a third-party environment.

Lastly, users and other applications need to be able to communicate with the trained model by sending data as well as receiving feedback and predictions. A very common approach is to use a RESTful API for model deployment. REST, short for Representational State Transfer, sets specific rules for how Application Programming Interfaces (APIs) are designed and function[12]. Modern web frameworks, such as FastAPI[13], are used for building such APIs. An example for use case 3 is given in the OpenIMPACT library[5] to show how an API could like for the developed GNN models.

[12] https://aws.amazon.com/what-is/restful-api/?nc1=h_ls
[13] https://fastapi.tiangolo.com/

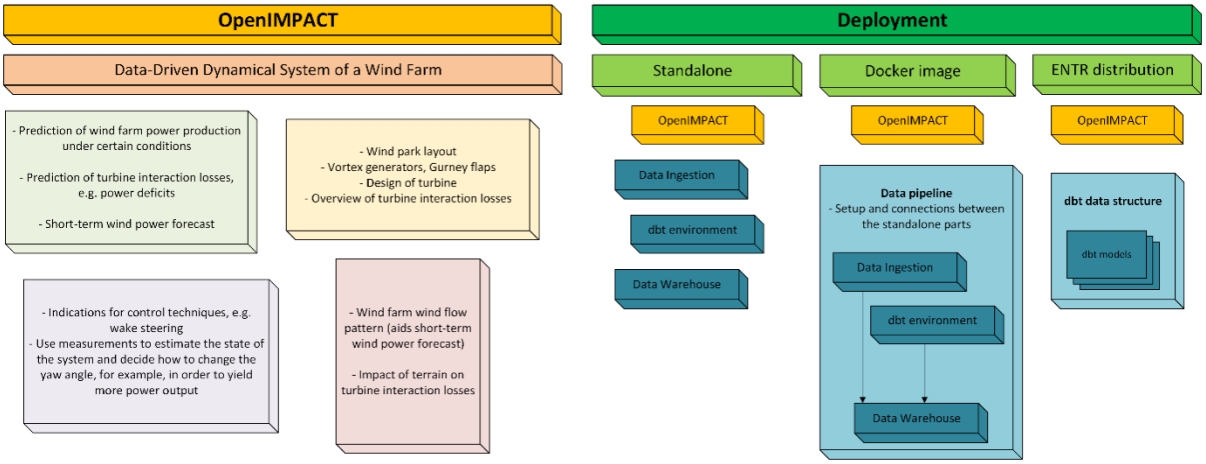


Figure 44. Deployment options for the data pipeline, model and production environment

The **openimpact** contains examples and documentation for the “Standalone” approach based on use case 3. A data processing pipeline is available, as well as scripts for hyperparameter tuning and training of a GNN. Furthermore, an API based on FastAPI is available, which contains its own simple and local server, through which we deployed the model. The API allows to send data to the model and request predictions.

As we plan to build challenges via the WeDoWind Framework, see Section 7, in which the **openimpact** will be used, we hope to be able to cooperate with further partners and continue the development of the “Docker image” and “ENTR distribution” deployment strategies.

4 Conclusions

The goal of this work was to develop of an open-source library for applying novel machine learning solutions for optimising wind farm performance in complex terrain based on SCADA data. In order to achieve this, three different use cases were investigated.

Two use cases provided by the industrial partners WinJi AG and Elektrizitätswerke des Kantons Zürich (EKZ) were concerned with wake interaction losses and turbine upgrade efficacy in wind farms, respectively. The third use case was based on an open-source dataset [18]. The main purpose of this use case was to showcase the developed **openimpact** library.

Firstly, a concept for a data pipeline that ingests data into a data warehouse and transforms it into a standardised data structure with standardised naming conventions based on the norm IEC 61400-25 was developed. Documentation was published in the **openimpact** library that helps guide users to develop and deploy this data pipeline.

For use case 1, we developed a simple graph model and a Graph Neural Network (GNN) for predicting wake-interaction effects. The simple graph model was able to capture the effect of wakes in the given dataset by relating the turbine power outputs by the power outputs of its connected neighbour turbines. However, the approach was limited in that only one feature – here the power output – could be incorporated to find those relationships. To leverage important features such as wind speeds, yaw errors and spatial values, such as coordinates, relative distances, and angles, a GNN was developed. We demonstrated how to search for GNN designs and how to optimise GNN hyperparameters through Bayesian optimisation. The GNN model was then able to learn attention coefficients that represent the amount of importance one turbine puts on a neighbouring turbine, while aggregating spatial information (turbine coordinates, relative distances, and angles) and features (wind speeds and yaw errors). We showed that the GNN model was able to capture wake interaction effects present in the dataset and predict power curves that show waked and non-waked behaviour. However, the model underestimates the total energy production by about 4% compared to the measurements. Furthermore, the provided dataset is limited to two years and is lacking important features such as turbulence intensity, which was shown to have a considerable effect on power predictions [54]. Therefore, in future work the development of a GNN could be accompanied by datasets over longer time periods as well as by datasets with a wider range of atmospheric conditions.

For use case 2, we looked at a large wind farm in which some of the turbines were retrofitted with vortex generators and Gurney flaps. Data analysis revealed that the wind speed measurement of the upgraded turbines might be unreliable due to missing recalibration of the anemometers after their installation. We therefore developed a GNN model that was able to predict wind speed values for the upgraded turbines based on neighbouring turbines. The capability of this prediction was shown by comparing the power coefficients for a period before and after the upgrade. A clear improvement was observed. However, further checks together with the project partner need to be undertaken to confirm this assumption and if there might be other or additional reasons for this behaviour. We were then able to identify a positive effect of the turbine upgrades on the turbine performances. The effect was quantified, showing an improvement of more than 13% based on the dataset with the supposedly faulty wind speed measurements. In comparison, when using the GNN corrected wind speeds the performance improvement due to the upgrades was about 4%, which is in line with values found in literature.

Lastly, the **openimpact** library was developed and published on Github. The library contains examples and documentation for a “Standalone” deployment approach based on the third use case. A data processing pipeline was made available, as well as scripts for hyperparameter tuning and training of a GNN. Furthermore, an API based on FastAPI was developed, which contains its own simple and local server, through which the model can be deployed. The API allows to send data to the model and request predictions.

5 Outlook and next steps

A further goal with the **openimpact** library is to use it for challenges via the WeDoWind Framework. We are already in the process of creating a new WeDoWind space together with the OpenOA team from NREL in order to encourage the use of the library. The plan to launch this new space with a public webinar at the start of the year 2024 is already in place.

This will allow us to further develop the library and tackle some of the problems encountered during the development of the GNN models:

- The complexity of the GNN requires high quality data for long periods of time, which also include essential features such as turbulence intensity. Through further challenges, more and more datasets can be used and tested. This will also help to test and improve the generalisability of the models, i.e., how well they work on unseen data from various wind farms.
- How to better deal with noisy data, i.e., field measurements? Most GNNs developed for wind energy applications were based on clean simulation data. Developing models based on both measurement and simulation data could potentially help to find GNN designs that can better deal with measurement data.
- More methods are currently being developed that deal with the problem of how to best incorporate spatial geometries of the underlying graph in GNNs [41]. Currently, coordinates and distances are used within the feature vectors of nodes and as edge attributes. Using geometric graphs instead, nodes have, in addition to a feature vector, a coordinate vector. This could greatly improve the accuracy of GNN wind farm models.
- The developed approach within this project also sets a strong constraint on how the various graphs are built based on the available data and then stored in a dataset. Currently, only points in time are chosen for which data for all turbines is available. The problem with this is as more and more turbines are modelled. If a data point is missing for one out of 50 turbines, the data points for all turbines corresponding to the same time are discarded. Here, a GNN could be leveraged in a pre-processing step, imputing missing data points, similar to what was already shown for use case 2.

Overall, GNNs are an active field of research, and we hope to have shown the viability of this approach for wind energy and the value that further research might bring. The main advantage to existing data-driven methods are the explicit incorporation of spatial information [42], [60]. In fact, many deep learning architectures with added geometric information are special cases of GNNs [40], [42].

6 National and international cooperation

For the data pipeline we worked together with the ENTR Alliance[7], NREL[8] as well as Apex Clean Energy[9] from the US We received input and guidance in terms of data standardisation as well as naming conventions, which are currently being developed for use by the broader wind energy community.

Furthermore, performance and scalability analysis for data ingestion together with Apex Clean Energy is planned.

With regards to power curve modelling we are already cooperating with Professor Yu Ding from the Georgia Tech (USA). A new challenge on the WeDoWind ecosystem concerning power curve model comparisons was developed, which we will participate in as part of this project[10].

7 Communication

The **openimpact** library was made available as a Python package on Github[14]. We will ensure national and international communication of the results by connecting it to WeDoWind, as described in Chapter 6, as well as submitted the newest results to an open-source journal such as the Wind Energy Science journal.

8 Publications

Part of the graph model, presented in Section 3.2.4, was developed during the Open Energy Data Hackdays 2022 in the Hightech Zentrum Aargau. Currently, a report for the SFOE about the learnings and experiences of the Hackdays is being written, in which the OpenIMPACT project will be mentioned.

The graph model and its results were presented at the Wake Conference 2023 in Uppsala, Sweden, as well as published in the Journal of Physics: Conference Series, with the title “Graph machine learning for predicting wake interaction losses based on SCADA data“ [61].

[14] https://github.com/weid-ost/openimpact

9 References

[1] J. Barcons, M. Avila, and A. Folch, 'Diurnal cycle RANS simulations applied to wind resource assessment', *Wind Energy*, vol. 22, no. 2, Art. no. 2, Feb. 2019, doi: 10.1002/we.2283.

[2] F. Schmid, J. Schmidli, M. Hervo, and A. Haefele, 'Diurnal Valley Winds in a Deep Alpine Valley: Observations', *Atmosphere*, vol. 11, no. 1, Art. no. 1, Jan. 2020, doi: 10.3390/atmos11010054.

[3] Internationale Elektrotechnische Kommission, Ed., *Power performance measurements of electricity producing wind turbines*, 1. ed., 2005–12. in Wind turbines, no. 12,1. Geneva: IEC, 2005.

[4] R. J. Barthelmie *et al.*, 'Quantifying the Impact of Wind Turbine Wakes on Power Output at Offshore Wind Farms', *Journal of Atmospheric and Oceanic Technology*, vol. 27, no. 8, pp. 1302–1317, Aug. 2010, doi: 10.1175/2010JTECHA1398.1.

[5] D. R. Houck, 'Review of wake management techniques for wind turbines', *Wind Energy*, vol. 25, no. 2, Art. no. 2, Feb. 2022, doi: 10.1002/we.2668.

[6] Y. Ding, *Data Science for Wind Energy*. CRC Press, 2019. [Online]. Available: https://books.google.ch/books?id=0qWbDwAAQBAJ

[7] H. Im, S. Kim, and B. Kim, 'Numerical analysis of the effect of vortex generator on inboard region of wind turbine blade', *Journal of Renewable and Sustainable Energy*, vol. 13, no. 6, p. 063306, Nov. 2021, doi: 10.1063/5.0065108.

[8] D. Astolfi, F. Castellani, and L. Terzi, 'Wind Turbine Power Curve Upgrades', *Energies*, vol. 11, no. 5, p. 1300, May 2018, doi: 10.3390/en11051300.

[9] M. Schlechtingen, 'Wind turbine condition monitoring based on SCADA data using normal behavior models. Part 1: System description', *Applied Soft Computing*, p. 12, 2013.

[10] E. Gonzalez, 'Using high-frequency SCADA data for wind turbine performance monitoring: A sensitivity study', *Renewable Energy*, p. 13, 2019.

[11] K. Kim, G. Parthasarathy, O. Uluyol, W. Foslien, S. Sheng, and P. Fleming, 'Use of SCADA Data for Failure Detection in Wind Turbines', in *ASME 2011 5th International Conference on Energy Sustainability, Parts A, B, and C*, Washington, DC, USA: ASMEDC, Jan. 2011, pp. 2071–2079. doi: 10.1115/ES2011-54243.

[12] G. Lee, Y. Ding, M. G. Genton, and L. Xie, 'Power Curve Estimation With Multivariate Environmental Factors for Inland and Offshore Wind Farms', *Journal of the American Statistical Association*, vol. 110, no. 509, Art. no. 509, Jan. 2015, doi: 10.1080/01621459.2014.977385.

[13] J. Maldonado-Correa and S. Mart, 'Using SCADA Data for Wind Turbine Condition Monitoring: A Systematic Literature Review', p. 20, 2020.

[14] Y. Pang, 'Spatio-temporal fusion neural network for multi-class fault diagnosis of wind turbines based on SCADA data', *Renewable Energy*, p. 15, 2020.

[15] M. Ulmer, E. Jarlskog, G. Pizza, and L. G. Huber, 'Cross-Turbine Training of Convolutional Neural Networks for SCADA-Based Fault Detection in Wind Turbines', p. 10, 2020.

[16] M. Ulmer, E. Jarlskog, G. Pizza, J. Manninen, and L. G. Huber, 'Early Fault Detection Based on Wind Turbine SCADA Data Using Convolutional Neural Networks', p. 9, 2020.

[17] H. Hwangbo, A. L. Johnson, and Y. Ding, 'Spline model for wake effect analysis: Characteristics of a single wake and its impacts on wind turbine power generation', *IISE Transactions*, vol. 50, no. 2, Art. no. 2, Feb. 2018, doi: 10.1080/24725854.2017.1370176.

[18] C. Plumley, 'Kelmarsh wind farm data'. Zenodo, Aug. 2023. doi: 10.5281/zenodo.8252025.

[19] F. Porté-Agel, M. Bastankhah, and S. Shamsoddin, 'Wind-Turbine and Wind-Farm Flows: A Review', *Boundary-Layer Meteorol*, vol. 174, no. 1, Art. no. 1, Jan. 2020, doi: 10.1007/s10546-019-00473-0.

[20] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, 'On Dynamic Mode Decomposition: Theory and Applications', 2013, doi: 10.48550/ARXIV.1312.0041.

[21] C. Yan, Y. Pan, and C. L. Archer, 'A general method to estimate wind farm power using artificial neural networks', *Wind Energy*, vol. 22, no. 11, pp. 1421–1432, Nov. 2019, doi: 10.1002/we.2379.

[22] A. Ghaderi, B. M. Sanandaji, and F. Ghaderi, 'Deep Forecast: Deep Learning-based Spatio-Temporal Forecasting', 2017, doi: 10.48550/ARXIV.1707.08110.

[23] L. Ø. Bentsen, N. Dilp Warakagoda, R. Stenbro, and P. Engelstad, 'Wind Park Power Prediction: Attention-Based Graph Networks and Deep Learning to Capture Wake Losses', *J. Phys.: Conf. Ser.*, vol. 2265, no. 2, p. 022035, May 2022, doi: 10.1088/1742-6596/2265/2/022035.
[24] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, 'Geometric Deep Learning: Going beyond Euclidean data', *IEEE Signal Process. Mag.*, vol. 34, no. 4, Art. no. 4, Jul. 2017, doi: 10.1109/MSP.2017.2693418.
[25] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, 'Koopman Invariant Subspaces and Finite Linear Representations of Nonlinear Dynamical Systems for Control', *PLoS ONE*, vol. 11, no. 2, p. e0150171, Feb. 2016, doi: 10.1371/journal.pone.0150171.
[26] N. Cassamo and J.-W. van Wingerden, 'On the Potential of Reduced Order Models for Wind Farm Control: A Koopman Dynamic Mode Decomposition Approach', *Energies*, vol. 13, no. 24, Art. no. 24, Dec. 2020, doi: 10.3390/en13246513.
[27] W. L. Hamilton, 'Graph Representation Learning', p. 141.
[28] F. Xia *et al.*, 'Graph Learning: A Survey', 2021, doi: 10.48550/ARXIV.2105.00696.
[29] L. Qiao, L. Zhang, S. Chen, and D. Shen, 'Data-driven graph construction and graph learning: A review', *Neurocomputing*, vol. 312, pp. 336–351, Oct. 2018, doi: 10.1016/j.neucom.2018.05.084.
[30] W. L. Hamilton, R. Ying, and J. Leskovec, 'Representation Learning on Graphs: Methods and Applications', p. 23.
[31] Z. Zhang, P. Cui, and W. Zhu, 'Deep Learning on Graphs: A Survey'. arXiv, Mar. 13, 2020. Accessed: Nov. 23, 2022. [Online]. Available: http://arxiv.org/abs/1812.04202
[32] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini, 'The Graph Neural Network Model', *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009, doi: 10.1109/TNN.2008.2005605.
[33] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, 'A Comprehensive Survey on Graph Neural Networks', 2019, doi: 10.48550/ARXIV.1901.00596.
[34] M. Khodayar and J. Wang, 'Spatio-Temporal Graph Deep Neural Network for Short-Term Wind Speed Forecasting', *IEEE Trans. Sustain. Energy*, vol. 10, no. 2, pp. 670–681, Apr. 2019, doi: 10.1109/TSTE.2018.2844102.
[35] T. Stańczyk and S. Mehrkanoon, 'Deep Graph Convolutional Networks for Wind Speed Prediction', 2021, doi: 10.48550/ARXIV.2101.10041.
[36] M. Yu *et al.*, 'Superposition Graph Neural Network for offshore wind power prediction', *Future Generation Computer Systems*, vol. 113, pp. 145–157, Dec. 2020, doi: 10.1016/j.future.2020.06.024.
[37] J. Park and J. Park, 'Physics-induced graph neural network: An application to wind-farm power estimation', *Energy*, vol. 187, p. 115883, Nov. 2019, doi: 10.1016/j.energy.2019.115883.
[38] J. Bleeg, 'A Graph Neural Network Surrogate Model for the Prediction of Turbine Interaction Loss', *J. Phys.: Conf. Ser.*, vol. 1618, no. 6, Art. no. 6, Sep. 2020, doi: 10.1088/1742-6596/1618/6/062054.
[39] J. You, R. Ying, and J. Leskovec, 'Design Space for Graph Neural Networks', 2020, doi: 10.48550/ARXIV.2011.08843.
[40] P. W. Battaglia *et al.*, 'Relational inductive biases, deep learning, and graph networks', 2018, doi: 10.48550/ARXIV.1806.01261.
[41] P. Veličković, 'Everything is Connected: Graph Neural Networks', *Current Opinion in Structural Biology*, vol. 79, p. 102538, Apr. 2023, doi: 10.1016/j.sbi.2023.102538.
[42] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, 'Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges', 2021, doi: 10.48550/ARXIV.2104.13478.
[43] T. N. Kipf and M. Welling, 'Semi-Supervised Classification with Graph Convolutional Networks'. arXiv, Feb. 22, 2017. Accessed: Nov. 30, 2023. [Online]. Available: http://arxiv.org/abs/1609.02907
[44] F. Wu, T. Zhang, A. H. de Souza Jr., C. Fifty, T. Yu, and K. Q. Weinberger, 'Simplifying Graph Convolutional Networks'. arXiv, Jun. 20, 2019. Accessed: Nov. 30, 2023. [Online]. Available: http://arxiv.org/abs/1902.07153

[45] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, 'Graph Attention Networks', 2017, doi: 10.48550/ARXIV.1710.10903.
[46] S. Brody, U. Alon, and E. Yahav, 'How Attentive are Graph Attention Networks?' arXiv, Jan. 31, 2022. Accessed: Nov. 30, 2023. [Online]. Available: http://arxiv.org/abs/2105.14491
[47] P. Veličković, 'Message passing all the way up'. arXiv, Feb. 22, 2022. Accessed: Nov. 30, 2023. [Online]. Available: http://arxiv.org/abs/2202.11097
[48] A. Clifton *et al.*, 'Grand Challenges in the Digitalisation of Wind Energy', Operation, condition monitoring, and maintenance, preprint, Apr. 2022. doi: 10.5194/wes-2022-29.
[49] Barber, Sarah, Clark, Thomas, Day, Justin, and Totaro, Philip, 'The IEA Wind Task 43 Metadata Challenge: A roadmap to enable commonality in wind energy data', Apr. 2022, doi: 10.5281/ZENODO.6457038.
[50] S. Letzgus, 'Change-point detection in wind turbine SCADA data for robust condition monitoring with normal behaviour models', *Wind Energ. Sci.*, vol. 5, no. 4, Art. no. 4, Oct. 2020, doi: 10.5194/wes-5-1375-2020.
[51] A. Clifton, L. Kilcher, J. K. Lundquist, and P. Fleming, 'Using machine learning to predict wind turbine power output', *Environ. Res. Lett.*, p. 10, 2013.
[52] P. McKay, R. Carriveau, and D. S.-K. Ting, 'Wake impacts on downstream wind turbine performance and yaw alignment: Wake impacts on turbine performance and yaw alignment', *Wind Energ.*, vol. 16, no. 2, Art. no. 2, Mar. 2013, doi: 10.1002/we.544.
[53] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning from data*, vol. 4. AMLBook New York, 2012.
[54] S. Barber, F. Hammer, and A. Tica, 'Improving Site-Dependent Wind Turbine Performance Prediction Accuracy Using Machine Learning', *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg*, Jan. 2022, doi: 10.1115/1.4053513.
[55] Y. E. Shin, Y. Ding, and J. Z. Huang, 'Covariate matching methods for testing and quantifying wind turbine upgrades', *Ann. Appl. Stat.*, vol. 12, no. 2, Art. no. 2, Jun. 2018, doi: 10.1214/17-AOAS1109.
[56] IEC International Electrotechnical Commission, 'Wind energy generation systems - Part 12-1: Power performance measurements of electricity producing wind turbines', no. IEC 61400-12-1:2017. Mar. 03, 2017.
[57] R. Wagner, 'Simulation of shear and turbulence impact on wind turbine performance', p. 56, 2010.
[58] I. Spinelli, S. Scardapane, and A. Uncini, 'Missing Data Imputation with Adversarially-trained Graph Convolutional Networks', *Neural Networks*, vol. 129, pp. 249–260, Sep. 2020, doi: 10.1016/j.neunet.2020.06.005.
[59] J. You, X. Ma, D. Y. Ding, M. Kochenderfer, and J. Leskovec, 'Handling Missing Data with Graph Representation Learning'. arXiv, Oct. 30, 2020. Accessed: Nov. 30, 2023. [Online]. Available: http://arxiv.org/abs/2010.16418
[60] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu, 'Interaction Networks for Learning about Objects, Relations and Physics', 2016, doi: 10.48550/ARXIV.1612.00222.
[61] F. Hammer, N. Helbig, T. Losinger, and S. Barber, 'Graph machine learning for predicting wake interaction losses based on SCADA data', *Journal of Physics: Conference Series*, vol. 2505, no. 1, p. 012047, May 2023, doi: 10.1088/1742-6596/2505/1/012047.