Long-Distance Wind Farm Flow Modelling

European Wind Energy Master Thesis

Frederik Peder Weilmann Rasmussen

TU Delft

Long-Distance Wind Farm Flow Modelling

European Wind Energy Master Thesis

Thesis report

by

Frederik Peder Weilmann Rasmussen

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on August 21, 2024

Thesis committee:	
Chair:	Dr. Ing. Roland Schmehl (TU Delft)
Supervisors:	Prof. Dr. Simon Watson (TU Delft)
	Dr. Pierre-Elouan Réthoré (DTU)
	Lina Poulsen (Ørsted)
External examiner:	Dr. Nikolay Dimitrov
Place:	DTU Risø Campus, Denmark
Project Duration:	November, 2023 - July, 2024
Student number:	5513278

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Faculty of Aerospace Engineering · Delft University of Technology

Preface

This thesis wraps up my two years in the European Wind Energy Master (EWEM) program, where I focused on Rotor Design and Aerodynamics. I had the chance to study at both the Technical University of Denmark (DTU) and Delft University of Technology (TU Delft), working alongside some brilliant minds and tapping into the resources of these top wind energy hubs.

I'm deeply thankful to those who helped me along the way: my DTU supervisor, Pierre-Elouan Réthoré, for keeping me on track with regular guidance; my TU Delft supervisor, Simon Watson, for his valuable input despite jumping in late; and Lina Poulsen from Ørsted, whose industry insights were incredibly helpful. I also appreciate Jesper Grønnegaard Pedersen's feedback during the final stretch of the project. Lastly, a big thanks to my friends and family for their endless support, especially for listening to me talk about wind farm wakes more times than they probably wanted to.

This thesis is not just the end of my studies; it's also my first step into the exciting world of wind energy. I hope the work I've done here can help push the field forward, making wind power even more efficient and sustainable.

Frederik P. W. Rasmussen
31-07-2024

Abstract

This thesis addresses the critical issue of underestimated wake effects between neighboring wind parks by developing efficient long-distance wind farm flow models using Convolutional Neural Networks (CNNs). The study compares three wake deficit models (Jensen, Bastankhah, and TurbOPark) and four neural network architectures (Convolutional Autoencoder (CAE), U-Net, CAE/MLP, and U-Net/MLP) to improve long-distance wake predictions.

A novel method for random wind park layout generation was developed, simulating diverse scenarios up to 768 rotor diameters downstream. Each wake model dataset comprised 1000 simulations, split 80/20 for training and testing. Results demonstrate that all neural networks effectively simulate wake datasets, with U-Net and U-Net/MLP consistently outperforming CAE approaches. Mean Absolute Errors (MAE) range from 4.75×10^{-4} m/s (Jensen) to 1.44×10^{-2} m/s (TurbOPark). The U-Net/MLP model also successfully predicted turbulence intensity, achieving MAEs between 1.30×10^{-4} (Frandsen model) and 2.21×10^{-4} (Crespo-Hernández model).

Crucially, neural networks significantly outperform traditional engineering models in computational efficiency. While engineering models' computational time scales linearly with turbine count, neural networks maintain a constant execution time of approximately 3 ms, regardless of wind park size. This breakthrough enables rapid assessment of large-scale wind farm layouts and performance optimization.

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Motivation . . . 3
- 1.3 Objectives . . . 5

2 Literature Review **7**
- 2.1 Wind Farm Wake Effects . . . 7
- 2.2 Analytical Wake Models . . . 9
  - 2.2.1 Jensen Wake Deficit Model . . . 11
  - 2.2.2 Bastankhah Wake Deficit Model . . . 12
  - 2.2.3 TurbOPark Wake Deficit Model . . . 12
  - 2.2.4 Wake-generated Turbulence Model . . . 13
  - 2.2.5 Wake Superposition Methods . . . 15
- 2.3 Neural Networks . . . 16
  - 2.3.1 Activation Functions . . . 17
  - 2.3.2 Adam Optimization Algorithm . . . 19
- 2.4 Convolutional Neural Networks . . . 20
  - 2.4.1 Convolutional Layers . . . 21
  - 2.4.2 Max Pooling . . . 22
  - 2.4.3 Dropout Regularization . . . 22
  - 2.4.4 Convolutional Autoencoders . . . 23
  - 2.4.5 U-Net . . . 23
  - 2.4.6 Networks with Multilayer Perceptrons . . . 24
- 2.5 Machine Learning in Wind Energy . . . 25
  - 2.5.1 Wind Farm Wake Prediction . . . 26
  - 2.5.2 Prediction of Airfoil Aerodynamic Performance . . . 27
  - 2.5.3 Wind Farm Power Predictions . . . 27

3 Methodology **29**
- 3.1 General Workflow . . . 29
- 3.2 Data Collection and Preprocessing . . . 29
  - 3.2.1 Wind Park Layout Generation . . . 30
  - 3.2.2 Wind Park Rasterisation . . . 35
- 3.3 Analytical Wake Models . . . 37
  - 3.3.1 PyWake Model Settings . . . 37
  - 3.3.2 Wind Turbine Model . . . 38
- 3.4 Neural Network Architectures . . . 38
  - 3.4.1 Convolutional Autoencoder . . . 39
  - 3.4.2 Convolutional Autoencoder with MLP . . . 39
  - 3.4.3 U-Net . . . 40
  - 3.4.4 U-Net with MLP . . . 41
- 3.5 Training and Validation . . . 41
  - 3.5.1 Loss Function Selection and Weighting . . . 41
  - 3.5.2 Data Transformation . . . 44
  - 3.5.3 Hyperparameters . . . 46

4 Results and Analysis **47**
4.1 Performance Comparison of Wake Models . . . 47
4.2 Neural Network Architecture Evaluation . . . 48
4.2.1 Error Analysis . . . 50
4.3 Long-Distance Flow Prediction Accuracy . . . 51
4.3.1 Jensen Dataset . . . 52
4.3.2 Bastankhah Dataset . . . 54
4.3.3 TurbOPark Dataset . . . 56
4.3.4 Wake-generated Turbulence Prediction . . . 58
4.4 Computational Efficiency . . . 61
4.4.1 Training Time . . . 64

5 Discussion **65**
5.1 Implications for Wind Farm Design . . . 65
5.2 Limitations of the Study . . . 67
5.3 Future Research Directions . . . 67

6 Conclusion **69**

Nomenclature **70**

A Graph Neural Networks: Foundations and Applications **82**
A.1 Graph Convolutional Network / Convolutional Neural Network Architecture . . . 82
A.2 Preliminary Results of the GCN-CNN Architecture . . . 83

B Wind Park Generation **86**

C Wake Deficit Prediction in 2D **88**
C.1 Convolutional Autoencoder . . . 88
C.1.1 Jensen Wake Deficit Model . . . 88
C.1.2 Bastankhah Wake Deficit Model . . . 89
C.1.3 TurbOPark Wake Deficit Model . . . 90
C.2 U-Net . . . 91
C.2.1 Jensen Wake Deficit Model . . . 91
C.2.2 Bastankhah Wake Deficit Model . . . 92
C.2.3 TurbOPark Wake Deficit Model . . . 93
C.3 Convolutional Autoencoder with Multilayer Perceptron . . . 94
C.3.1 Jensen Wake Deficit Model . . . 94
C.3.2 Bastankhah Wake Deficit Model . . . 95
C.3.3 TurbOPark Wake Deficit Model . . . 96
C.4 U-Net with Multilayer Perceptron . . . 97
C.4.1 Jensen Wake Deficit Model . . . 97
C.4.2 Bastankhah Wake Deficit Model . . . 98
C.4.3 TurbOPark Wake Deficit Model . . . 99
C.5 Wake-generated Turbulence Prediction . . . 100
C.5.1 Crespo-Hernández Model . . . 100
C.5.2 Frandsen Model (2005) . . . 101
C.5.3 Frandsen Model (2017) . . . 102

D Wind Park Layouts **102**
D.1 Training Dataset Wind Parks 104
D.2 Test Dataset Wind Parks 112

1 Introduction

Wind energy has become a cornerstone of the global transition to sustainable power generation. As the industry expands rapidly, particularly in offshore environments, understanding and modeling wind farm dynamics have become increasingly crucial. This section explores the background, motivation, and objectives of a study aimed at improving wind farm flow modeling using advanced machine learning techniques.

The background subsection provides an overview of wind energy's growing importance and the challenges posed by wake effects in wind farms. It discusses the limitations of current modeling approaches and introduces the potential of machine learning, specifically Convolutional Neural Network (CNN), in addressing these challenges. The motivation subsection highlights the pressing need for accurate and computationally efficient wake models, particularly in the context of dense offshore wind farm development. It uses the North Sea as a prime example of the industry's rapid growth and the complex planning required for future expansions.

Finally, the objectives subsection outlines the specific goals of the research. These include developing and evaluating CNN-based models for long-distance wind farm flow prediction, assessing various neural network architectures, and exploring the models' potential applications in wind farm design and operation.

1.1 Background

Wind energy has emerged as a crucial renewable resource in the global shift towards sustainable power generation. As the world seeks to reduce its reliance on fossil fuels and combat climate change, wind farms have become a significant contributor to electricity grids worldwide (see Figure 1.1). This growth is not limited to any specific region; both onshore and offshore installations are expanding rapidly, driven by technological advancements, cost reductions, and supportive environmental policies [45, 37]. Wind energy's role in the modern energy landscape underscores its importance in achieving a sustainable and resilient energy future.

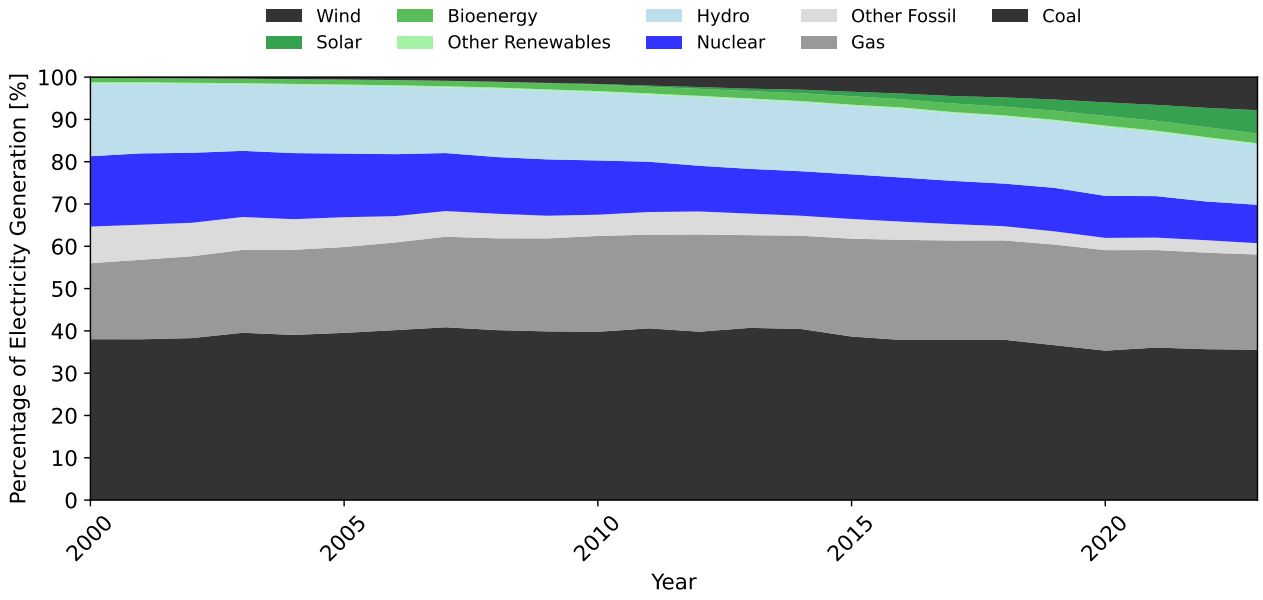


Figure 1.1: Percentage share of world electricity generation by source (Data Source: [28]).

Wind energy has emerged as a crucial renewable resource in the global transition towards sustainable power generation. However, the efficient design and operation of wind farms present significant challenges, particularly in understanding and mitigating wake effects. When wind turbines extract energy from the air, they create zones of reduced wind speed and increased turbulence downstream, known as wakes. These wakes can extend for several kilometers, impacting the performance of downwind turbines and neighboring wind farms [60, 62]. Wake effects occur both internally within wind farms [90, 15] and between neighboring installations, leading to substantial power losses and increased fatigue loads on turbine components. The accurate modeling of wake effects, both within individual wind farms and between neighboring installations, is paramount for optimizing wind farm layouts, predicting power output, and managing turbine operations. Traditional approaches have relied on analytical wake models, which offer simplicity but often lack accuracy over large distances. As wind farms are increasingly built in larger clusters, it has become important to understand and accurately model inter-farm wakes, also known as cluster wakes [31].

Recent studies have shown that cluster wakes can persist over remarkable distances, being observed 55 km downstream using Synthetic Aperture Radar (SAR) and single lidar scans [79]. These long-distance wakes have been observed through various methods, including satellite-based SAR [44], aircraft measurements [71], high-fidelity Large Eddy Simulation (LES) simulations [19], dual-Doppler radar [63], and analysis of wind farm production data [64]. There is growing consensus within the scientific community that the impact of neighboring wind farms has been historically underestimated, leading to an overprediction bias in wind energy production estimates [69]. For instance, Fischereit et al. [31] investigated the wake impact of the Nysted wind farm on the Rødsand II wind farm at wind speeds around 10 m/s. Their study revealed that engineering wake models designed for wind farm production estimates and layout optimization predicted power reductions between 0.1% and 4.5%. In contrast, high-fidelity computational fluid dynamics models showed significantly higher reductions between 8.9% and 12.2%.

This discrepancy highlights the need for advanced modeling techniques that can accurately capture the complex dynamics of long-distance wake effects. Such models are essential for optimizing wind farm layouts, predicting power output with greater precision, and developing effective strategies for wake mitigation. Computational Fluid Dynamics (CFD) simulations are powerful tools for modeling wind farm flows, offering high-fidelity results that capture intricate wake interactions and atmospheric boundary layer effects. These simulations employ various numerical methods to approximate solutions to the Navier-Stokes equations, typically using either Reynolds-averaged Navier-Stokes (RANS) models or LES techniques. RANS models are based on time-averaged equations and assume that turbulent fluctuations can be modeled as separate terms, while LES directly simulates larger turbulent structures and models smaller scales. For instance, Stieren and Stevens [84] used LES to calculate wake effects between two aligned wind farms. These approaches provide valuable insights into wake behavior, turbulence characteristics, and energy transfer processes within wind farms, balancing computational cost with the level of detail required.

CFD models can account for complex terrain, atmospheric stability, and turbine-specific features, making them valuable for both research and industry applications. However, the computational intensity of CFD simulations poses significant challenges for their widespread adoption in operational settings. High-resolution CFD models often require substantial computing resources and long run times, making them impractical for real-time applications or rapid iterative design processes [80], for example by wind farm layout optimization. This limitation has spurred

the search for alternative approaches that can balance accuracy with computational efficiency. Recent advances in machine learning, particularly in the field of CNN, offer new possibilities for modeling complex fluid dynamics problems [40, 76, 35]. These data-driven approaches have shown promise in capturing the non-linear behavior of fluid flows, potentially bridging the gap between simple analytical models and resource-intensive CFD simulations.

This research aims to explore the application of machine learning techniques, specifically CNNs, to improve long-distance wind farm flow modeling. By leveraging data from existing wake models (see subsection 2.2), the goal is to develop accurate and computationally efficient models to predict wake effects on large spatial scales. As the accuracy of these models depends on their training data, future improvements could be achieved by incorporating higher fidelity datasets.

1.2 Motivation

The increasing density of offshore wind parks presents an important challenge in optimizing layouts to minimize wake effects. Turbine wakes significantly reduce energy output and increase mechanical stress on downstream turbines, leading to higher maintenance costs and shorter lifespans. This situation necessitates the development of accurate and computationally efficient wake deficit models for effective wind farm layout optimization.

Conventional wake models often struggle to provide the accuracy needed for complex, densely arranged wind farms [6, 31]. While CFD models have the potential for greater precision, their computational demands make them impractical for assessing numerous inflow conditions and layouts [19]. This study aims to bridge this gap by developing a reliable and computationally efficient wake deficit model. The proposed model is designed to handle wake deficit propagation calculations for multiple large-scale offshore wind farms, potentially enabling more comprehensive wind park layout optimization and sensitivity analyses. This would allow for a better understanding of how various design parameters impact overall performance.

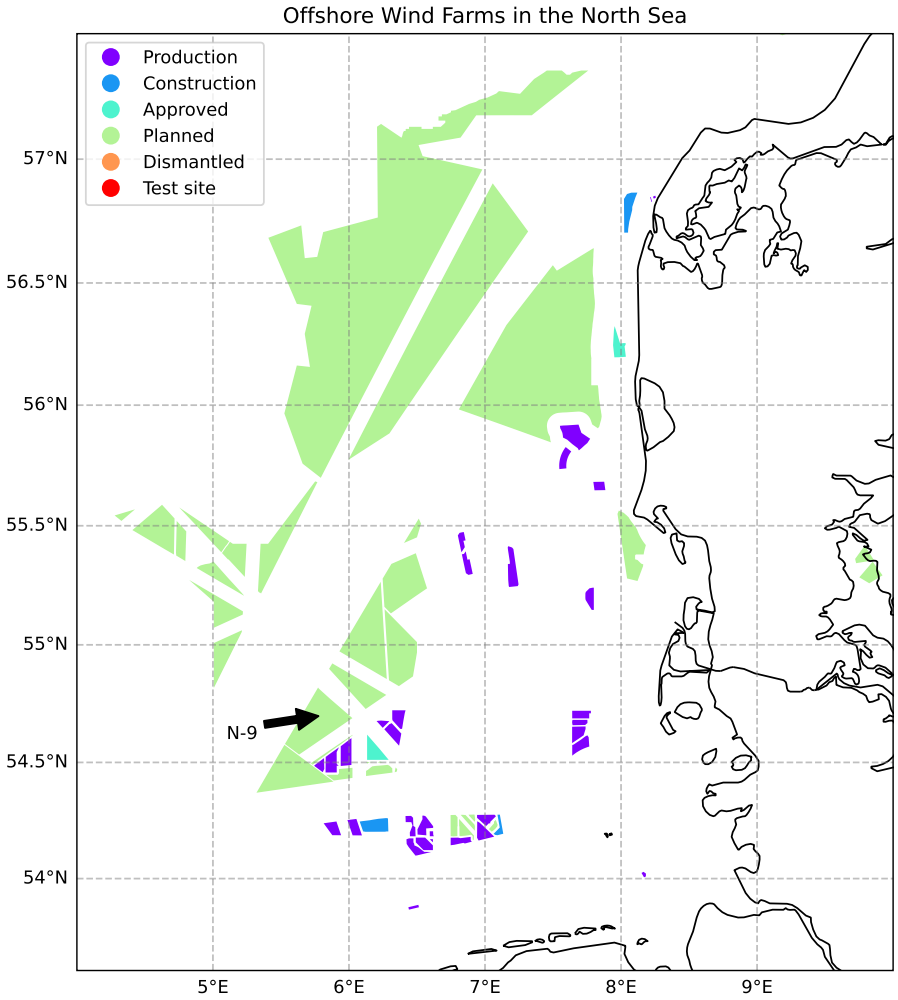


Figure 1.2: Overview of the constructed and planned wind designation areas in the eastern North Sea (Data Source: European Marine Observation and Data Network [30]).

The North Sea exemplifies the rapid growth and future potential of offshore wind energy. It already hosts some of the world's largest offshore wind parks. Notable examples include Hornsea One and Two, with a combined nameplate capacity of 2.6 GW [67]. Another significant development is the Dogger Bank wind farm cluster (A, B, C) and Sofia Offshore Wind Farm, projected to reach a total capacity of 5.1 GW upon completion [25, 82]. These massive wind farms are located off the coasts of the United Kingdom, Belgium, the Netherlands, Germany, and Denmark. As shown in Figure 1.2, a significant portion of the eastern North Sea is already occupied by operational wind parks. Even more striking is the vast area designated in the Marine Spatial Planning (MSP) for future wind energy development.
The competitive nature of the offshore wind industry further underscores the importance of efficient wake modeling. For instance, in the German Exclusive Economic Zone (EEZ), the designation area N-9 has been reserved for wind energy development and subdivided into sites N-9.1, N-9.2, and N-9.3 (see Figure 1.3). These sites, covering approximately 421 km^2 with an expected capacity of 5500 MW, were put out for tender on March 1, 2024 Memija [56]. For such large-scale tenders, energy companies must submit comprehensive proposals based on extensive simulations, emphasizing the need for efficient modeling tools that can rapidly evaluate different layouts and turbine selections.

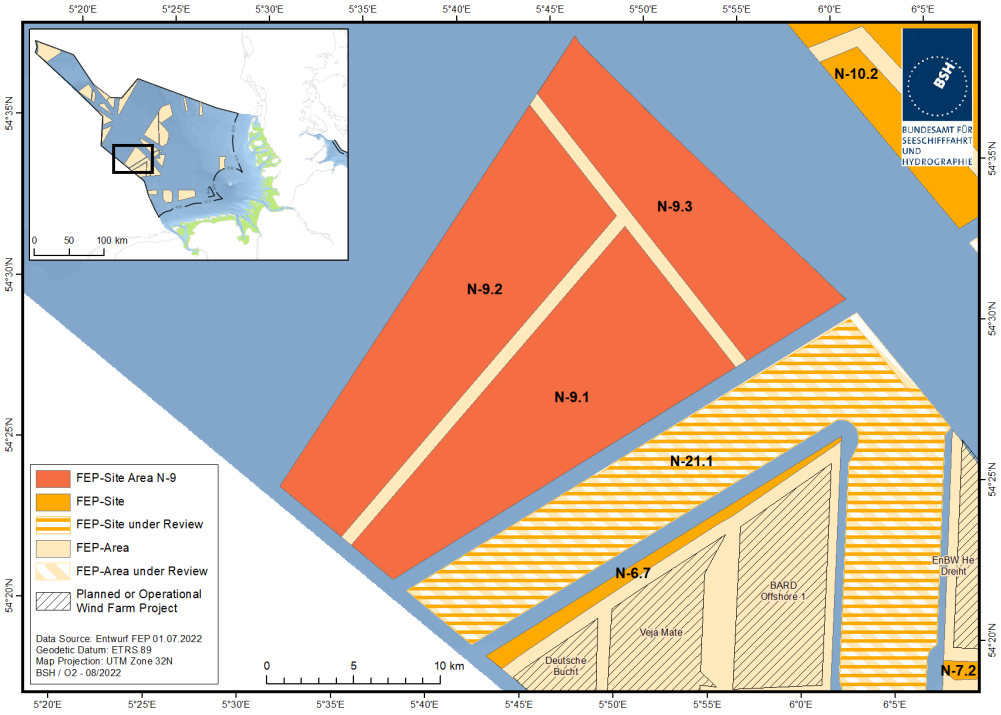


Figure 1.3: Overview of wind development designation area N-9 in the German EEZ (Source Bundesamt für Seeschifffahrt und Hydrographie [17]).

Recent advancements in machine learning, particularly in CNNs, offer a promising solution to wind wake modeling challenges. CNNs can learn complex spatial patterns from large datasets, making them well-suited for this task. This research aims to develop a neural network-based wake model that combines the accuracy of higher-fidelity models with the computational efficiency required for practical, large-scale use. In collaboration with Ørsted, this study focuses on applying CNNs and other advanced neural network architectures to predict wind farm wake effects. The goal is to create a robust wake model that demonstrates good accuracy compared to its training data. To achieve this, the research employs multiple engineering wake models and various neural network architectures to determine the most effective approach for long-distance wind farm flow modeling. This comparative study aims to identify whether combining traditional wake models with machine learning techniques can lead to accurate and efficient wake prediction. The study also considers the potential for future research to train these models on higher-fidelity wake models, potentially enhancing their predictive capabilities.

1.3 Objectives

The main objectives of this thesis are to develop and evaluate CNN-based models for predicting wind farm flow over long distances. The research focuses on designing and implementing neural network architectures that can process wind farm layout and inflow conditions to predict wake effects up to 768 rotor diameters downstream. This large distance is chosen based on observed wind farm wake effects at comparable scales, as discussed in subsection 1.1.

A key aspect of the study is evaluating how neural networks perform when trained on various datasets. The research utilizes three engineering wake models: the Jensen wake deficit model, Bastankhah model, and TurbOPark model, which are detailed in subsection 2.2. The study assesses several neural network architectures for wake modeling, including a Convolutional Autoencoder (CAE), U-Net, and hybrid designs combining CAE or U-Net with a Multilayer Perceptron (MLP) layer. These neural network architectures are covered in subsection 3.4. By

analyzing the strengths and weaknesses of each architecture in capturing wake dynamics, the research aims to identify the most effective approach for this application.

Another crucial objective is to evaluate the accuracy and computational efficiency of the developed models. This involves comparing CNN predictions against test datasets for each of the analytical models. The study also assesses the computational requirements and inference speed of the trained models. The research explores the potential applications of these models in improving wind farm design and operation. Specifically, it investigates their use in wind farm layout optimization and yield assessment analysis.

The study also aims to quantify model uncertainty and assess generalization capabilities. This is achieved by evaluating model performance across various wind farm configurations and inflow conditions. Through these objectives, this research seeks to advance wind farm flow modeling. The goal is to provide tools that balance accuracy, computational efficiency, and practical applicability for the wind energy industry.

2 Literature Review

This section presents a comprehensive review of the current literature on wind farm wake effects and the application of machine learning in wind energy. The review begins with an in-depth analysis of the complex aerodynamic phenomena associated with wind turbine wake effects, emphasizing their critical influence on wind farm performance and efficiency. A detailed examination follows of the engineering wake deficit models currently employed in the industry for predicting wake effects within wind farms and between neighboring wind farms. The section highlights the equations underlying three key wake deficit models: Jensen, Bastankhah, and Turbulence Optimized Park (TurbOPark). Each model's complexity and accuracy in typical use cases are discussed, providing insight into their strengths and limitations.
The review then explores the integration of advanced machine learning techniques in wake effect modeling and prediction. Particular attention is given to Convolutional Autoencoders (CAEs) and U-Net architectures, discussing their potential to enhance the accuracy and efficiency of wake effect analysis. This exploration extends to the broader applications of machine learning in wind energy, encompassing wind farm wake prediction, airfoil aerodynamic performance prediction, and wind farm power forecasting. By examining current knowledge and identifying gaps in existing research, this literature review establishes a solid foundation for the subsequent research and analysis presented in this study. It aims to contextualize the importance of both traditional engineering models and emerging machine learning approaches in advancing our understanding and prediction of wind farm wake effects.

2.1 Wind Farm Wake Effects

Wind turbine wake effects refer to the complex aerodynamic phenomena that occur when wind flows through a turbine, causing a wake area of reduced wind speed and increased turbulence downstream. Understanding these wake effects is crucial for optimizing wind farm performance, as they significantly influence the efficiency [60] and lifespan [85] of turbines within the farm and neighbouring wind farms. When wind passes through a wind turbine, the kinetic energy of the wind is transferred to the rotor blades, which convert this kinetic energy into mechanical energy. The rotor blades, connected to a generator, then convert the mechanical energy into electrical energy. This process of energy extraction reduces the kinetic energy in the wind flow, resulting in a wake region of slower-moving, turbulent air immediately downstream of the turbine. For wind farms, this effect is amplified, as the individual turbine wakes are mixed leading to a large scale wind farm wake deficit effect downwind, which consists of the superposition (see subsubsection 2.2.5) of the individual turbine wakes.

This phenomenon has been famously observed at the Horns Rev Offshore Wind Park, as shown in Figure 2.1. The image clearly demonstrates how the air behind wind turbines becomes turbulent, creating expanding wakes that are visible as clouds downwind from the turbines. As covered by Hasager et al. [43], wind turbine wakes are typically invisible, but became visible in this instance due to a unique atmospheric phenomenon. A temperature inversion occurred when a layer of cold humid air resided above a warmer sea surface. The wind passing through the turbines generated a counter-rotating swirl, causing warm, humid air from the sea surface to be up-drafted. This warm air mixed with the cold air layer above, resulting in fog condensation within the turbine wakes. The condensation was particularly prominent in wake areas with high axial velocities and turbulent kinetic energy. Consequently, the usually imperceptible turbine wakes became clearly visible as spiraling bands, revealing the large-scale structure of the wake.

Figure 2.1: Wakes in the wind farm Horns Rev 1 [February 12th 2008]. (Owned by: Vattenfall. Photographer: Christian Steiness)

The primary characteristic of a wake is a reduction in wind speed. As the wind energy is harnessed by the turbine blades, the wind exiting the rotor has significantly less kinetic energy. This reduction in speed is most pronounced directly behind the turbine and gradually recovers to ambient wind speeds as the air mass moves further downstream. The extent of these wake regions can span several kilometres, with their length influenced by factors such as wind speed, atmospheric conditions, and turbine operating parameters. Studies have observed wake effects at scales beyond 50 kilometres [44]. Wake regions are characterized not only by reduced wind speed but also by increased turbulence. The interaction between rotor blades and wind flow generates turbulent eddies and vortices, resulting in higher turbulence intensity within the wake. This increased turbulence amplifies the mechanical loads on downstream turbines, potentially leading to accelerated wear and tear on turbine components. Thomsen and Sørensen [85] demonstrated this effect, finding a 5% to 15% increase in fatigue loads for wind turbines operating in wakes compared to those in freestream air.

As seen from Figure 2.2, the wake region behind a wind turbine can be divided into two distinct zones: the near wake and the far wake. The near wake begins immediately behind the turbine and extends approximately 2–4 rotor diameters (D) downstream [32, 55]. This region is strongly influenced by the rotor geometry, characterized by the formation of blade tip vortices, steep pressure and axial velocity gradients, and wake expansion. In contrast, the far wake is primarily defined by reduced wind speeds and increased turbulence intensities, with limited influence from the specific rotor geometry [38]. Turbulence becomes the dominant physical property in this region [23]. The far wake is affected not only by rotor-induced turbulence but also by large-scale (or atmospheric) turbulence further downstream. This turbulent mixing accelerates wake recovery in terms of both velocity deficit and turbulence intensity. As the wake develops, the velocity deficit in the far wake approaches a Gaussian profile, which is axisymmetric and self-similar [8].

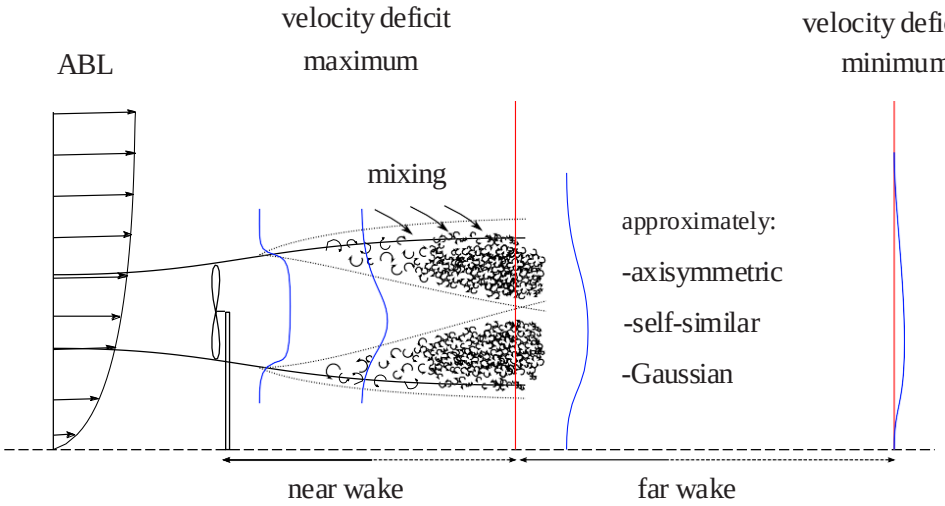


Figure 2.2: Wind turbine wakes in the Atmospheric Boundary Layer (ABL). Mixing of the ABL with turbine wakes above the turbine region leads to wake recovery further downstream (Reprinted from Sanderse [78]).

The wake effects described are not limited to interactions between individual turbines within a single wind farm. As wind farms grow larger and more numerous, the collective impact of multiple turbines creates a wind farm scale wake that can affect neighboring wind farms. This large-scale wake is the result of the superposition of individual turbine wakes, leading to a more extensive area of reduced wind speeds and increased turbulence. Recent studies have shown that these wind farm wake effects can extend far beyond the immediate vicinity of a wind farm, potentially impacting the performance of other wind farms located tens of kilometers downstream. For instance, Nygaard et al. [64] observed wake effects between offshore wind farms separated by distances of up to 15 km. These inter-farm wake interactions can result in noticeable power losses for downstream wind farms, though the exact magnitude can vary significantly depending on factors such as farm layout, atmospheric conditions, and distance between farms.

Understanding and mitigating these large-scale wake effects has become increasingly important as the density of wind farms continues to grow, particularly in offshore environments where space is at a premium. Accurate modeling and prediction of wind farm wakes are crucial for optimizing the layout and operation of wind farms, as well as for assessing the potential impacts on existing and planned wind energy projects in the vicinity.

2.2 Analytical Wake Models

Analytical wake models are mathematical frameworks designed to predict the behavior of wakes generated by wind turbines. These models are essential for understanding and optimizing the performance of wind farms, as they provide insights into the wind speed deficits and turbulence characteristics that affect downstream turbines. By simplifying the complex aerodynamic interactions into manageable equations, analytical wake models offer a balance between computational efficiency and accuracy, making them widely used in wind farm design and operational planning. The primary purpose of analytical wake models is to estimate the impact of a wind turbine's wake on the wind flow and subsequently on the performance of other turbines within the same wind farm and between neighbouring wind farms.

There are several well-known analytical wake models, each with its own approach and assumptions. Some of the earlier models, like the Jensen model [46] and the Larsen model [51], assume

a top-hat shaped velocity deficit profile with linear and non-linear wake expansion, respectively. These models are relatively simple and computationally efficient, making them popular for initial wind farm design stages. The Frandsen model [33] takes a different approach, using an area overlap method to assess wake interactions at the wind farm scale. This model is particularly useful for layout optimization in large wind farms.

More recent models have introduced more sophisticated wake profiles. The Bastankhah and Porté-Agel [9] model proposes a Gaussian wake profile, providing a more realistic representation of the wake velocity deficit distribution. This self-similar solution offers improved accuracy while maintaining computational efficiency. The Ainslie model [2] employs an eddy-viscosity approach, solving simplified Navier-Stokes equations. While more computationally intensive, it offers a more physics-based representation of wake development from the near to far wake regions. Building upon the Gaussian wake concept, Blondel and Cathelain [14] introduced a super-Gaussian profile that accounts for wake rotation effects. This model provides a flexible framework that can capture various wake shapes observed in different atmospheric conditions. Each of these models represents a different trade-off between simplicity, computational efficiency, and physical accuracy. The TurbOPark model was developed in response to Ørsted's discovery in 2019 that industry-standard wake models were underestimating the effects of wakes from neighboring wind farms [66]. This finding, based on analysis of production data from Ørsted's offshore wind farm portfolio, led to Ørsted's development of a more accurate model capable of predicting wake effects over longer distances. TurbOPark was designed to accurately capture both internal wind farm wakes and cluster wakes from neighboring wind farms, addressing the gap in the industry's ability to forecast wind farm performance, particularly in scenarios where multiple offshore wind farms are in close proximity [69].

The choice of wind farm wake deficit model depends on the specific application, available computational resources, and required level of detail in wake representation. For this study, the Jensen, Bastankhah and TurbOPark wake deficit models are used, which can been seen compared in Figure 2.3.

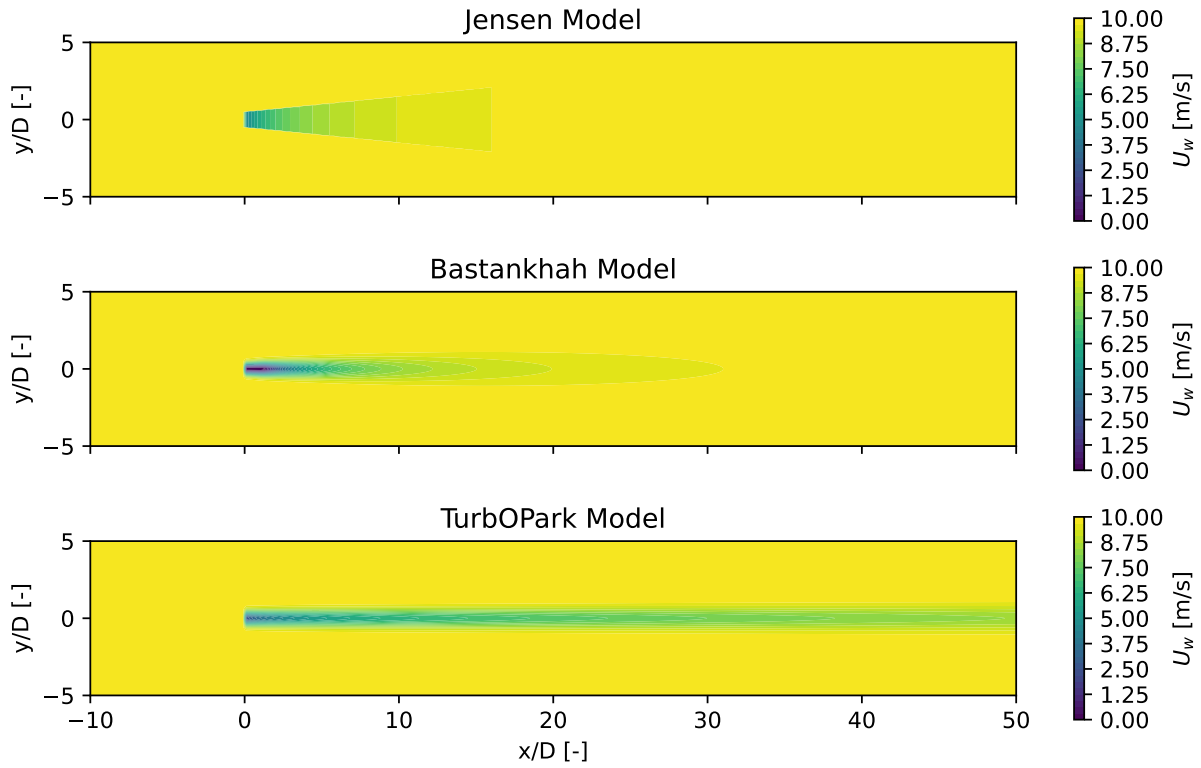


Figure 2.3: Comparison of wake deficit downwind from a single Vestas V80 wind turbine at wind speed $U_0 = 10\,\mathrm{m/s}$ and ambient turbulence intensity of $I_0 = 0.1$. The models used are the Jensen, Bastankhah and TurbOPark wake deficit models.

2.2.1 Jensen Wake Deficit Model

The Jensen wake deficit model, formulated by Jensen [46] in 1983, presents a simplified approach to understanding the wake effects behind a wind turbine. The core assumption of the Jensen model is that the wake behind a wind turbine expands linearly with distance downstream from the turbine. The wake radius r is proportional to the downwind distance x:

$$r = r_0 + k_w x \tag{1}$$

where r_0 is the rotor radius and k_w is the wake decay constant.
The velocity deficit in the wake can be expressed as:

$$\delta(x) = \frac{U_0 - U_w}{U_0} = 2a \left(\frac{r_0}{r_0 + k_w x} \right)^2 \tag{2}$$

where U_w is the velocity in the wake at distance x from the turbine, U_0 is the ambient wind velocity, and a is the axial induction factor. Assuming ideal conditions of axial symmetry, without rotation, turbulence, and with a conic-shaped wake profile, the axial induction factor can be expressed as:

$$a = \frac{1 - \sqrt{1 - C_T}}{2} \tag{3}$$

Katic et al. [47] extended this model to calculate wake effects from multiple turbines. Their approach propagates the wake downwind and uses the Sum of Squares (SOS) of individual wake deficits. The model accounts for partial wake-rotor overlap by considering the incident wind speed. This modified Jensen model, later adopted by Wind Atlas Analysis and Application

Program (WAsP), became known as the original Park model. Due to its simplicity and low computational cost, the Jensen wake deficit model is widely used for preliminary wind farm design and optimization.

2.2.2 Bastankhah Wake Deficit Model

The Bastankhah wake deficit model [9] is an analytical model proposed to predict the wind velocity distribution downwind of a wind turbine. This model is derived using the principles of mass and momentum conservation and assumes a Gaussian distribution for the velocity deficit in the wake. The normalized velocity deficit δ is defined with a Gaussian distribution:

$$\delta(x) = C(x) \exp\left(-\frac{r^2}{2\sigma^2}\right) \tag{4}$$

Here σ is the standard deviation of the Gaussian profile. The function for C represents the maximum normalized velocity deficit at the center of the wake for each downwind location and is given by:

$$C(x) = 1 - \sqrt{1 - \frac{C_T}{8(\sigma/D_0)^2}} \tag{5}$$

If we assume a linear expansion for the wake region similar to Jensen [46], σ/D_0 can be written as $\sigma/D_0 = k^*(x/D_0) + \epsilon$, where $k^* = \partial\sigma/\partial x$ is the wake growth rate and ϵ is the value of σ/D_0 as x approaches zero. The final equation for the normalized velocity deficit is:

$$\delta(x) = \left(1 - \sqrt{1 - \frac{C_T}{8(k^*x/D_0 + \epsilon)^2}}\right) \exp\left(-\frac{1}{2(k^*x/D_0 + \epsilon)^2}\left(\left(\frac{z - z_H}{D_0}\right)^2 + \left(\frac{y}{D_0}\right)^2\right)\right) \tag{6}$$

This equation describes the normalized velocity deficit in the wake as a function of the downwind distance, spanwise and vertical coordinates, thrust coefficient, and wake growth rate. While the Bastankhah model offers a straightforward approach to estimating wind turbine wakes, recent research has provided mixed results regarding its performance. Krutova et al. [50] found it to perform comparably to similar engineering wake deficit models. However, a more recent study by Binsbergen et al. [12] suggests that its performance may be inferior to other comparable models.

2.2.3 TurbOPark Wake Deficit Model

The TurbOPark model is an advanced wake model developed by Ørsted [64, 69, 61] to accurately predict wake effects within wind farms and between neighboring wind farms. It improves upon the classic Park model (see subsubsection 2.2.1) by incorporating both ambient atmospheric turbulence and wake-generated turbulence, resulting in a more realistic description of wake recovery dynamics. The key innovation of the TurbOPark model is its turbulence-dependent wake expansion rate, expressed as:

$$\frac{d\sigma_w}{dx} = AI(x) \tag{7}$$

where σ_w is the characteristic wake width, x is the distance downstream of the turbine, A is a wake expansion calibration parameter, and I is the local turbulence intensity. The local turbulence intensity combines ambient and wake-generated turbulence:

$$I(x) = \sqrt{{I_0}^2 + {I_{add}(x)}^2} \tag{8}$$

where I_0 is the ambient turbulence intensity and I_{add} is the additional wake-generated turbulence. The wake-generated turbulence is modeled using the Frandsen wake turbulence model [34]:

$$I_{add}(x) = \frac{1}{c_1 + c_2 \frac{x/D_0}{\sqrt{C_T(V_{in})}}} \tag{9}$$

where $c_1 = 1.5$ and $c_2 = 0.8$ are empirical constants, D_0 is the rotor diameter, and $C_T(V_{in})$ is the thrust coefficient at the inflow wind speed V_{in}. The wind speed deficit is described using the Gaussian wake profile proposed by Bastankhah and Porté-Agel [9] in Equation 4, but using the characteristic wake width $\sigma_w(x)$. The maximum normalized velocity deficit C, derived from conservation of momentum, is formulated similar to Bastankhah and Porté-Agel [9] and can be seen in Equation 5. The unique feature of TurbOPark is its non-linear wake expansion, which evolves according to:

$$\frac{\sigma_w(x)}{D_0} = \epsilon + \frac{AI_0}{\beta}\left(\sqrt{(\alpha+\beta x/D_0)^2+1} - \sqrt{1+\alpha^2} - \ln\left[\frac{(\sqrt{(\alpha+\beta x/D_0)^2+1}+1)\alpha}{(\sqrt{1+\alpha^2}+1)(\alpha+\beta x/D_0)}\right]\right) \tag{10}$$

where ϵ is the initial normalized characteristic wake width, $\alpha = c_1 I_0$ and $\beta = \frac{c_2 I_0}{\sqrt{C_T(V_{in})}}$. Based on tuning and validation using data from 19 offshore wind farms, the recommended value for the wake expansion calibration parameter is $A = 0.04$ for all offshore applications [61]. For turbines affected by multiple wakes, the deficits from overlapping wakes are combined using the SOS method.

The TurbOPark model offers several key advantages. It significantly improves predictions for cluster wakes and long-distance wake interactions, addressing limitations of the Park model at larger distances [69]. The integration of turbulence effects results in a more accurate representation of wake recovery dynamics. The TurbOPark model is open-source and available online [61]. For this study, TurbOPark, along with the other engineering wake deficit models, was implemented using PyWake [74].

2.2.4 Wake-generated Turbulence Model

Crespo-Hernández turbulence model: The Crespo-Hernández model, developed by Crespo and Hernández [23] in 1996, provides a set of analytical expressions to estimate turbulence characteristics in wind turbine wakes. This model distinguishes between near wake (1-3 rotor diameters downstream) and far wake (5-15 rotor diameters downstream) regions, offering separate correlations for each. For the near wake, the model proposes a simple expression for the turbulence intensity added in the wake:

$$I_{add} = 0.362[1 - (1 - C_T)^{1/2}] \tag{11}$$

where C_T is the thrust coefficient. This equation is derived from a simplified analysis of the annular shear layer formed immediately behind the turbine, where maximum turbulence production occurs in the upper part of the wake. In the far wake region, the model suggests a more complex correlation based on numerical simulations:

$$I_{add} = 0.73 a^{0.8325} {I_0}^{-0.0325} (x/D_0)^{-0.32} \tag{12}$$

where a is the induction factor, I_0 is the ambient turbulence intensity, x is the downstream distance, and D_0 is the rotor diameter. This expression accounts for the gradual decay of added turbulence with distance from the turbine. It's worth noting that in the PyWake implementation [70], Equation 12 is applied to model turbulence intensity throughout the entire wake, including the near wake region, despite the original model's distinction between near and far wake correlations.

The Crespo-Hernández model has been widely adopted in the wind energy community due to its relative simplicity and reasonable accuracy, making it a valuable tool for initial assessments of wake turbulence in wind farm design and optimization.
In the Crespo-Hernández model, the turbulence intensity I at a distance x downstream of a wind turbine is calculated by combining the ambient atmospheric turbulence intensity I_0 with the additional wake-induced turbulence I_{add}. These two sources of turbulence are added in quadrature, as seen in Equation 8.

Frandsen turbulence model: The Frandsen turbulence model, developed by Frandsen [34], offers an empirical approach to estimating turbulence intensity in wind turbine wakes. This model is particularly useful for wind farm layout design and analysis due to its straightforward methodology. The wake turbulence intensity is calculated by combining the added wake turbulence and the ambient turbulence intensity, similar to the Crespo-Hernández model, by taking the square root of the sum of their squared values as seen in Equation 8. This approach assumes that the ambient and wake-induced turbulence are uncorrelated, which is a reasonable approximation for most practical purposes. The additional turbulence in the wake, I_{add}, is described empirically as [34]:

$$I_{add} = \frac{1}{1.5 + 0.3\frac{x}{D_0}\sqrt{V_{in}}} \tag{13}$$

where x is the downstream distance from the turbine, D_0 is the rotor diameter, and V_{in} is the hub height wind speed. The model has been implemented by the International Electrotechnical Commission (IEC) in different iterations. In 2005 [88], the added turbulence intensity was defined as:

$$I_{add} = \frac{0.9}{1.5 + 0.3\frac{x}{D_0}\sqrt{V_{in}}} \tag{14}$$

Later, in 2019 [89], the formulation was changed to the same formula found in Equation 9. In Frandsen's original thesis, a bell-shaped turbulence wake profile was formulated to reflect the smoothed-out distribution of turbulence intensity due to rotor-averaging. The turbulence intensity I experienced at the center of the wake-affected rotor is given by:

$$I(\theta) = I_0 \left[1 + \alpha \exp\left(- \left(\frac{\theta}{\theta_w} \right)^2 \right) \right] \tag{15}$$

Here, I_0 is the ambient turbulence intensity, α is a constant related to the maximum added wake turbulence intensity, θ is the angle between the connection line of the two wind turbine units and the wind direction, and θ_w is the characteristic view-angle of the wake-generating unit and s is the separation distance between the turbines. The formulations in Equation 14 and Equation 9 are implemented in PyWake as *STF2005TurbulenceModel* and *STF2017TurbulenceModel* respectively. These versions of the Frandsen model, along with the bell-shaped wake distribution defined in Equation 15, have been used in later sections to estimate the added

turbulence intensity downwind from wind farms.

As illustrated in Figure 2.4, the Crespo-Hernández model exhibits a simpler top-hat shape and a slower dissipation rate compared to the two Frandsen model formulations, which show a bell-shaped wake profile. The differences between the Frandsen (2005) and Frandsen (2017) models for a single turbine are relatively minor.

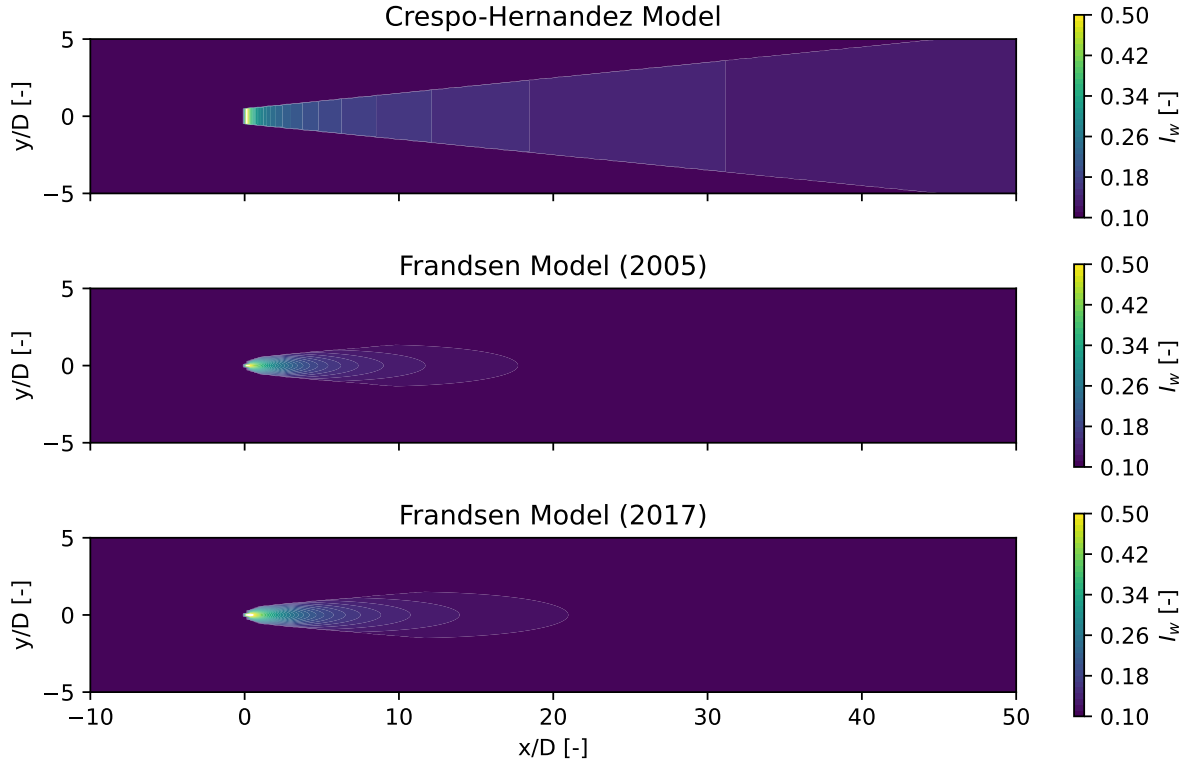


Figure 2.4: Comparison of wake deficit downwind from a single Vestas V80 wind turbine at a wind speed $U_0 = 10\,\mathrm{m/s}$ and ambient turbulence intensity of $I_0 = 0.1$. The models used are the Crespo-Hernández, Frandsen (2005) and Frandsen (2017) wake turbulence models.

2.2.5 Wake Superposition Methods

Wake superposition is a important component of wake modeling in wind farm analysis. When calculating the total velocity deficit from multiple upstream turbines, two primary approaches are commonly employed in the literature, as noted by Bastankhah et al. [10]:

$$\textbf{Linear sum:} \quad \delta(x) = \sum_{i=1}^{n} \Delta U_i \tag{16}$$

$$\textbf{SOS:} \quad \delta(x)^2 = \sum_{i=1}^{n} (\Delta U_i)^2 \tag{17}$$

where n represents the total number of upstream turbines and ΔU_i denotes the velocity deficit caused by the ith wind turbine. While these two approaches are prevalent, other superposition models have also been developed and applied in the field. These include the energy balance method, the geometric sum method, and the maximum deficit method [16]. The development of various wake superposition methods stems from the challenge of combining multiple wake effects, given the inherently non-linear nature of the Navier-Stokes equations that govern fluid dynamics.

2.3 Neural Networks

Artificial Neural Network (ANN) are computational processing systems that draw inspiration from the functioning of biological nervous systems, particularly the human brain. These networks consist of numerous interconnected computational nodes, known as neurons, which operate collaboratively in a distributed manner. The collective goal of these neurons is to learn from the input data and optimize the network's final output.

The fundamental structure of an ANN typically includes an input layer, one or more hidden layers, and an output layer, as illustrated in Figure 2.5. This figure depicts a basic feedforward neural network with one hidden layer. The process begins with the input layer receiving data, often in the form of a multidimensional vector, which it then distributes to the hidden layers. Within these hidden layers, complex decision-making occurs as each layer processes information from the previous one. The neurons in these layers assess how stochastic changes within themselves affect the final output, either improving or detracting from it. This iterative process of adjustment and evaluation constitutes the learning mechanism of the network. When multiple hidden layers are stacked upon each other, the resulting architecture is commonly referred to as deep learning, allowing for more intricate pattern recognition and data processing capabilities.

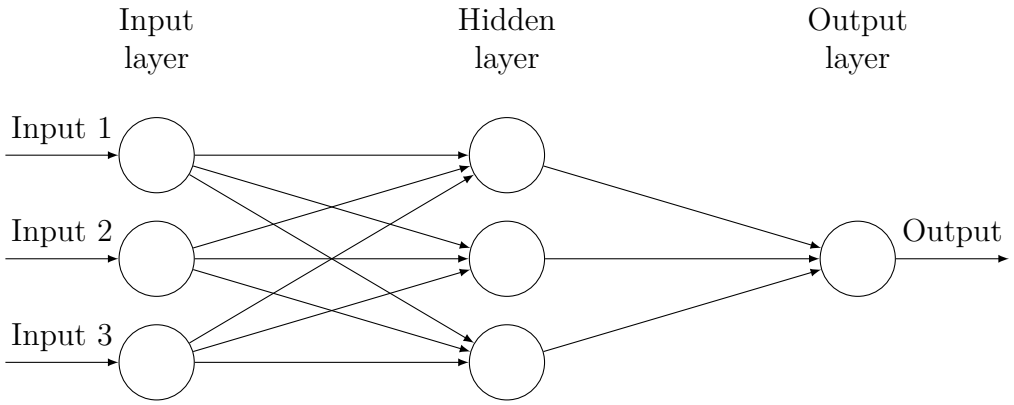


Figure 2.5: Basic structure of a feedforward neural network with one hidden layer.

To understand the functioning of individual neurons within the network, we can examine the perceptron model shown in Figure 2.6. This model represents the basic computational unit of a neural network. It receives multiple inputs (x_1, x_2, x_3), each associated with a weight (w_1, w_2, w_3). These weighted inputs are summed along with a bias term (b). The result is then passed through an activation function (f) to produce the neuron's output (y). This process mimics the way biological neurons process and transmit signals, forming the foundation for the network's ability to learn and make decisions.

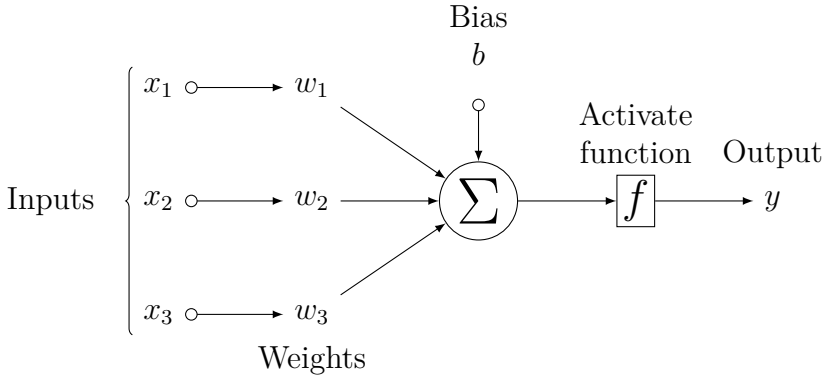


Figure 2.6: Diagram of a perceptron model showing inputs, weights, bias, summation, activation function, and output.

Mathematically, we can express the operation of a perceptron as follows:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{18}$$

Here, n is the number of inputs, w_i are the weights, x_i are the inputs, b is the bias, and f is the activation function. The choice of activation function is further discussed in subsubsection 2.3.1.

The learning process in neural networks involves adjusting the weights and bias to minimize a loss function. The subject of loss functions and how they can be minimized using a gradient descent method is further explored in subsubsection 3.5.1 and subsubsection 2.3.2
The two primary learning paradigms in image processing tasks are supervised and unsupervised learning. Supervised learning involves training with pre-labelled inputs that serve as targets. Each training example in this paradigm consists of a set of input values (vectors) and one or more associated designated output values. The main objective of supervised training is to minimize the model's overall classification error by correctly calculating the output value of each training example through the training process. In contrast, unsupervised learning operates without labelled training data. The success of unsupervised learning is typically evaluated by the network's ability to either reduce or increase an associated cost function. It is noteworthy that the majority of image-focused pattern-recognition tasks generally rely on classification using supervised learning techniques. This approach allows for more direct and precise training in tasks where specific output categories or values are known and desired.

In both learning paradigms, the network adjusts the weights and biases of its neurons (as shown in Figure 2.6) to optimize its performance on the given task. This adjustment process, known as backpropagation in many learning scenarios, allows the network to learn from its mistakes and improve its accuracy over time. The flexibility of neural network architectures, combined with their ability to learn complex patterns, makes them powerful tools for a wide range of applications, including image recognition, natural language processing, and decision-making systems.

2.3.1 Activation Functions

Activation functions, also known as non-linearities, are essential for the functionality of individual neurons within a neural network. Without these non-linear functions, the network would collapse into a linear model, effectively reducing its capabilities. In such a scenario, no matter how many layers the network has, it can be simplified to $y = W \cdot x$. This limitation means the

model can only solve problems where the data is linearly separable. Several activation functions are commonly used in neural networks, each with its characteristics and benefits:

Sigmoid: This was one of the first activation functions utilized in neural networks. The Sigmoid function maps input values to the range (0, 1). However, it is prone to the vanishing gradient problem, where input values significantly above or below zero lead to gradients that are close to zero. This sensitivity makes it challenging for the network to learn effectively.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{19}$$

tanh: The hyperbolic tangent function is another early activation function. It is similar to the Sigmoid function but outputs values in the range (-1, 1), allowing for negative values. Despite this advantage, it also suffers from the vanishing gradient problem at extreme input values.

$$f(x) = \tanh(x) = \frac{\sinh(x)}{\cosh(x)} \tag{20}$$

ReLU: The Rectified Linear Unit (ReLU) has become the standard activation function in modern deep learning, particularly in computer vision. ReLU outputs the input directly if it is positive; otherwise, it outputs zero. While effective, it can lead to dying ReLUs where neurons become inactive for all inputs.

$$f(x) = \max(0, x) \tag{21}$$

Leaky ReLU: To address the dying ReLU problem, Leaky ReLU was introduced. This function allows a small, non-zero gradient when the input is negative, thus keeping the neuron active and improving learning.

$$f(x) = \begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{otherwise} \end{cases} \tag{22}$$

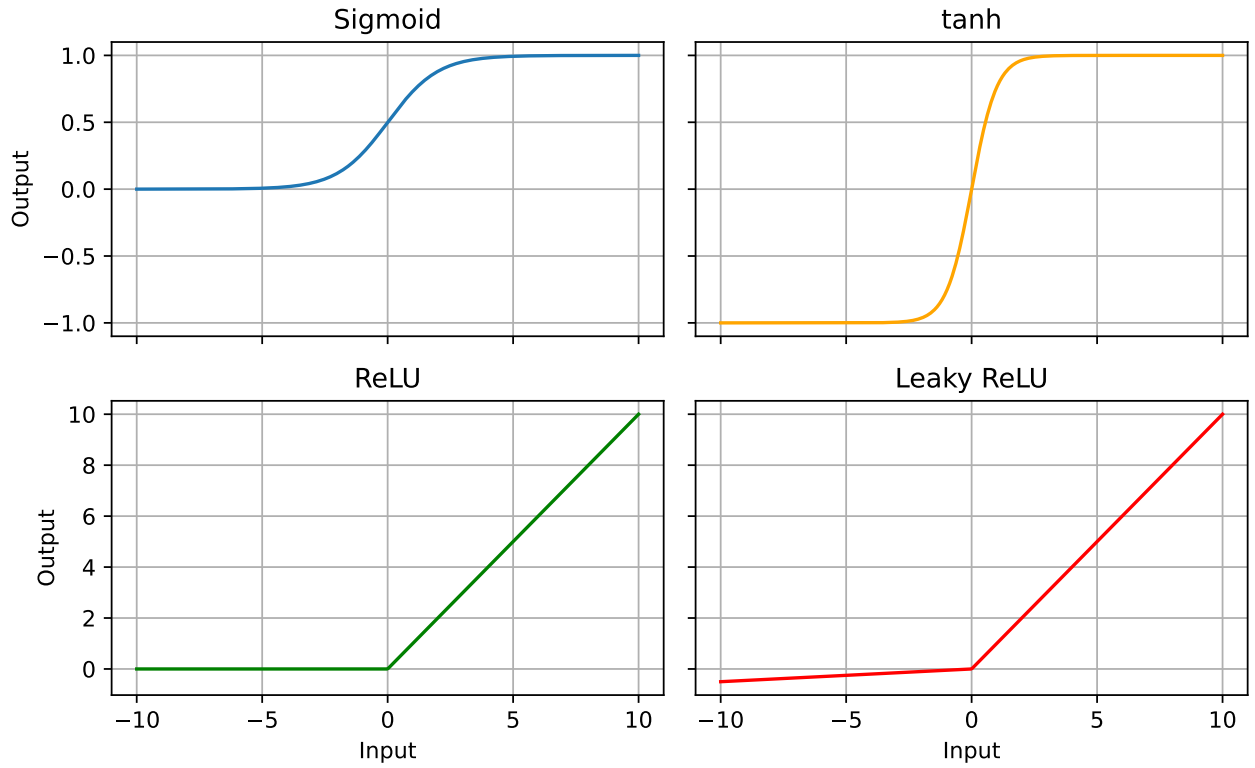


Figure 2.7: Comparison of the input-output relationships of various activation functions: Sigmoid, tanh, ReLU, and Leaky ReLU.

2.3.2 Adam Optimization Algorithm

The Adaptive Moment Estimation (Adam) optimization algorithm, introduced by Kingma and Ba [48] in 2014, is a popular method for stochastic optimization in deep learning. It combines ideas from two other optimization techniques: Root Mean Square Propagation (RMSProp) and Momentum with an added bias correction [65]. Adam is designed to be computationally efficient, have little memory requirements, and be well-suited for problems with large datasets and many parameters. Adam adapts the learning rates of each parameter individually by utilizing estimates of the first and second moments of the gradients. The key steps of the algorithm include: initializing parameters and hyperparameters, computing gradients, updating biased first and second moment estimates, computing bias-corrected moment estimates, and updating parameters.
Let θ_t be the parameters at timestep t, and g_t be the gradient of the objective function with respect to the parameters at timestep t. The Adam update rule is defined as [68]:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{23}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{24}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{25}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{26}$$

$$\theta_t = \theta_{t-1} - \frac{\gamma \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{27}$$

Here, m_t is the estimate of the first moment (mean) of the gradients, v_t is the estimate of the second moment (uncentered variance) of the gradients, $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected estimates

of the first and second moments, γ is the learning rate, β_1 and β_2 are exponential decay rates for moment estimates, and ϵ is a small constant for numerical stability.

Adam's key features include adaptive learning rates, which adjust for each parameter based on estimates of the gradients' first and second moments. This allows the algorithm to handle parameters of varying scales effectively. The algorithm employs bias correction to account for the zero initialization of moment estimates. The first moment estimate (mean) introduces momentum, accelerating convergence in relevant directions and dampening oscillations. The second moment estimate (uncentered variance) provides per-parameter scaling, similar to RMSProp, modulating step size to control overshooting. Adam typically uses the following default values for its hyperparameters [68]: $\alpha = 0.001$ (learning rate), $\beta_1 = 0.9$ (exponential decay rate for the first moment estimate), $\beta_2 = 0.999$ (exponential decay rate for the second moment estimate), and $\epsilon = 10^{-8}$ (small constant for numerical stability). These defaults generally work well for a wide range of problems, though they may need tuning for specific applications. Adam's adaptive learning rate approach is particularly effective for problems with sparse gradients or noisy data. It often requires little tuning and can handle non-stationary objectives well.

2.4 Convolutional Neural Networks

CNNs represent a significant leap forward in the field of artificial intelligence, embodying the longstanding goal of mimicking the human brain's cognitive processes. Rooted in the study of biological visual systems, CNNs have revolutionized machine learning, particularly in the domains of image recognition, object detection, and visual data analysis.
The CNN was first introduced by Fukushima [36] in 1980. However, it was LeCun et al. [52] who popularised CNNs with the development of LeNet-5, a seven-layer convolutional network that utilized backpropagation and adaptive weights. This architecture has since become the blueprint for modern CNN designs, with most current architectures being variations or expansions of LeNet-5's core principles [3].

At their core, CNNs aim to replicate the functioning of visual sensory organs in living beings. They achieve this through a structured sequence of operations: convolutional operations, activation functions, pooling and more. This process allows CNNs to recognize various types of objects, from simple digits to complex images or even specific actions within an image. The key innovation of CNNs lies in their use of convolutional layers, which apply learnable filters across the input space. This approach enables the network to capture local patterns and spatial relationships within the data, significantly reducing the number of parameters compared to fully connected networks while maintaining high performance.

2.4.1 Convolutional Layers

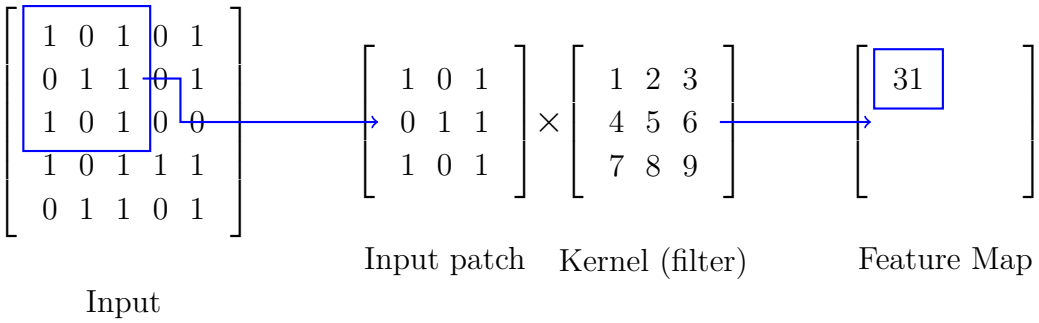


Figure 2.8: Illustration of a convolutional layer operation. The diagram shows a 5x5 input, with a 3x3 input patch, being convolved with a 3x3 kernel. The resulting output is a feature map generated by sliding the kernel over the input.

Convolutional layers form the core of CNNs, serving as the primary mechanism for feature extraction from input data. These layers are designed to capture local patterns and spatial hierarchies, making them particularly effective for processing grid-like data such as images. The key operation in a convolutional layer is the convolution itself. In this process, a small array of numbers called a kernel or filter slides across the input data, performing element-wise multiplication and summation at each position, as seen in Figure 2.8. This operation can be mathematically expressed as:

$$(f * g)(x, y) = \sum_i \sum_j f(i, j) * g(x - i, y - j) \tag{28}$$

where f represents the input, g is the kernel, and (x, y) are the coordinates in the output feature map. The kernel acts as a feature detector, identifying specific patterns or features in the input. As the kernel slides, it produces a feature map, highlighting areas where the specific feature is detected. Multiple kernels are typically used in each convolutional layer, each learning to detect different features.

Convolutional layers are characterized by local connectivity, parameter sharing, and spatial hierarchy. Local connectivity means each neuron in the layer connects to only a small region of the input, known as the receptive field. This reduces the number of parameters compared to fully connected layers. Parameter sharing involves using the same set of weights (kernel) across the entire input, further reducing the number of parameters. Spatial hierarchy allows the network to learn increasingly complex and abstract features as data progresses through multiple convolutional layers. The output size of a convolutional layer depends on three main hyperparameters: kernel size (the dimensions of the filter, e.g., 3x3, 5x5), stride (the step size as the kernel slides across the input), and padding (additional pixels added to the input borders to control the output size). The output dimensions can be calculated using the formula:

$$\text{Output size} = \left(\frac{\text{Input size} + 2 \times \text{Padding} - \text{Kernel size}}{\text{Stride}} \right) + 1 \tag{29}$$

Convolutional layers are typically followed by an activation function, such as ReLU, to introduce non-linearity into the network. This combination of convolution and activation allows CNNs to learn complex, non-linear mappings from input to output.

2.4.2 Max Pooling

Pooling layers are a important component in CNN, typically inserted between successive convolutional layers. The primary functions of pooling are to reduce the spatial dimensions of the feature maps, decrease the computational load, and introduce a degree of translational invariance. By doing so, pooling helps to control overfitting and improve the network's ability to generalize. The most common type of pooling operation is max pooling, although other variants such as average pooling and L2-norm pooling exist. In this section, we will focus primarily on max pooling due to its widespread use and effectiveness. Max pooling operates by dividing the input feature map into rectangular subregions and outputting the maximum value for each subregion, as seen in Figure 2.9.

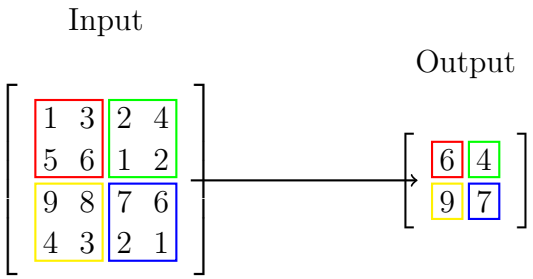


Figure 2.9: Max pooling in a CNN. The input matrix is divided into 2x2 regions, and the maximum value from each region is selected to form the output matrix.

The pooling window size and stride are key parameters in a pooling layer. A typical setup is a 2x2 window with a stride of 2, which halves the spatial dimensions of the feature map in both width and height. However, pooling can lead to loss of spatial information. Modern architectures, especially those needing detailed spatial information, may reduce or eliminate pooling layers, using strided convolutions or other methods instead.

2.4.3 Dropout Regularization

Dropout is a powerful regularization technique introduced by Srivastava et al. [83]. This method addresses the challenge of overfitting in Deep Neural Network (DNN), particularly when dealing with complex models and limited training data. The core concept of dropout is straightforward yet effective [83]: during training, randomly drop out (i.e., set to zero) a proportion of neurons in each layer. This process can be conceptualized as training an ensemble of many different neural networks, each with a subset of the full network's neurons. During training, dropout creates a different "thinned" network for each mini-batch, forcing the network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons. At inference time, the full network is used with all neurons, but the weights are scaled to ensure that the expected output of each neuron at test time is the same as its expected output during training time.

Dropout offers several key advantages, including reduced overfitting, an ensemble effect, and encouragement of more robust feature learning [39]. However, it also comes with considerations such as increased training time and the need for hyperparameter tuning of the dropout rate. By preventing complex co-adaptations on the training data, dropout helps neural networks to generalize better to unseen data, making it a valuable tool in the development of more robust and reliable deep learning models.

2.4.4 Convolutional Autoencoders

CAE are a special case of convolutional encoder-decoder models, where the model input and output are the same [57]. This architecture is particularly well-suited for processing image data and other types of data with spatial or temporal structure. A typical CAE consists of two main components: an encoder and a decoder. The encoder uses convolutional layers to compress the input into a lower-dimensional representation, often called the latent space or bottleneck. The decoder then uses transposed convolutional layers (also known as deconvolutional layers) to reconstruct the original input from this compressed representation. The key characteristics of CAEs include parameter sharing through convolutional operations, which reduces the number of parameters compared to fully connected autoencoders. They also preserve spatial relationships in the data, making them effective for image-related tasks.

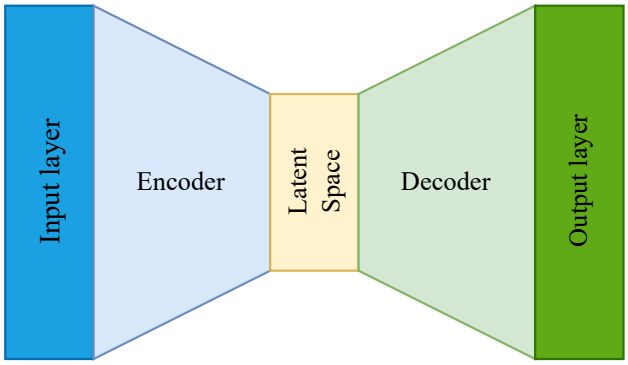


Figure 2.10: A CAE architecture consists of an encoder and a decoder. The encoder compresses the input data into a lower-dimensional latent space using convolutional layers, while the decoder reconstructs the original input from this compressed representation using transposed convolutional layers.

In the domain of wind farm modeling, CAEs demonstrate significant potential for processing and reconstructing complex spatial patterns of wind fields. By incorporating additional inflow conditions and turbine-specific data, these models can capture the intricate dynamics of wind farm environments. A notable example of this approach is the WakeNet model developed by Asmuth and Korb [7]. This convolutional encoder-decoder network showcases the capability to estimate the three-dimensional wake behind a single turbine with remarkable accuracy. The WakeNet model encodes turbine and inflow data to generate a full 3D representation of the wake, achieving a mean square root error of just 0.56%. This level of precision underscores the potential of CAE-based architectures in advancing our understanding and prediction of wind farm dynamics.

2.4.5 U-Net

U-Net is a Fully Convolutional Neural Network (FCN) architecture designed for image segmentation, introduced in 2015 by Ronneberger et al. [77]. Known for its accuracy, U-Net has become a standard in medical imaging applications [81]. The architecture features an encoding path, or contracting path, and a decoding path, or expanding path, forming a U-shape. This design allows the network to capture both local features and global context, resulting in precise segmentation.

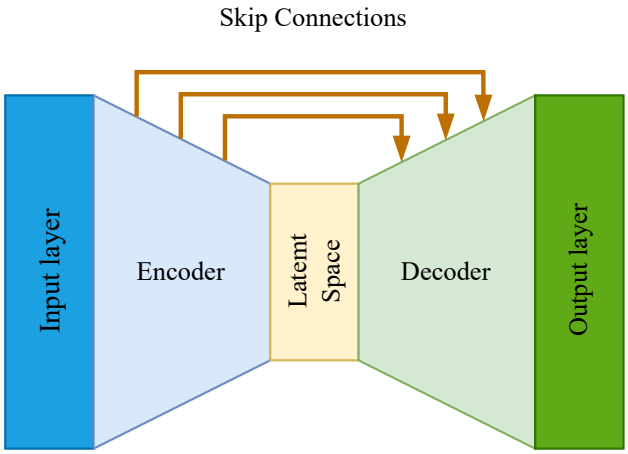
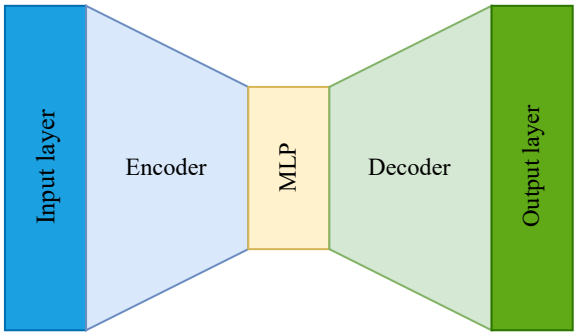


Figure 2.11: The U-Net architecture consists of an encoder and a decoder with skip connections in-between.

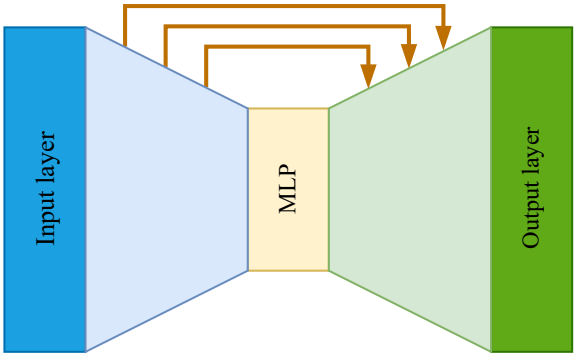
Similar to subsubsection 2.4.4, in the contracting path, convolutional layers are followed by max pooling operations, reducing the spatial dimensions of the input image while capturing high-resolution, low-level features. The expanding path uses transposed convolutions, also known as deconvolutions or upsampling layers, to increase the spatial resolution of the feature maps. This upsampling process helps the network reconstruct a dense segmentation map [73]. Skip connections in U-Net link corresponding layers from the encoding and decoding paths, allowing the network to combine local and global information. This integration of feature maps from earlier layers with those in the decoding path preserves essential spatial information and improves segmentation accuracy. Skip connections are implemented using concatenation, which merges feature maps from the encoding path with upsampled feature maps from the decoding path. This combination incorporates multi-scale information, leveraging both high-level context and low-level features, as seen in Figure 2.11.

2.4.6 Networks with Multilayer Perceptrons

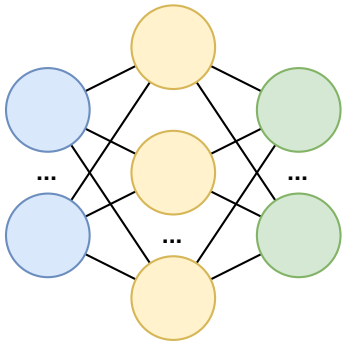
Incorporating a MLP into neural network architectures significantly improves their ability to model complex relationships in data. MLPs are feedforward neural networks characterized by fully connected layers and nonlinear activation functions such as ReLU or sigmoid (see subsubsection 2.3.1). This structure enables MLPs to learn intricate patterns beyond simple linear relationships, making them valuable in various applications including image recognition, natural language processing, and time series forecasting [24].



(a) CAE architecture with an MLP integrated in the latent space.



(b) U-Net architecture with an MLP integrated in the latent space.



(c) Schematic representation of a MLP.

Figure 2.12: Neural network architectures incorporating MLPs: (a) CAE with MLP in latent space, (b) U-Net with MLP in latent space, and (c) Basic structure of an MLP.

Researchers have successfully integrated MLPs into existing architectures like convolutional autoencoders (Figure 2.12a) and U-Net models (Figure 2.12b), as illustrated in Figure 2.12. This integration enhances the networks' performance by allowing MLPs to learn complex interactions among features extracted by the encoder and input conditions, which are typically combined in the latent space. In the field of wake effect modeling, MLPs have proven particularly effective at encoding relevant information in the latent space. For example, Xu and Duraisamy [91] used a CAE with a dense MLP in the latent space to predict transient flow around a cylinder with 99.95% accuracy and airwake behind a ship with 99.7% accuracy. In another study, Donglin et al. [26] employed a U-Net with an MLP in the latent space, combined with a Conditional Generative Adversarial Network (cGAN), to estimate flow fields around airfoils.
For these subsections (Wind Farm Wake Effects, Analytical Wake Models, Neural Networks and Machine Learning in Wind Energy) write an introduction the the section "Literature Review" containing these subsections

2.5 Machine Learning in Wind Energy

Machine learning has increasingly been recognized as a transformative technology in the field of wind energy, offering innovative solutions to enhance the efficiency and performance of wind farms. As wind energy continues to grow as a key component of renewable energy portfolios worldwide, the need for sophisticated modeling and prediction techniques becomes ever more critical. Machine learning algorithms, with their ability to process large datasets and uncover

complex patterns, are particularly well-suited to tackle the unique challenges presented by wind energy systems. This section explores the various applications of machine learning in wind energy, focusing on three primary areas: wind farm wake prediction, prediction of airfoil aerodynamic performance, and wind farm power predictions.

In the realm of wind farm wake prediction, machine learning models have been developed to understand and forecast the complex flow behaviors resulting from interactions between wind turbines. Traditional wake modeling methods, often relying on simplified physical models, can struggle to accurately capture these interactions, especially under varying operational conditions. Machine learning offers a promising alternative by leveraging diverse datasets and advanced neural network architectures to predict wake effects more accurately and efficiently.

2.5.1 Wind Farm Wake Prediction

The field of wind turbine wake modeling using machine learning is emerging, with various approaches explored to predict downstream flow behavior from individual turbines and wind farms, as shown in Table 2.1. While most studies focus on single turbine wakes, several extend their neural networks to model entire wind farm wakes. This is typically achieved through linear or sum of squares (SOS) superposition of individual wake calculations. However, Zhang and Zhao [93] and Zhang et al. [95] employ a different method, training their models on smaller turbine arrays. This approach enables the network to model wind turbine responses to non-uniform inflow from upstream turbines. Currently, this technique has only been demonstrated for wind farms arranged in perfect grid formations.

Data sources for these models vary widely, including RANS simulations, LES, Light Detection and Ranging (LiDAR) measurements, and Supervisory Control and Data Acquisition (SCADA) systems. The modeling techniques employed range from Random Forest (RF) and DNN to more complex architectures like cGAN and CNN.

Table 2.1: An overview of studies on data-driven modelling of wind-farm flow.

Author(s)	**Data**	**Method**	**Multiple Turbines**
Wilson et al. [87]	RANS	RF	No
Ti et al. [86]	RANS	DNN	SOS and linear superposition
Zhang and Zhao [93]	LES	cGAN	Small arrays
Renganathan et al. [75]	LiDAR	DNN	No
Nai-Zhi et al. [58]	SCADA	RF	SOS superposition
Zhang et al. [95]	LES	CNN	Small arrays
Nakhchi et al. [59]	LES	XGBoost	No
Asmuth and Korb [7]	LES	CNN	No
Anagnostopoulos et al. [5]	LES	CNN	Linear superposition

A related research area focuses on using data-driven models to simulate dynamic wake behavior using Proper Orthogonal Decomposition (POD) [41, 92, 4]. These models, trained on LES data, can predict temporal wake patterns for single or multiple turbines based on initial flow conditions. They're well-suited for LiDAR-based incoming flow measurement and subsequent prediction, enabling yaw and pitch adjustments to optimize power output and reduce turbine loads in changing inflow conditions. However, this approach isn't designed for static estimation of long-distance wake deficits given specific upstream wind speed and turbulence intensity values.

2.5.2 Prediction of Airfoil Aerodynamic Performance

An emerging research field in the wind energy and aviation industry involves using neural networks to predict the aerodynamic performance of airfoils. These studies generally fall into two main categories:

Prediction of Aerodynamic Coefficients: These models aim to predict the lift, drag, and moment coefficients of a given airfoil shape. This is often achieved through a series of convolutional layers or fully connected hidden layers, where the input geometry is reduced into a one-dimensional array containing the coefficients. For instance, Chen et al. [20] developed models for predicting multiple aerodynamic coefficients using neural networks. Zhang et al. [94] applied CNNs to predict aerodynamic coefficients, demonstrating improved prediction accuracy. Similarly, Cai et al. [18] created an efficient model for predicting aerodynamic coefficients with enhanced computational performance.

Prediction of Airfoil Flow Field: The second category of studies focuses on predicting the entire flow field around an arbitrary airfoil or shape. This is generally accomplished using CAE or CNN with a U-Net architecture, where a 2D image of an airfoil is encoded and then decoded into a velocity and/or pressure map of the flow around the airfoil. Chen et al. [21] and Leer and Kempf [53] have developed a fast flow field prediction model for arbitrary shapes using U-Net architecture. Portal-Porras et al. [72] utilized a U-Net architecture to control and predict flow fields around airfoils. Afshar et al. [1] used an encoder/decoder style network to predict aerodynamic flow fields, and Donglin et al. [26] implemented a conditional cGAN with a U-Net like architecture featuring skip connections and a MLP in the latent space for flow field prediction.

Predicting flow fields is particularly relevant to this report, as techniques used for predicting the flow around an airfoil can be adapted to model the flow around a wind park. By leveraging similar neural network architectures, such as U-Net CNNs, it is possible to encode and decode the complex flow interactions within a wind park. This approach can significantly aid in optimizing turbine placement and assessing performance. The cross-application of neural network methodologies highlights the versatility and potential of machine learning in advancing aerodynamic studies in both aviation and wind energy sectors.

2.5.3 Wind Farm Power Predictions

Graph Neural Network (GNN) have emerged as a promising approach for predicting wind farm power output. The layout-dependent interactions between wind turbines in a farm, where wake effects from upstream turbines impact downstream turbines, naturally lend themselves to being modeled as a graph structure. Several recent studies have explored using GNNs for wind farm power prediction tasks.

Bleeg [13] demonstrated the use of a GNNs as a surrogate model to predict turbine interaction losses in wind farms. Their approach modeled individual turbines as nodes in a graph, with edges representing wake interactions between turbines. By training on data from higher-fidelity simulations, the GNNs was able to learn to predict power losses due to wake effects for arbitrary farm layouts.Expanding on this concept, Duthé et al. [27] developed a more comprehensive GNN framework for predicting not just power, but also local flow conditions and turbine loads throughout a wind farm. Their Encode-Process-Decode GNNs architecture takes arbitrary farm layouts and inflow conditions as input, and outputs predictions of rotor-averaged

wind speed, turbulence intensity, power production, and damage equivalent loads for each individual turbine. By training on data generated from the PyWake wind farm simulator, their GNNs model was able to generalize to unseen farm layouts and inflow conditions.

A key advantage of the GNN approach highlighted by both studies is its ability to handle arbitrary farm layouts in a layout-agnostic manner. Once trained, the same GNNs model can be applied to predict power output for any farm configuration without needing to be retrained. This flexibility, combined with the fast inference time of GNNs, makes them well-suited as computationally efficient surrogates for more expensive physics-based simulations in applications like wind farm design optimization and real-time control. The results from these studies demonstrate the potential of GNNs to capture the complex wake interactions in wind farms and provide accurate power predictions. However, further work is still needed to improve prediction accuracy for challenging cases and to validate GNNs models against real-world wind farm data. Nonetheless, GNNs represent a promising direction for developing fast and flexible wind farm power prediction tools.

3 Methodology

The methodology section outlines the comprehensive approach used in this study to model and analyze long-distance wake effects in offshore wind farms. This section is divided into four main subsystems. Data Collection and Preprocessing details the novel wind park layout generation algorithm developed to create diverse and realistic offshore wind farm configurations. It also describes the rasterisation process used to transform continuous turbine position data into a format suitable for CNNs. The Analytical Wake Models subsection discusses the use of PyWake, an open-source wind farm simulation tool, and the specific wake models (Jensen, Bastankhah, and TurbOPark) employed in this study. This part also covers the simulation domain parameters and the wind turbine model used. The Neural Network Architectures section introduces the four main neural network architectures explored in the study: CAE, CAE with CAE/MLP, U-Net, and U-Net with MLP. Each architecture's structure and key components are detailed. Finally, the Training and Validation subsection covers critical aspects of the model training process, including loss function selection and weighting, data transformation techniques, and the hyperparameters used in training the neural networks. Together, these methodological components form a robust framework for investigating and predicting long-distance wake effects in offshore wind farms, balancing computational efficiency with spatial accuracy and model complexity.

3.1 General Workflow

Figure 3.1 shows the overall process used in this work. The pipeline starts with data generation, explained in subsection 3.2. Since real-world wake prediction data is scarce, PyWake is used to make synthetic data that mimics real wind turbine wakes using the engineering models defined in subsection 2.2. This step is important as it affects the ground accuracy of how the data-driven surrogate model performs. PyWake uses specific inflow parameters like the wind speed of wind farm layout to create wake fields, which serve as the ground truth for the experiments. The next part of the pipeline deals with data processing and data splitting.

The model training part is covered in subsection 3.4, exploring the architectures and hyperparameters of the most promising data-driven models. Each model is designed to capture complex relationships downwind from the wind farms using the generated wake data. The final part of the figure shows the evaluation, discussed in section 4.

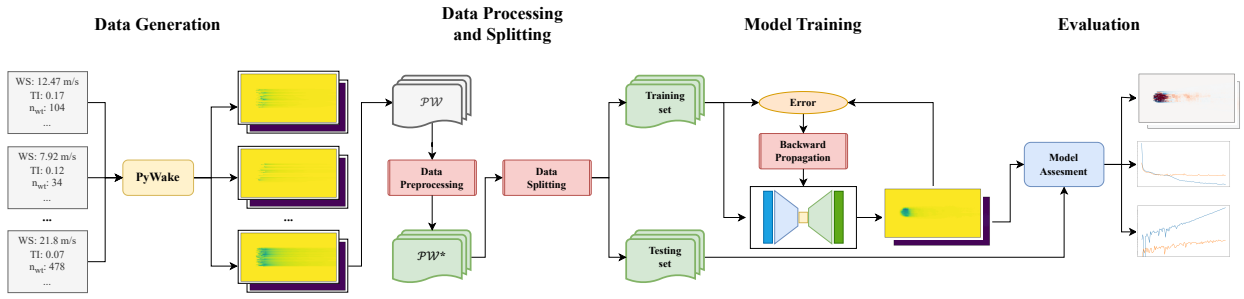


Figure 3.1: Schematic representation of the general pipeline used in this work, showing the flow from data generation to model evaluation.

3.2 Data Collection and Preprocessing

The accuracy and effectiveness of machine learning model predictions heavily depend on the quality and representativeness of the input data. This section outlines the comprehensive ap-

proach to data collection and preprocessing employed in this study, focusing on two critical aspects: wind park layout generation and data rasterisation. The wind park layout generation process aims to create diverse and realistic wind farm configurations that capture the complexity of modern offshore wind parks. By developing a novel algorithm that combines random polygon generation with strategic turbine placement, we ensure a wide range of scenarios for model training and evaluation.

Following layout generation, the rasterisation process transforms the continuous turbine position data into a format suitable for CNNs. This step is crucial in preserving spatial information while enabling efficient processing by the neural network architecture. Together, these preprocessing steps form the foundation of our data pipeline, enabling the subsequent machine learning models to learn from a rich and varied dataset that reflects real-world wind park layouts.

3.2.1 Wind Park Layout Generation

Existing open-source models for random wind farm layout generation include PLayGen by Harrison-Atlas et al. [42], which is capable of generating random turbine clusters and strings, and the model by Duthé et al. [27], which uses randomly distributed points in various regular polygons and ovals to generate its wind park layouts. While effective, these models don't fully capture current offshore wind park layout characteristics. A more comprehensive model is needed to simulate diverse scenarios. Current offshore wind park layouts often feature irregular polygon boundaries with varying edge numbers. Turbine positioning typically falls into two categories:

1. Regular grid layouts (e.g., Horns Rev 1, Figure 3.2)
2. Irregular layouts from optimization algorithms (e.g., Horns Rev 3, Figure 3.2)

Some layouts, like Horns Rev 2 or Anholt Offshore Wind Farm with regular, curved grids, aren't addressed by this algorithm. Neural networks trained solely on irregular polygon layouts might not accurately predict wake deficits for these parks.

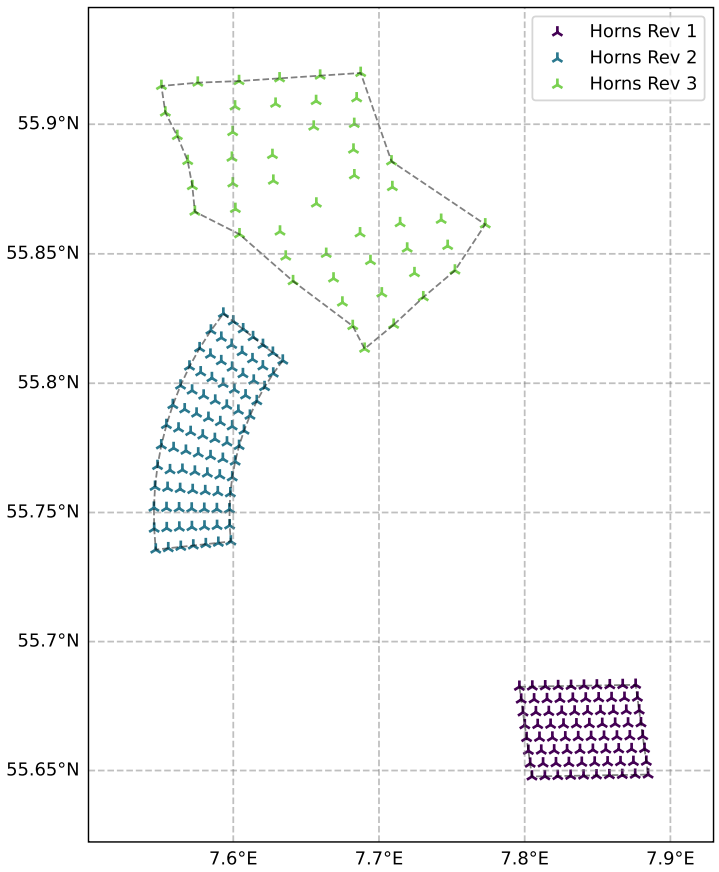


Figure 3.2: Horns Rev Offshore Wind Farm, Denmark, showing various layout strategies.

Future work should focus on incorporating algorithms for curved grid layouts and other real-world wind farm patterns. This would enhance the model's versatility, enabling accurate representation of a wider range of wind park configurations. As a result, wake deficit predictions would become more reliable and applicable across diverse offshore wind scenarios.

The wind park layout generation algorithm consists of three main steps. First, a random polygon is generated to outline the wind park boundaries. Next, individual wind turbine positions are created from a transformed regular grid within the polygon boundaries. Finally, PyWake[70] calculates the wind farm wake deficit using the given flow conditions. A detailed breakdown of the generation process follows:

1. Receive input parameters from Table 3.1.
2. Generate random angles for polygon edges using normal distribution, with irregularity controlling standard deviation. Normalize angles to sum to 360 degrees.
3. Create random radii using normal distribution, with average radius as mean and spikiness controlling standard deviation. Clip radii between 0.2 and 2 times average radius.
4. Calculate vertex coordinates using angles and radii. Create Polygon object.
5. Generate grid points within polygon bounds using specified spacing.
6. Apply boolean mask to keep only points inside polygon.
7. Rotate filtered points by random angle.

8. Add random noise to rotated points, scaled by spacing and noise parameter.

9. Adjust x-coordinates to move wind park to negative coordinates, accommodating weighted loss function in subsubsection 3.5.1 and ensuring consistent spatial reference for easier model training.

10. Use wind park layout to initiate PyWake simulation with set domain, resolution, and inflow conditions.

Table 3.1: Input variable ranges for wind park layout training data generation.

Parameter	**Description**	**Lower**	**Upper**
num_vertices	Polygon vertex count	3	10
irregularity	Variance in angle spacing (0-1)	0.1	0.4
spikiness	Variance in vertex distance from center (0-1)	0.1	0.4
avg_radius	Average center-to-vertex distance	10	150
spacing	Distance between initial turbine grid points	7	12
noise	Random displacement added to turbine positions	0	0.5
wind_speed	Wind speed at hub height	3	25
turbulence	Turbulence intensity at hub height	0	0.3
turbine_amount	Number of turbines in the wind farm	3	1000

Variables in Table 3.1 utilize uniform distributions, assigning equal probability to all values within their specified lower and upper bounds. This approach contrasts with models like those of Duthé et al. [27], which employ typical real-world distributions (e.g., Weibull for wind speed, log-normal for turbulence intensity). The uniform distribution ensures equal model confidence across the entire training domain, including boundary conditions such as high wind speeds or low turbulence intensity.

However, a training dataset reflecting actual inflow conditions might be preferable when access to training data is limited (e.g., CFD-generated) or if the model is expected to be used primarily under the most likely inflow conditions.

Exceptions to this uniform distribution approach are the *turbine_amount* and *avg_radius* parameters. The *turbine_amount* parameter serves as a cap for the wind farm size, ensuring that it is neither too small nor too large, which could lead to calculation failures due to limited system memory resources. The *avg_radius* parameter is defined as:

$$\mathcal{U}\left[r_{min}^3, r_{max}^3\right]^{\frac{1}{3}} \tag{30}$$

This formulation ensures that the wind park areas and wind turbine amounts in the training data follow a relatively uniform distribution. While the relationship between a circle's area and its radius is $A = \pi r^2$, the radius needs to be raised to a higher power to account for the jagged shape of the wind farm polygons. Empirically, it was found that raising the radius to the third power produces a relatively uniform area distribution.

To quantify wind farm shape, this study employs the isoperimetric quotient (*Polygon Roundness*). This metric provides a measure of how closely a wind farm's layout approximates a circular configuration, offering insights into the geometric characteristics that can impact wake effects and overall farm performance. The shape characterization factor S for a wind farm is calculated as:

$$S = \frac{A}{A_c} = \frac{4\pi A}{P^2} \tag{31}$$

Where A represents the wind farm's area, P denotes its perimeter, and A_c is the area of a circle with a perimeter equal to that of the wind farm. This calculation yields a dimensionless value ranging from 0 to 1, with 1 representing a perfect circle. Higher values indicate a more circular wind farm layout, while lower values suggest more irregular or elongated configurations.

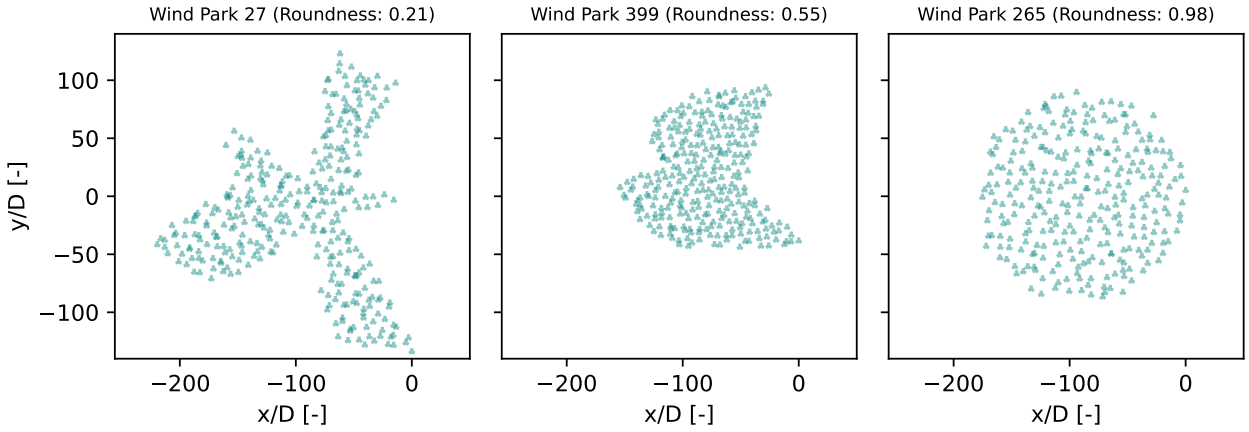


Figure 3.3: Examples of roundness for different wind park layouts.

A comprehensive overview of the parameter distributions used for training data generation can be found in Figure 3.4 and Figure B.1.

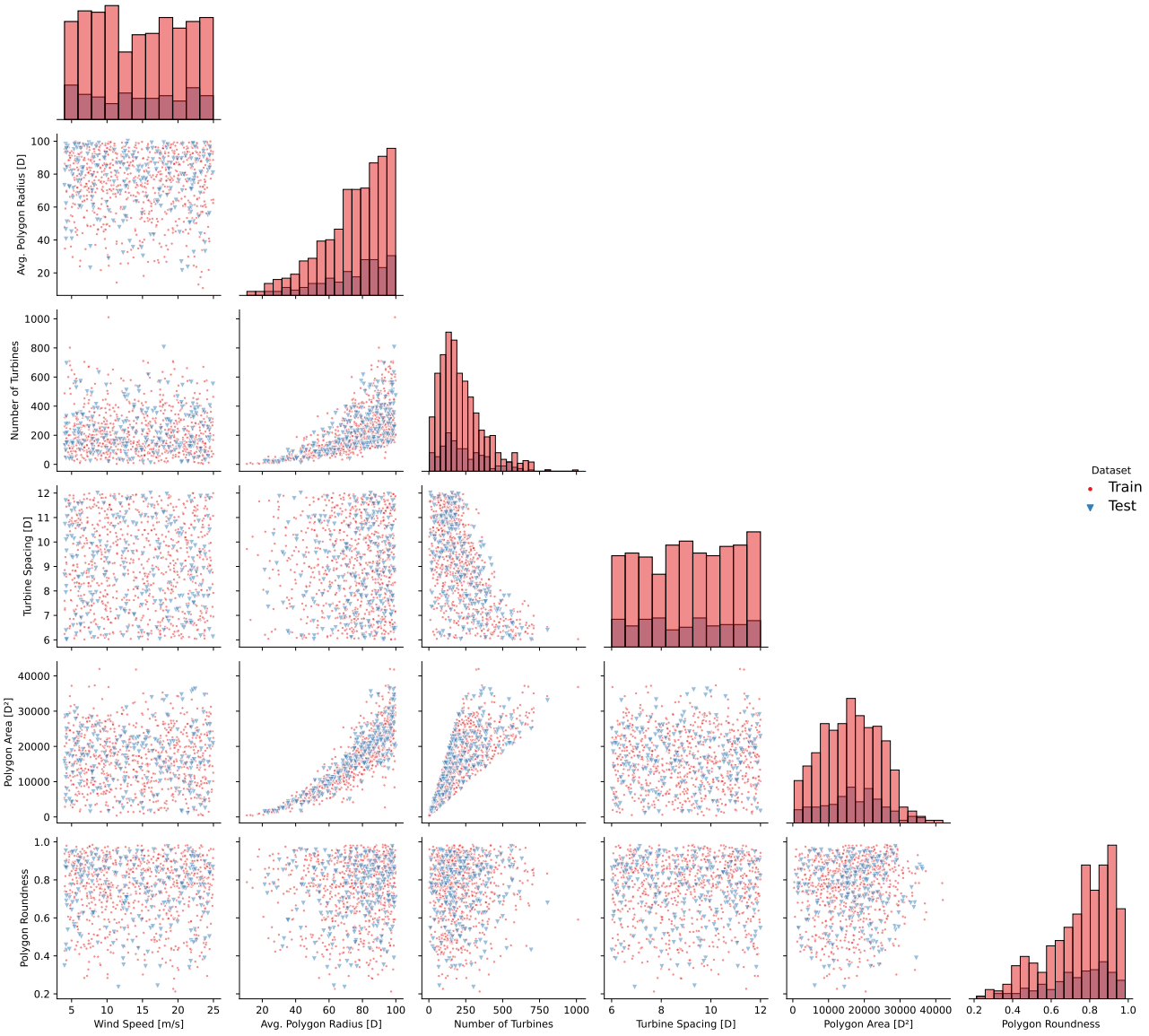


Figure 3.4: Distribution of input parameters for wind park generation algorithm.

An overview and example of the wind park layout algorithm process is illustrated in Figure 3.5. The figure demonstrates the sequential steps of the algorithm. First, a wind park border is defined using the random polygon algorithm. Next, a regular grid with spacing determined by the *turbine_spacing* parameter is overlaid on the area. All turbines falling within the polygon boundary are then identified. Finally, these turbines are rotated around the origin, given a random offset, and shifted into the negative x-coordinate domain. This process ensures a realistic yet randomized layout for each generated wind park.

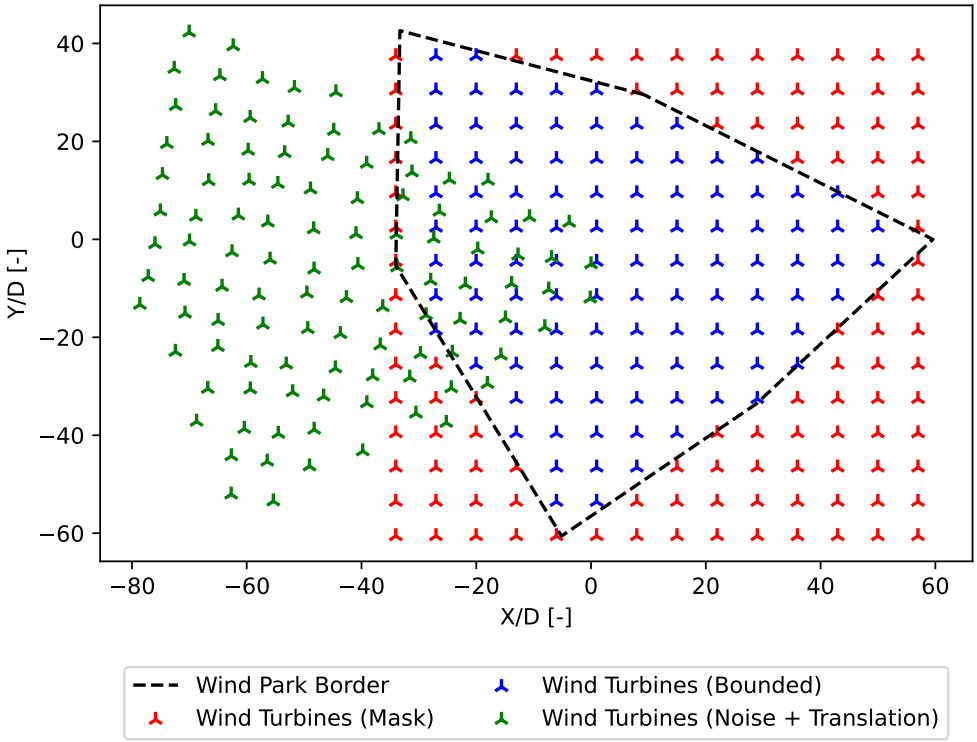


Figure 3.5: Example of wind park layout generation algorithm.

3.2.2 Wind Park Rasterisation

To prepare wind turbine positional data for input into CNNs, a rasterisation process is necessary. This process converts the continuous float values representing turbine positions into a discrete 2D tensor within a predefined domain. The evolution of this rasterisation method reflects a balance between computational efficiency and spatial accuracy. Initially, a boolean rasterisation approach was employed. In this method, each element in the input tensor to the CNN is assigned a binary value: 1 if a wind turbine's position falls within the bounds of the element's grid domain, and 0 otherwise. While computationally efficient, this method suffers from loss of precise positional information, with a potential positional error of:

$$\text{error}_{\text{max}} = \sqrt{x_{\text{res}}^2 + y_{\text{res}}^2} \tag{32}$$

Where x_{res} and y_{res} represent the resolution of the grid in the x and y directions, respectively. To address this limitation, the boolean approach was replaced with a Supersampling Anti-Aliasing (SSAA) rasterisation method, inspired by the work of Zhang et al. [94]. In this approach, the value of each element in the tensor reflects the proximity of a wind turbine to the center of the element. This method, illustrated in Figure 3.6, allows for the reconstruction of exact wind turbine positions from the grid tensor, enabling the neural network to differentiate between very similar wind parks.

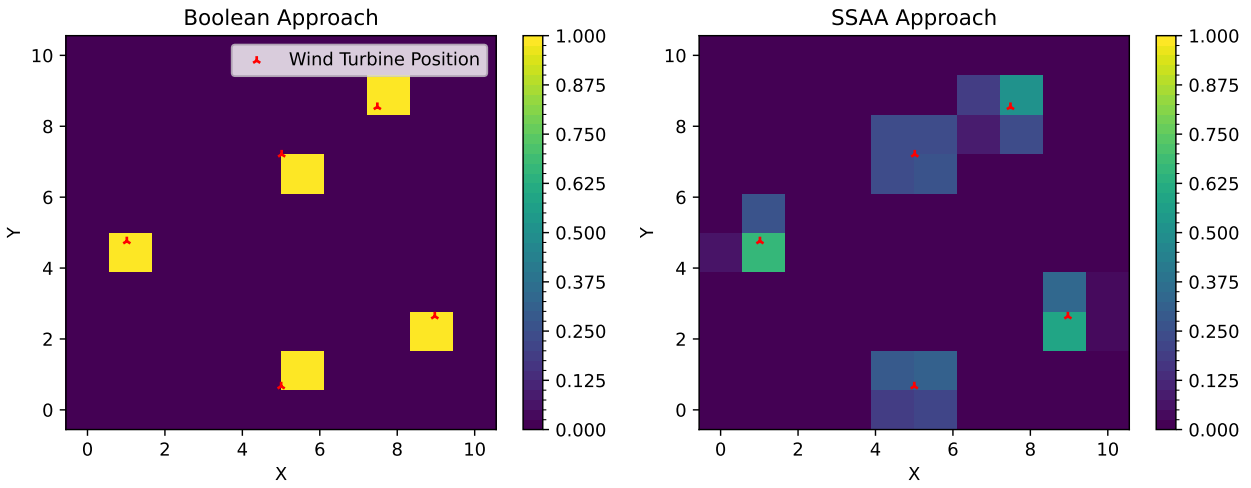


Figure 3.6: Comparison between boolean rasterisation (left) and SSAA rasterisation (right), showing improved spatial representation with SSAA.

The SSAA method was chosen over other alternatives due to its balance between accuracy and computational feasibility. While more complex methods exist, SSAA provides sufficient spatial precision without excessive computational overhead, making it suitable for the large-scale data processing required in this study. Both approaches are susceptible to errors when multiple turbines are in close proximity. To ensure no overlapping, turbines must be spaced at least one spatial resolution apart for the boolean method, and two grid spatial resolutions apart for the SSAA method. The algorithm for generating the SSAA grid is more complex and computationally intensive than the boolean approach. It involves a binary search for the wind turbine index position and requires looping over all wind turbines individually, as detailed in Algorithm 1. This enhanced method allows for more accurate spatial representation and differentiation of wind parks, albeit at the cost of increased computational complexity and resolution requirements.

Algorithm 1 Position to Grid using SSAA

Input: x_{range}, y_{range}, wt_{pos}
Output: $grid$

$x_{num} \leftarrow$ length of x_{range}
$y_{num} \leftarrow$ length of y_{range}
$grid \leftarrow \text{zeros}(x_{num}, y_{num})$
for each (x, y) in wt_{pos} **do**
    $x_{idx} \leftarrow \text{binarysearch}(x_{range}, x)\ -1$
    $y_{idx} \leftarrow \text{binarysearch}(y_{range}, y)\ -1$
    $x_{dist} \leftarrow \frac{x - x_{range}[x_{idx}]}{x_{range}[1] - x_{range}[0]}$
    $y_{dist} \leftarrow \frac{y - y_{range}[y_{idx}]}{y_{range}[1] - y_{range}[0]}$
    $weights \leftarrow \begin{bmatrix}(1 - x_{dist}) \cdot (1 - y_{dist}) & x_{dist} \cdot (1 - y_{dist}) \\ (1 - x_{dist}) \cdot y_{dist} & x_{dist} \cdot y_{dist}\end{bmatrix}$
    $grid[y_{idx} : y_{idx} + 2, x_{idx} : x_{idx} + 2]\ +\!= weights$
end for

This revised approach to rasterisation ensures a more accurate representation of wind turbine positions, which is important for the subsequent neural network analysis of wind farm layouts and their effects on wake patterns.

3.3 Analytical Wake Models

PyWake, an open-source Python-based wind farm simulation tool developed at the Technical University of Denmark (DTU) [70], is employed in this study. It efficiently computes flow fields, individual turbine power production, and Annual Energy Production (AEP) for entire wind farms. PyWake interfaces with various engineering models and CFD RANS (PyWakeEllipSys), enabling accurate calculations of wake propagation and quantification of turbine interactions within wind farms. The primary function of PyWake is to calculate wake interactions in wind farms for a range of steady-state conditions efficiently. This capability is crucial for assessing power production while accounting for wake losses in specific wind farm layouts. PyWake supports multiple engineering wake models, including Jensen, Bastankhah, and TurbOPark, which are utilized in this study.

3.3.1 PyWake Model Settings

This study employs three different wake models: Jensen, Bastankhah, and TurbOPark. Each model utilizes specific settings within the PyWake framework to simulate wake effects. To maximise the variety in the different wind park layout used throughout the studies, all the simulations are done at a wind direction of 270°, instead of running multiple simulations on the same wind park layout for different wind directions. This means, it the trained neural networks are to be used to estimate the wake deficit of wind parks, the wind park layouts needs to be rotated around the wind park centroid (mean turbine position). This should achieve the same effect are varying the wind direction.

All three models use the PyWake *PropagateDownwind* method for wake propagation. The *PropagateDownwind* model calculates wind farm effects by iterating over turbines in downstream order. For each turbine, it computes the effective wind speed by subtracting upstream deficits from the free stream speed. This effective speed then determines the turbine's impact on downstream locations. This method, while fast, excludes upstream blockage effects. The specific settings for each model are detailed in Table 3.2.These models provide different approaches to calculating wake deficits, allowing for a comprehensive analysis of wake effects under various conditions. By employing multiple models, this study aims to capture a range of potential wake behaviors, enhancing the robustness of the neural network training data.

Table 3.2: Overview of PyWake Engineering Wind Farm Models used for simulations.

Setting	**Jensen**	**Bastankhah**	**TurbOPark**
Wake Deficit	NOJensen	BastankhahGaussianDeficit	TurboGaussianDeficit
Superposition	SquareSum	SquareSum	SquareSum
Blockage	None	None	None
Turbulence	CrespoHernandez	STF2005TurbulenceModel	STF2017TurbulenceModel
Propagation Method	PropagateDownwind	PropagateDownwind	PropagateDownwind

The simulations were conducted with a grid resolution set as high as computationally reasonable to capture detailed wake behavior. The computational domain was discretized into 512 elements along the wind direction (x-axis) and 256 elements perpendicular to the wind direction (y-axis). The spatial extent of the simulations was defined in terms of rotor diameters (D) for comparability across different turbine sizes. The domain ranges from $-256D$ to $768D$ in the wind direction and from $-256D$ to $256D$ perpendicular to the wind direction. This large domain allows for the observation of wake effects far downwind, which is the main interest of

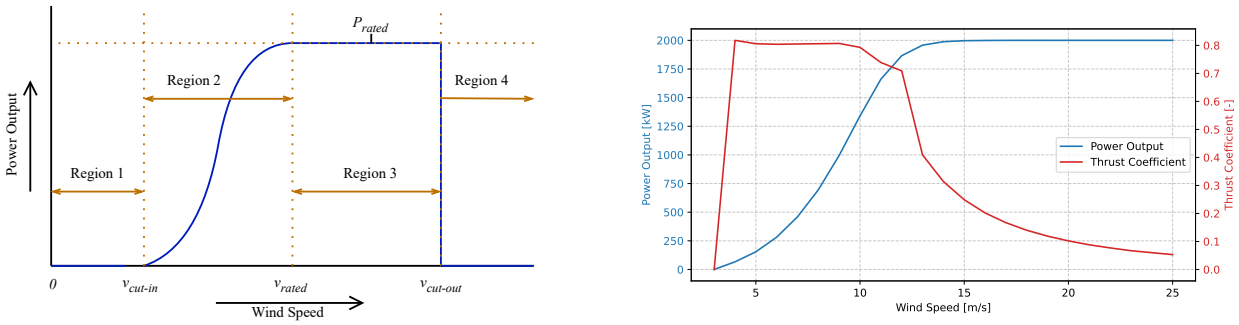
this thesis. Given the number of elements and the range of the domain, each grid cell represents an area of $2D \times 2D$. This resolution ensures that two turbines are almost never placed in the same element. As seen from Table 3.1, the minimum turbine spacing used is 7 diameters in the grid. Thus, even when considering the random position noise and rasterization method discussed in subsubsection 3.2.2, turbine overlap should be minimal. The key parameters of the simulation domain are summarized in Table 3.3.

Table 3.3: Simulation Domain Characteristics

Domain Attribute	**Streamwise (x)**	**Spanwise (y)**
Grid Resolution	512 cells	256 cells
Domain Extent	1024D	512D
Lower Boundary	-256D	-256D
Upper Boundary	768D	256D
Cell Size	2D × 2D	2D × 2D

3.3.2 Wind Turbine Model

The power curve of a typical pitch-regulated wind turbine can be divided into distinct regions, as illustrated in Figure 3.7a. Region 1 represents wind speeds below the cut-in speed, where the turbine does not produce power. Region 2 is characterized by a steep, approximately cubic increase in power output as wind speed rises. In this region, small changes in wind speed lead to significant changes in power output, making accurate wake modeling crucial for power prediction. Region 3 begins at the rated wind speed, where the turbine reaches its maximum power output. In this region, the power output remains constant despite increasing wind speed, typically maintained through pitch control. Finally, at the cut-out speed, the turbine stops operating for safety reasons. For this study, the Vestas V80 model has been used in all simulations, with its specific power and thrust curves shown in Figure 3.7b.



(a) Typical power curve of a pitch-regulated wind turbine

(b) Vestas V80 2MW Wind Turbine Power Curve and Thrust Coefficient

Figure 3.7: Power curves illustrating (a) general regions of wind turbine operation and (b) specific characteristics of the Vestas V80 model used in this study.

3.4 Neural Network Architectures

This section presents the various neural network architectures employed in our study. We explore four main architectures: Convolutional Autoencoder (CAE), Convolutional Autoencoder with Multi-Layer Perceptron (CAE-MLP), U-Net, and U-Net with Multi-Layer Perceptron (U-Net-MLP). These architectures represent different approaches to processing and learning from

spatial data, each with its own strengths and characteristics. The following subsections detail the structure and key components of each architecture.
In all architectures, ReLU activation functions are utilized following each encoding and decoding step, except for the final decoding layer. This activation function introduces non-linearity into the models, enabling them to capture more complex patterns and representations, as explained in subsubsection 2.3.1. Additionally, a dropout layer is implemented after the Fully Connected (FC) layer in the latent space of each model, with a 30% probability of zeroing elements. This technique helps prevent overfitting, as detailed in subsubsection 2.4.3.

3.4.1 Convolutional Autoencoder

The Convolutional Autoencoder (CAE) serves as our baseline architecture. It consists of an encoder and a decoder, both utilizing convolutional layers. The encoder compresses the input data into a lower-dimensional latent space, while the decoder reconstructs the original input from this compressed representation.

Table 3.4: Convolutional Autoencoder Architecture

Layer Group	**Layers**	**In Channels**	**Out Channels**	**Kernel**	**Stride**	**Padding**	**Out Padding**
Encoder Block 1	Encode 1-3, MaxPool	$in_{channels}$	8	5×5, 2×2	1,2,1,2	2,2,2,–	–
Encoder Block 2	Encode 4-5, MaxPool	8	16	5×5, 2×2	2,1,2	2,2,–	–
Encoder Block 3	Encode 6-7, MaxPool	16	32	5×5, 2×2	2,1,2	2,2,–	–
Encoder Block 4	Encode 8-9, MaxPool	32	64	3×3, 2×2	1,1,2	1,1,–	–
Encoder Block 5	Encode 10-11, MaxPool	64	128	3×3, 2×2	1,1,2	1,1,–	–
FC Encode	–	256 + [WS, TI]	256	–	–	–	–
Decoder Block 1	Decode 1-2, Unpool	128	64	3×3, 2×2	1,1,2	1,1,–	–
Decoder Block 2	Decode 3-4, Unpool	64	32	3×3, 2×2	1,1,2	1,1,–	–
Decoder Block 3	Decode 5-6, Unpool	32	16	5×5, 2×2	1,2,2	2,2,–	–,1,–
Decoder Block 4	Decode 7-8, Unpool	16	8	5×5, 2×2	1,2,2	2,2,–	–,1,–
Decoder Block 5	Decode 9-10, Unpool	8	4	5×5, 2×2	1,2,2	2,2,–	–,1,–
Final Decode	Decode 11	4	$out_{channels}$	5×5	1	2	–

As shown in Table 3.4, the CAE architecture comprises five encoder blocks and five decoder blocks, with a fully connected layer in between. Each encoder block consists of multiple convolutional layers followed by max pooling, progressively reducing the spatial dimensions while increasing the number of channels. The decoder mirrors this structure, using unpooling operations to upsample the feature maps. The encoder starts with $in_{channels}$ and gradually increases to 128 channels, while the decoder reverses this process, ending with $out_{channels}$. The fully connected layer in the middle (FC Encode) takes the flattened output of the last encoder block and additional inputs (WS and TI, representing wind speed and turbulence intensity, respectively) to produce a 256-dimensional latent vector.

3.4.2 Convolutional Autoencoder with MLP

The CAE with Multi-Layer Perceptron (CAE-MLP) builds upon the basic CAE architecture by introducing a more complex fully connected layer structure in the latent space.

Table 3.5: Convolutional Autoencoder Architecture

Layer Group	Layers	In Channels	Out Channels	Kernel	Stride	Padding	Out Padding
Encoder Block 1	Encode 1-3, MaxPool	$in_{channels}$	8	5×5, 2×2	1,2,1,2	2,2,2,–	–
Encoder Block 2	Encode 4-5, MaxPool	8	16	5×5, 2×2	2,1,2	2,2,–	–
Encoder Block 3	Encode 6-7, MaxPool	16	32	5×5, 2×2	2,1,2	2,2,–	–
Encoder Block 4	Encode 8-9, MaxPool	32	64	3×3, 2×2	1,1,2	1,1,–	–
Encoder Block 5	Encode 10-11, MaxPool	64	128	3×3, 2×2	1,1,2	1,1,–	–
FC Encode	Linear 1	256 + [WS, TI]	1024	–	–	–	–
	Linear 2	1024	1024	–	–	–	–
	Linear 3	1024	256	–	–	–	–
Decoder Block 1	Decode 1-2, Unpool	128	64	3×3, 2×2	1,1,2	1,1,–	–
Decoder Block 2	Decode 3-4, Unpool	64	32	3×3, 2×2	1,1,2	1,1,–	–
Decoder Block 3	Decode 5-6, Unpool	32	16	5×5, 2×2	1,2,2	2,2,–	–,1,–
Decoder Block 4	Decode 7-8, Unpool	16	8	5×5, 2×2	1,2,2	2,2,–	–,1,–
Decoder Block 5	Decode 9-10, Unpool	8	4	5×5, 2×2	1,2,2	2,2,–	–,1,–
Final Decode	Decode 11	4	$out_{channels}$	5×5	1	2	–

As illustrated in Table 3.5, the CAE-MLP maintains the same convolutional structure as the basic CAE in both the encoder and decoder. However, the fully connected layer in the middle is replaced by a three-layer MLP. This MLP expands the latent representation to 1024 dimensions before compressing it back to 256 dimensions. This additional complexity in the latent space allows for more nuanced feature extraction and potentially improved reconstruction capabilities. As with the CAE, the MLP also incorporates wind speed (WS) and turbulence intensity (TI) information in its input.

3.4.3 U-Net

The U-Net architecture introduces skip connections between the encoder and decoder, allowing for direct information flow from earlier layers to later layers.

Table 3.6: U-Net Architecture with Skip Connections

Layer Group	Layers	In Channels	Out Channels	Kernel	Stride	Padding	Out Padding	Skip Connection
Encoder Block 1	Encode1-3, MaxPool	$in_{channels}$	8	5×5, 2×2	1,2,1,2	2,2,2,–	–	To Decode 10
Encoder Block 2	Encode 4-5, MaxPool	8	16	5×5, 2×2	2,1,2	2,2,–	–	To Decode 8
Encoder Block 3	Encode 6-7, MaxPool	16	32	5×5, 2×2	2,1,2	2,2,–	–	To Decode 6
Encoder Block 4	Encode 8-9, MaxPool	32	64	3×3, 2×2	1,1,2	1,1,–	–	To Decode 4
Encoder Block 5	Encode 10-11, MaxPool	64	128	3×3, 2×2	1,1,2	1,1,–	–	To Decode 2
FC Encode	–	256 + [WS, TI]	256	–	–	–	–	–
Decoder Block 1	Decode 1-2, Unpool	128, 64+128	64	3×3, 2×2	1,1,2	1,1,–	–	From Encode 11
Decoder Block 2	Decode 3-4, Unpool	64, 32+64	32	3×3, 2×2	1,1,2	1,1,–	–	From Encode 9
Decoder Block 3	Decode 5-6, Unpool	32, 16+32	16	5×5, 2×2	1,2,2	2,2,–	–,1,–	From Encode 7
Decoder Block 4	Decode 7-8, Unpool	16, 8+16	8	5×5, 2×2	1,2,2	2,2,–	–,1,–	From Encode 5
Decoder Block 5	Decode 9-10, Unpool	8, 4+8	4	5×5, 2×2	1,2,2	2,2,–	–,1,–	From Encode 3
Final Decode	Decode 11	4	$out_{channels}$	5×5	1	2	–	–

Table 3.6 details the U-Net architecture. The encoder and decoder structures remain similar to the CAE, but with the addition of skip connections. These connections concatenate the output of each encoder block with the input of the corresponding decoder block, effectively doubling the number of channels in the decoder inputs. This design helps preserve fine-grained spatial information that might otherwise be lost during the encoding process, potentially leading to more accurate reconstructions. The fully connected layer in the middle also incorporates wind speed and turbulence intensity data.

3.4.4 U-Net with MLP

The U-Net with Multi-Layer Perceptron (U-Net-MLP) combines the skip connection concept of U-Net with the enhanced latent space representation of the CAE-MLP.

Table 3.7: UNet/MLP Architecture with Skip Connections

Layer Group	Layers	In Channels	Out Channels	Kernel	Stride	Padding	Out Padding	Skip Connection
Encoder Block 1	Encode1-3, MaxPool	$in_{channels}$	8	5×5, 2×2	1,2,1,2	2,2,2,–	–	To Decode 10
Encoder Block 2	Encode 4-5, MaxPool	8	16	5×5, 2×2	2,1,2	2,2,–	–	To Decode 8
Encoder Block 3	Encode 6-7, MaxPool	16	32	5×5, 2×2	2,1,2	2,2,–	–	To Decode 6
Encoder Block 4	Encode 8-9, MaxPool	32	64	3×3, 2×2	1,1,2	1,1,–	–	To Decode 4
Encoder Block 5	Encode 10-11, MaxPool	64	128	3×3, 2×2	1,1,2	1,1,–	–	To Decode 2
FC Encode	Linear 1	256 + [WS, TI]	1024	–	–	–	–	–
	Linear 2	1024	1024	–	–	–	–	–
	Linear 3	1024	256	–	–	–	–	–
Decoder Block 1	Decode 1-2, Unpool	128, 64+128	64	3×3, 2×2	1,1,2	1,1,–	–	From Encode 11
Decoder Block 2	Decode 3-4, Unpool	64, 32+64	32	3×3, 2×2	1,1,2	1,1,–	–	From Encode 9
Decoder Block 3	Decode 5-6, Unpool	32, 16+32	16	5×5, 2×2	1,2,2	2,2,–	–,1,–	From Encode 7
Decoder Block 4	Decode 7-8, Unpool	16, 8+16	8	5×5, 2×2	1,2,2	2,2,–	–,1,–	From Encode 5
Decoder Block 5	Decode 9-10, Unpool	8, 4+8	4	5×5, 2×2	1,2,2	2,2,–	–,1,–	From Encode 3
Final Decode	Decode 11	4	$out_{channels}$	5×5	1	2	–	–

As shown in Table 3.7, this architecture maintains the skip connections of the U-Net while incorporating the three-layer MLP in the latent space, similar to the CAE-MLP. This hybrid approach aims to leverage the benefits of both the U-Net's ability to preserve spatial information and the MLP's capacity for complex feature extraction in the latent space. The MLP in this architecture also takes into account the wind speed and turbulence intensity data.

3.5 Training and Validation

This subsection explores the most important aspects of training and validating the neural network models used in this study for long-distance wake modeling in wind farms. It covers three key areas fundamental to the model's performance and reliability. The first area addressed is loss function selection and weighting. This includes an exploration of the choice between Mean Absolute Error (MAE) and MAE as loss functions, explaining the rationale for selecting Mean Squared Error (MSE) in this study. Additionally, a novel weighted loss function designed to emphasize long-distance wake effects is introduced. The second area focuses on data transformation. This section describes the data preprocessing techniques applied to wind speed and turbulence intensity data. The rationale behind these transformations and their impact on the model's ability to capture subtle wake effects at greater distances from the wind farm are explained. The final area covers the hyperparameters used in training the neural network models. This includes details on learning rate, batch size, number of epochs, and other critical settings that influence the training process.

3.5.1 Loss Function Selection and Weighting

Loss functions are a crucial component in training neural networks, as they measure how well the model's predictions match the actual data. The goal of training is to minimize this loss, thereby improving the accuracy of the model. In regression tasks, where continuous values are predicted, two commonly used loss functions are MAE and MSE. MAE calculates the average of the absolute differences between the predicted values and the actual values. It is defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{33}$$

where n is the number of observations, y_i represents the actual value, and $\hat{y}_i$ denotes the predicted value. MAE provides a straightforward measure of average error magnitude, treating all errors equally regardless of their size.

MSE, on the other hand, calculates the average of the squares of the differences between the predicted values and the actual values. It is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{34}$$

MSE places a higher penalty on larger errors due to the squaring of differences, which can be beneficial for certain applications where larger errors are particularly undesirable.

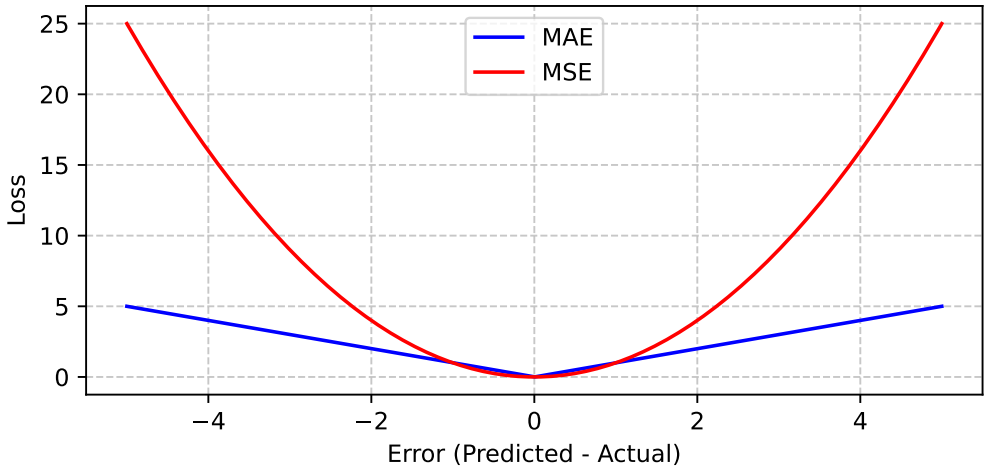


Figure 3.8: Comparison between MAE loss (blue) and MSE loss (red).

For this thesis, MSE has been chosen over MAE for several reasons:

- **Error Penalization:** MSE penalizes larger errors more heavily by squaring the differences. In wind farm flow modeling, large deviations can significantly impact the accuracy of flow predictions. MSE ensures that these large errors are minimized more effectively than MAE.
- **Gradient Properties:** MSE provides a smooth gradient, which is beneficial for optimization algorithms using gradient descent. This smooth gradient helps in efficiently converging to the minimum loss during the training process. In contrast, MAE has a constant gradient, which can lead to slower convergence. While MAE is technically non-differentiable at zero, this is not an issue in practice due to the extremely low probability of exact zero errors and the use of numerical optimization techniques.
- **Suitability for Data Characteristics:** The training data in this study is synthetically generated with well-behaving engineering models and is thus not expected to contain significant outliers. If the model were trained on SCADA data, which can be affected by sensor errors, or trained by results from mesh-based CFD, where singularities can produce non-physical results, a loss function less affected by outliers might be considered.
- **Computational Efficiency:** While MSE is generally slightly more computationally expensive due to the squaring operations, it is expected to converge faster due to its variable gradient (proportional to the error), whereas MAE has a constant gradient.

To further refine the model's focus on long-distance wake predictions, a weighted loss function was implemented. This function assigns different weights to losses within and beyond the wind farm:

$$\text{Weighted MSE} = \frac{1}{n}\sum_{i=1}^{n} w_i(y_i - \hat{y}_i)^2 \tag{35}$$

where w_i is the weight assigned to each prediction, defined as:

$$w_i = \begin{cases} 0.1, & \text{if } x_i \text{ is within the wind farm} \\ 0.9, & \text{if } x_i \text{ is downstream of the wind farm} \end{cases} \tag{36}$$

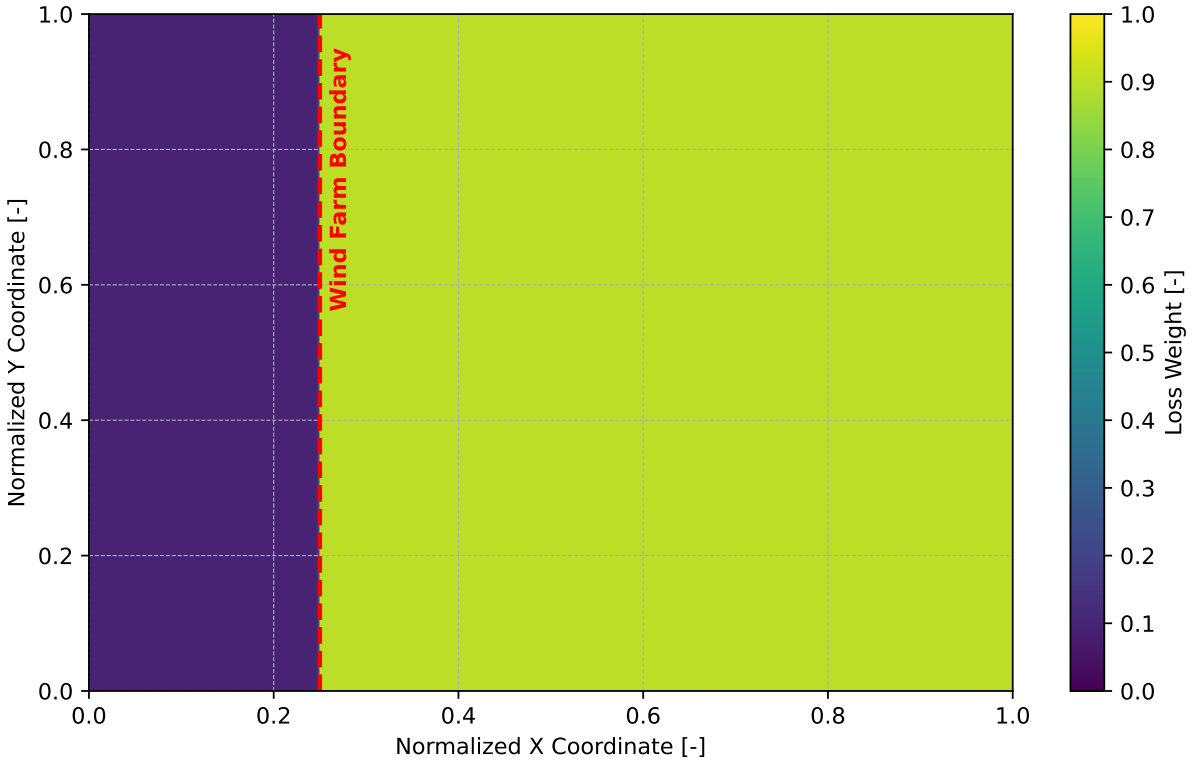


Figure 3.9: Weighted loss function distribution across the wind farm domain. The function assigns 10% weight to losses within the wind farm (left of the dashed line) and 90% weight to losses in the downstream wake region (right of the dashed line).

This weighting encourages the model to focus on accurately modeling long-distance wakes while still considering the flow within the wind farm. It's worth noting that alternative weighting schemes have been explored in similar studies. For instance, Bertolani [11] experimented with mathematically more complex loss functions in his Master's thesis, including weighting that increases inversely proportional to the distance from the wind farm center and functions that dampen the value of y using the hyperbolic tangent function. However, it was found that a simple linear weighting, similar to the approach used in this thesis, produced better models (smaller loss) than these more complex alternatives. By combining MSE with this weighted approach, the loss function is tailored to the specific requirements of long-distance wake modeling in wind farms, balancing the need for overall accuracy with a focus on downstream effects.

3.5.2 Data Transformation

To improve the model's ability to capture long-distance wake effects, a data transformation was applied to the wind speed and turbulence intensity data. The original dataset showed a heavily skewed distribution, with large variations near the wind farm and diminishing wake effects farther downwind.

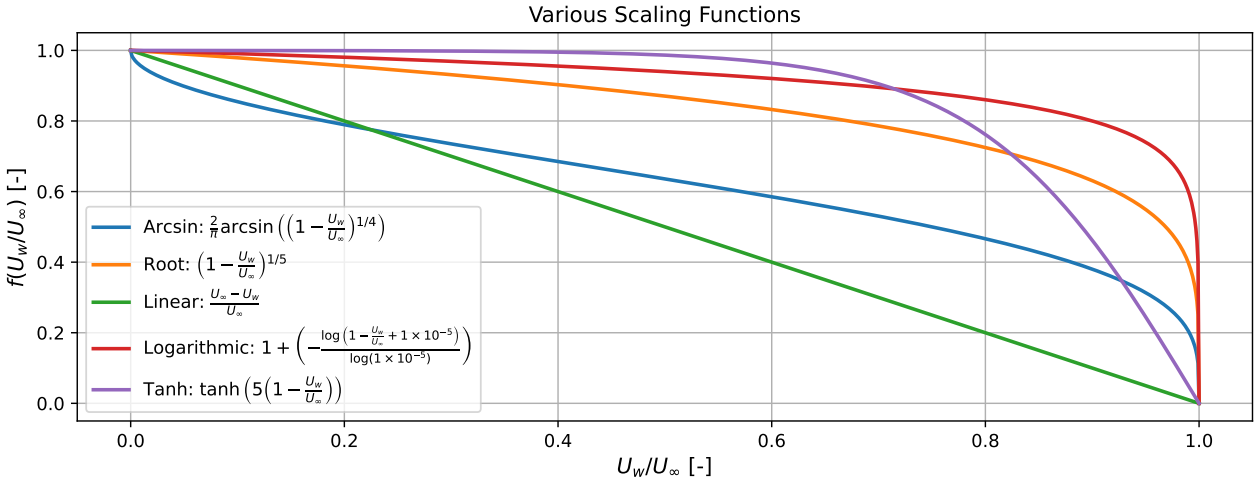


Figure 3.10: Overview over different data scaling functions considered for the transformation of wind speed and turbulence intensity.

This study focuses on modeling wake effects at greater distances from the wind farm, making it crucial to transform the data for a more uniform distribution. The transformation aims to increase the model's sensitivity to subtle changes in wind speed and turbulence intensity far downwind, where wake effects are less pronounced but still important for accurate long-distance modeling. Multiple different transformations have been considered (see Figure 3.10), however, the root method used for the transformation in this study is defined as:

$$y_{\text{root}} = \left(1 - \frac{x}{ws}\right)^{1/5} \tag{37}$$

This method stretches the data in regions with smaller wake effects, making the model more responsive to minor variations in these areas. It enhances the loss function's sensitivity to errors in the far-wake region, encouraging the model to capture and predict these subtle effects more accurately. The transformed dataset provides a more balanced representation of wake effects across the entire downwind distance. This approach ensures that the machine learning models can learn and predict long-distance wake characteristics more effectively, aligning with the study's primary goal of improving long-distance wind farm flow modeling. In Figure 3.11 and Figure 3.12 the effect of the data transformation can be seen for a wind farm of 200 wind turbines with a U_0 of 10 m/s and a Turbulence Intensity (TI) of 0.1. It should be noted that the TI transformation uses a root transformation exponent of 1/2 instead of the 1/5 used for the wind speed transformation. These exponents were chosen based on empirical testing, as they seemed to provide a favorable data transformation for model training, balancing the stretching of small variations in the far wake while maintaining the overall structure of the data.

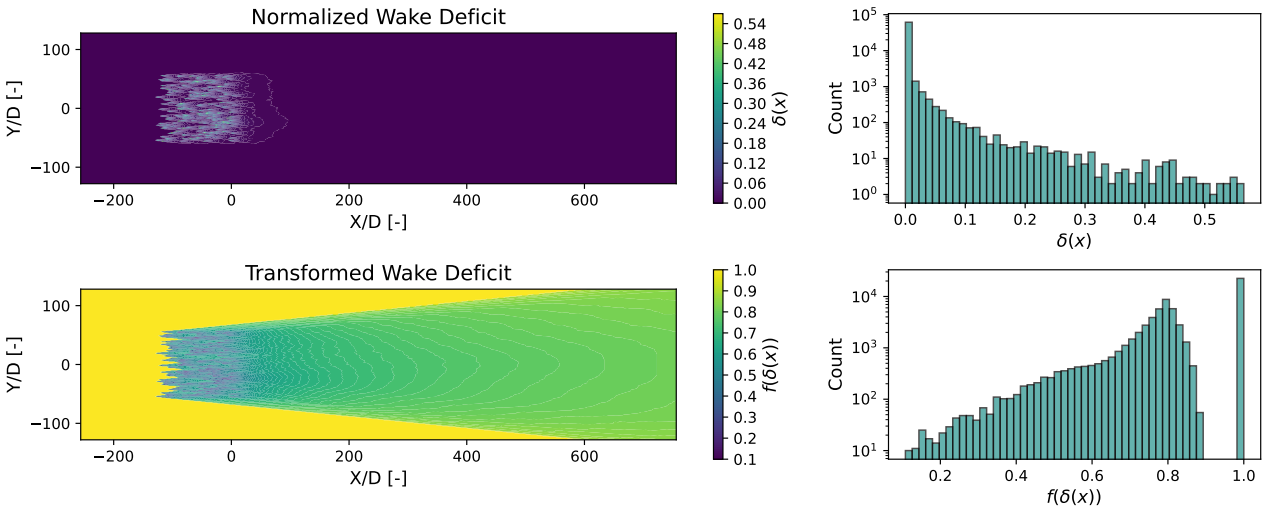


Figure 3.11: Comparison of original and transformed wind speed data. Left: Normalized wake deficit before and after transformation. Right: Histograms showing the distribution of normalized wake deficit before and after transformation. The transformation with an exponent of 1/5 stretches the data in regions with smaller wake effects, providing a more uniform distribution.

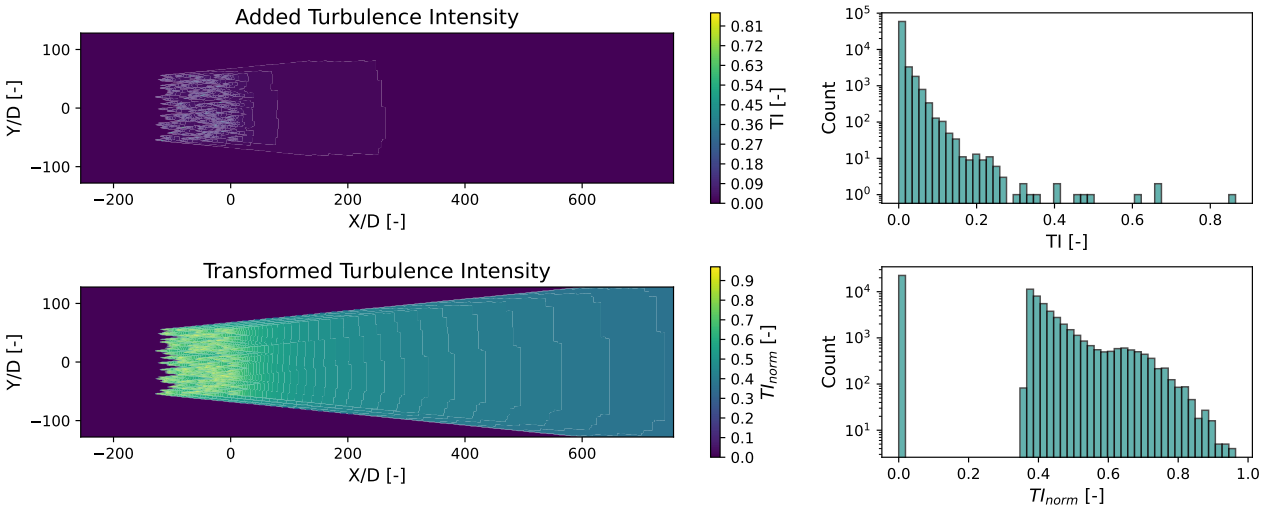


Figure 3.12: Comparison of original and transformed turbulence intensity data. Left: Added turbulence intensity before and after transformation. Right: Histograms showing the distribution of added turbulence intensity before and after transformation. The transformation with an exponent of 1/2 provides a more balanced representation of turbulence intensity variations across the domain.

The different exponents for wind speed (1/5) and turbulence intensity (1/2) were chosen based on the different characteristics of these parameters. Wind speed tends to have a more gradual decay in the far wake, requiring a stronger transformation (smaller exponent) to highlight these subtle changes. Turbulence intensity, on the other hand, typically shows more pronounced variations even in the far wake, so a less aggressive transformation (larger exponent) was found to be sufficient. These choices were made through small scale iterative testing to achieve the most favorable data distribution for model training, balancing the need to highlight far-wake effects while preserving the overall structure of the data.

3.5.3 Hyperparameters

The training process used the same hyperparameters for all models. Table 3.8 shows the key settings used. These hyperparameters were initially set using standard values commonly found in similar deep learning tasks and were then experimentally tested on a small scale. It's worth noting that advanced hyperparameter tuning techniques were not employed in this study, so there may be potential for slightly better results through more comprehensive optimization. A learning rate of 1×10^{-4} was chosen to balance speed and stability. The batch size was set to 4, limited by memory constraints. Training ran for 1000 epochs to ensure convergence. Adam optimizer was used for its adaptive capabilities (see subsubsection 2.3.2). The loss function used MSE with a 1:9 weighting (see subsubsection 3.5.1), focusing more on the wake region. A 30% dropout rate was applied to combat overfitting. The *CosineAnnealingLR* scheduler was used to gradually decrease the learning rate during training. This scheduler is typically used for fine-tuning, as it steadily reduces the learning rate, helping the model to settle into optimal parameter values [22].

Hyperparameter	**Value**
Learning Rate	1×10^{-4}
Batch Size	4
Number of Epochs	1000
Optimizer	Adam
Loss Function	MSE (1:9 Weighting)
Dropout Rate	30%
Learning Rate Scheduler	*CosineAnnealingLR*

Table 3.8: Training Hyperparameters used for all neural network models.

These hyperparameters were found to provide satisfactory results in the experimental testing. However, it's important to acknowledge that more extensive hyperparameter tuning, such as grid search or random search methods, could potentially yield marginally improved performance. The current set of hyperparameters represents a balance between model performance and computational resources available for this study.

4 Results and Analysis

This section presents a comprehensive comparison of the neural network models developed to predict wake deficits behind wind farms. The analysis focuses on prediction accuracy, computational efficiency, and training costs to assess each model's practical applicability in real-world wind farm optimization scenarios.

The examination begins with model accuracy, comparing predictions against ground truth data generated by established engineering models. This comparison highlights the predictive capabilities of each neural network architecture and their potential to match or surpass traditional methods. The analysis then addresses computational aspects, including inference time and resource requirements, which are critical in an industry where rapid decision-making can significantly improve energy yield. Training costs associated with each model are also explored, providing insight into the initial investment required for implementation and long-term viability in the wind energy sector. The investigation extends to model robustness and generalization capabilities across various wind farm configurations and environmental conditions, ensuring reliability in diverse real-world scenarios.

Additionally, a comparative analysis between these neural network approaches and traditional engineering models offers perspective on potential methodological shifts in wind farm wake modeling. This comprehensive evaluation aims to provide a clear understanding of each neural network model's strengths and limitations, guiding future applications and research directions in wind farm optimization. The results presented include quantitative performance metrics, visual representations of predictions versus actual wake deficits, computational efficiency data, and training cost analyses. Statistical analyses support the findings, offering a rigorous basis for model comparison. Through this thorough examination, the section seeks to contribute valuable insights to the field of wind farm modeling and optimization using machine learning techniques.

4.1 Performance Comparison of Wake Models

To evaluate the performance of different wake models, three representative wind parks were selected from the test sample group. These wind parks were chosen to showcase the models' effectiveness across a diverse range of wind farm characteristics. Table 4.1 presents the simulation data and parameters for these selected wind parks.The chosen wind parks exhibit significant variations in key attributes:

- **Wind Farm Size:** Ranging from smaller installations like Wind Park 987 with 37 turbines to larger facilities such as Wind Park 960 with 465 turbines.
- **Wind Speed:** Covering a broad spectrum from low wind speeds (Wind Park 987 at 4.70 m/s) to high wind speeds (Wind Park 960 at 19.95 m/s).
- **Turbulence Intensity:** Varying from 0.13 to 0.28, representing different atmospheric stability conditions.
- **Layout Complexity:** Differing in terms of turbine spacing, polygon irregularities, and number of vertices.

This diverse selection aims to demonstrate the robustness and versatility of the trained models across a wide range of wind park configurations and environmental conditions. By testing

the models on these varied scenarios, we can assess their generalisability and performance consistency.

Parameter	Wind Park 987	Wind Park 954	Wind Park 960
Wind Speed [m/s]	4.70	15.05	19.95
Turbulence Intensity [-]	0.13	0.28	0.19
Wind Direction [°]	270.0	270.0	270.0
Turbine Spacing [D]	11.50	9.52	7.25
Turbine Position Noise [-]	0.21	0.49	0.37
Polygon Avg Radii [D]	72.32	77.23	89.24
Polygon Irregularities [-]	0.47	0.01	0.16
Polygon Spikinesses [-]	0.47	0.32	0.25
Polygon Num Vertices [-]	5	10	15
Wind Turbine Amount [-]	37	230	465
Polygon Area [D^2]	4955.97	20749.96	24693.81
Polygon Roundness [-]	0.43	0.80	0.58

Table 4.1: Simulation data for selected wind farm layouts.

The subsequent analysis will focus on how different wake models perform across these diverse wind park configurations, providing insights into their applicability and limitations in various real-world scenarios.

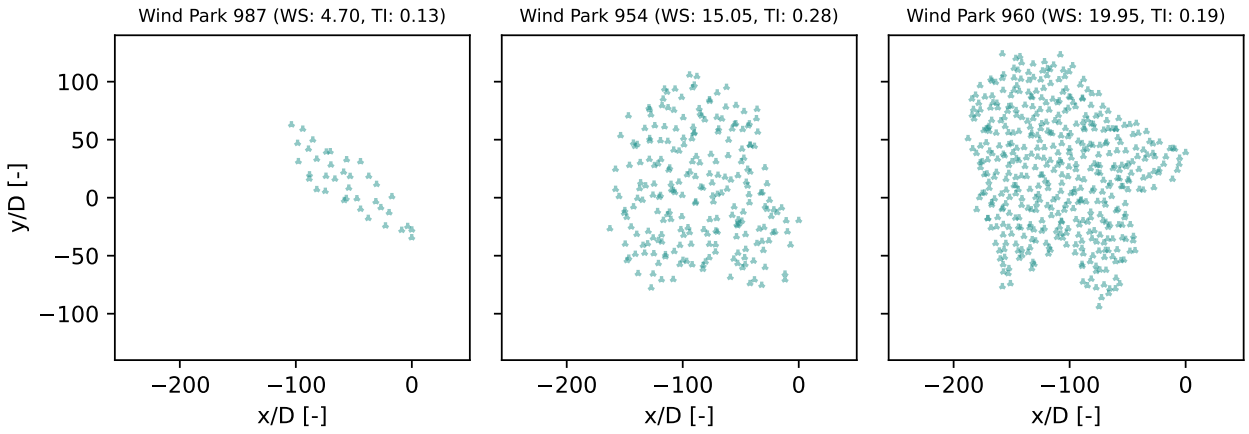


Figure 4.1: Layout of test wind parks.

A full overview of the layout of all 1000 wind parks used in this study can be found in Appendix D.

4.2 Neural Network Architecture Evaluation

Figure 4.2 illustrates the training and test MSE losses for four neural network architectures: CAE, U-Net, and their variants with MLP (CAE/MLP and U-Net/MLP) over 1000 epochs. Solid lines represent training losses, while dashed lines indicate test losses. All architectures exhibit a rapid initial decrease in losses, followed by a more gradual decline. The U-Net-based models achieve the lowest final MSE values, with U-Net/MLP slightly outperforming U-Net in both training and test scenarios. The CAE-based models show higher final MSE values compared to their U-Net counterparts. A small, consistent gap between training and test losses

is observed across all models, which is expected. The continuous decrease in test loss suggests that none of the models are overfitting to the training data, indicating good generalization capabilities.

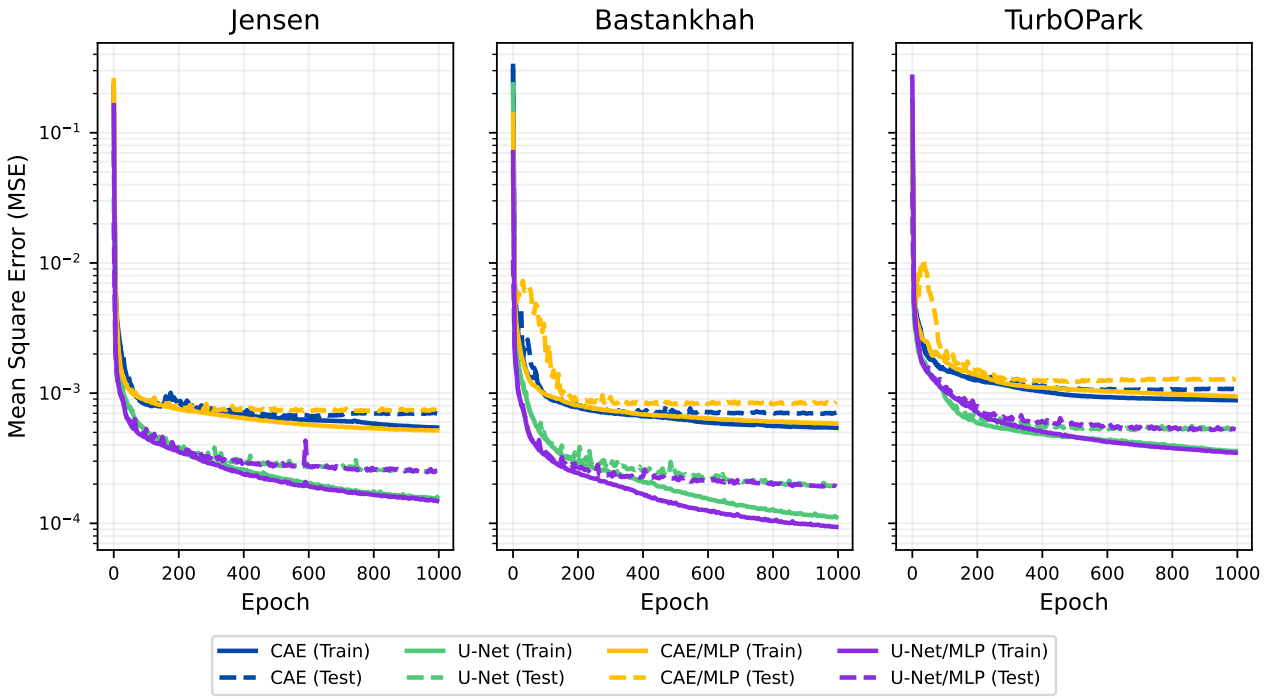


Figure 4.2: Comparison of training and test MSE losses for different neural network architectures and datasets over 1000 epochs.

The CAE/MLP model displays unusual behavior in its test loss curve between epochs 0 and 150 for the Bastankhah and TurbOPark test datasets, showing significant fluctuations before stabilizing. This could indicate initial instability in the learning process for this particular architecture. By the 1000th epoch, the test losses for all models appear to have largely plateaued. While the training losses for the U-Net and U-Net/MLP models show a continuing slight downward trend, this is not reflected in their test losses, suggesting that further training may not lead to improved generalization performance. It is important to note that the losses displayed in Figure 4.2 are weighted losses (see subsubsection 3.5.1). As such, they do not directly represent the absolute performance of the models. Instead, these losses serve two primary purposes: firstly, to provide a comparative measure of performance between different models, and secondly, to offer an overview of how effectively each model has been trained.

Model	Jensen		Bastankhah		TurbOPark	
	Train MSE	Test MSE	Train MSE	Test MSE	Train MSE	Test MSE
CAE	5.45×10^{-4}	7.00×10^{-4}	5.41×10^{-4}	6.99×10^{-4}	8.83×10^{-4}	1.08×10^{-3}
U-Net	1.55×10^{-4}	2.52×10^{-4}	1.10×10^{-4}	1.95×10^{-4}	3.58×10^{-4}	5.38×10^{-4}
CAE/MLP	5.16×10^{-4}	7.37×10^{-4}	5.84×10^{-4}	8.41×10^{-4}	9.45×10^{-4}	1.27×10^{-3}
U-Net/MLP	$\mathbf{1.51 \times 10^{-4}}$	$\mathbf{2.46 \times 10^{-4}}$	$\mathbf{9.39 \times 10^{-5}}$	$\mathbf{1.94 \times 10^{-4}}$	$\mathbf{3.46 \times 10^{-4}}$	$\mathbf{5.29 \times 10^{-4}}$

Table 4.2: Comparison of Train MSE and Test MSE for different models and datasets. Boldface indicates the best (lowest) value for each metric and dataset.

4.2.1 Error Analysis

To assess the accuracy of the neural networks, we examine the MAE and MSE as described in subsubsection 3.5.1. During training, a weighted loss function was employed to calculate losses, which provides insight into the relative performance of individual models but does not accurately represent their absolute accuracy. Furthermore, after training, the models are placed in inference mode, disabling the dropout layer, which may alter accuracy performance. Given these considerations, we recalculate the final MAE and MSE post-training. As the primary objective of these models is to assess wake deficit downwind of wind farms, we focus our error calculations on the downwind portion of the ground truth and neural network output (specifically, the last 3/4 of the array in the downwind direction). Table 4.3 presents the results of these MAE and MSE calculations for each model across three different test datasets: Jensen, Bastankhah, and TurbOPark.

Model	Jensen		Bastankhah		TurbOPark	
	MAE [m/s]	MSE [m^2/s^2]	MAE [m/s]	MSE [m^2/s^2]	MAE [m/s]	MSE [m^2/s^2]
CAE	7.53×10^{-4}	1.02×10^{-4}	1.46×10^{-3}	3.29×10^{-4}	1.47×10^{-2}	4.80×10^{-3}
U-Net	$\mathbf{4.75 \times 10^{-4}}$	4.30×10^{-5}	$\mathbf{9.34 \times 10^{-4}}$	$\mathbf{1.19 \times 10^{-4}}$	$\mathbf{1.24 \times 10^{-2}}$	$\mathbf{3.38 \times 10^{-3}}$
CAE/MLP	9.05×10^{-4}	1.15×10^{-4}	2.02×10^{-3}	4.07×10^{-4}	1.64×10^{-2}	5.59×10^{-3}
U-Net/MLP	5.46×10^{-4}	$\mathbf{4.20 \times 10^{-5}}$	1.00×10^{-3}	1.24×10^{-4}	1.44×10^{-2}	4.01×10^{-3}

Table 4.3: Comparison of MAE and MSE for different models and test datasets. Boldface indicates the best (lowest) value for each metric and dataset.

The results in Table 4.3 demonstrate the performance of different models across the three wake models. U-Net based architectures consistently outperform the CAE models. Minimal variance is observed between the base models (CAE and U-Net) and their amplified versions incorporating a MLP in the latent space. U-Net models exhibit particularly strong performance with the Bastankhah and TurbOPark models. This suggests that the U-Net architecture is better suited to capturing the intricate details in these more sophisticated wake models. A clear trend is evident in the error metrics, which increase progressively from Jensen to Bastankhah to TurbOPark. This progression corresponds to the increasing complexity of these wake models. The rising errors for more complex wake models indicate that accurate prediction of wake effects becomes more challenging as the data complexity increases. This pattern of increasing difficulty with data complexity can be observed in both classification and regression tasks [54].

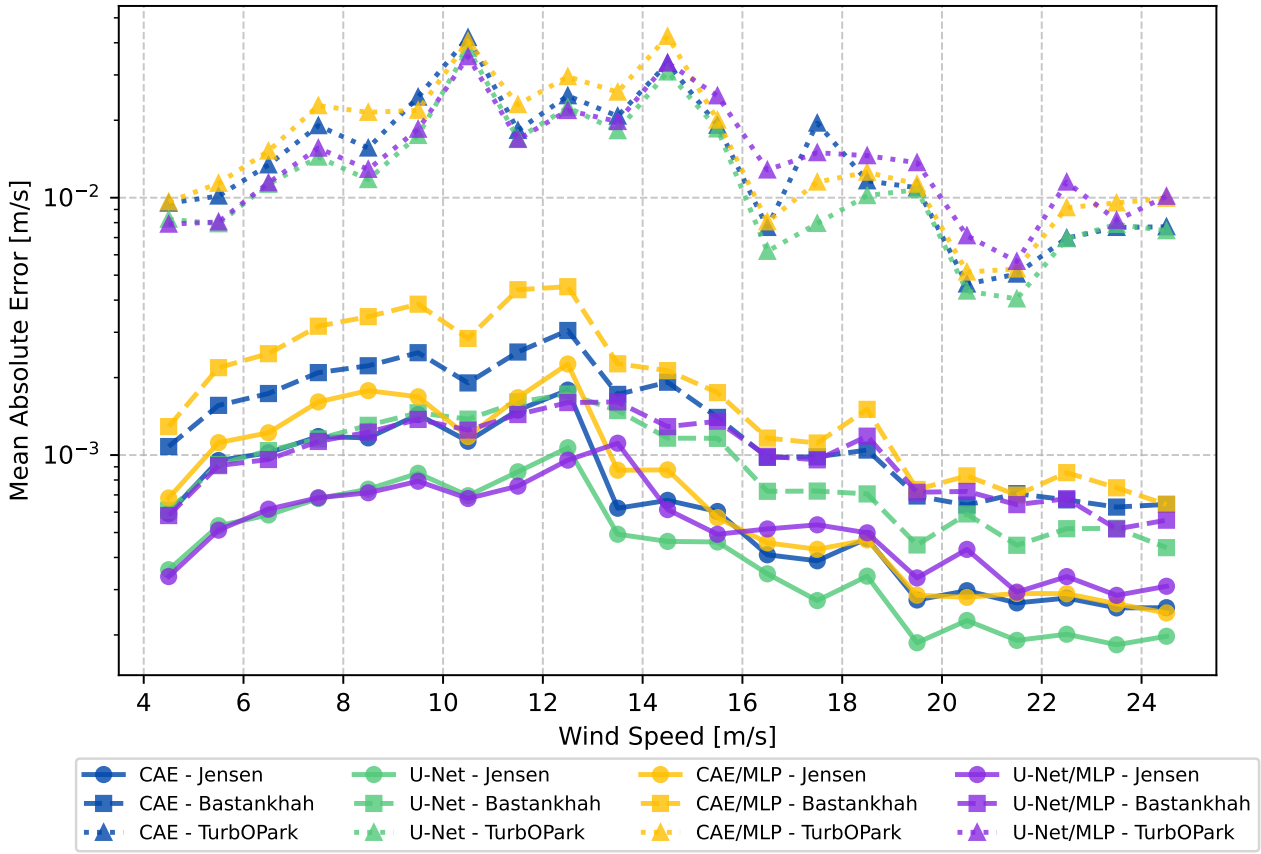


Figure 4.3: MAE distribution across wind speed bins (1 m/s) for four neural network models (CAE, U-Net, CAE/MLP, U-Net/MLP) trained on different datasets.

Figure 4.3 displays the MAE distribution across 1 m/s wind speed bins for the four neural network models described in subsection 3.4. Each model's performance is shown for different neural network models, represented by separate bars within each bin. The plot reveals a consistent trend across all models and training datasets: The MAE is not constant across wind speeds but follows a distinct pattern. Starting with relatively low values at 4 m/s, the error increases, reaching its peak between 9-13 m/s, and then gradually decreases for higher wind speeds. This trend is visible across all training dataset and neural network models. This pattern suggests that the models' prediction accuracy varies significantly with wind speed. The lower MAE at low wind speeds might be due to more consistent and predictable wake behavior in these conditions. The peak error in the 9-11 m/s range could indicate more complex wake interactions and variability at these wind speeds. The gradual decrease in MAE for higher wind speeds might be attributed to more uniform flow conditions or reduced wake effects at these velocities. This trend is consistent across all four neural network architectures, indicating a fundamental characteristic of wind farm wake behavior rather than a model-specific phenomenon.

4.3 Long-Distance Flow Prediction Accuracy

In the following section, wake predictions of the different neural network architectures trained on the Jensen, Bastankhah, and TurbOPark datasets are compared against the three wind parks detailed in subsection 4.1.

4.3.1 Jensen Dataset

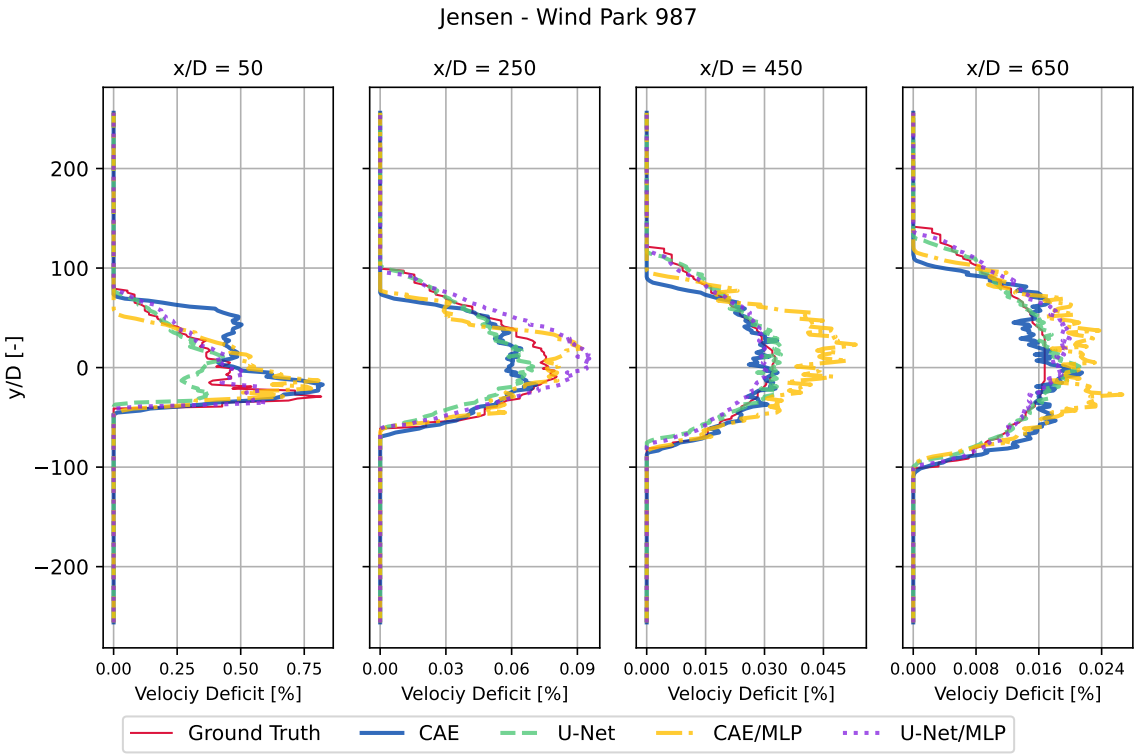


Figure 4.4: Wake deficit profiles for Wind Park 987 using the Jensen model. Profiles shown at multiple downstream distances (x/D) behind wind farms for all neural network architectures.

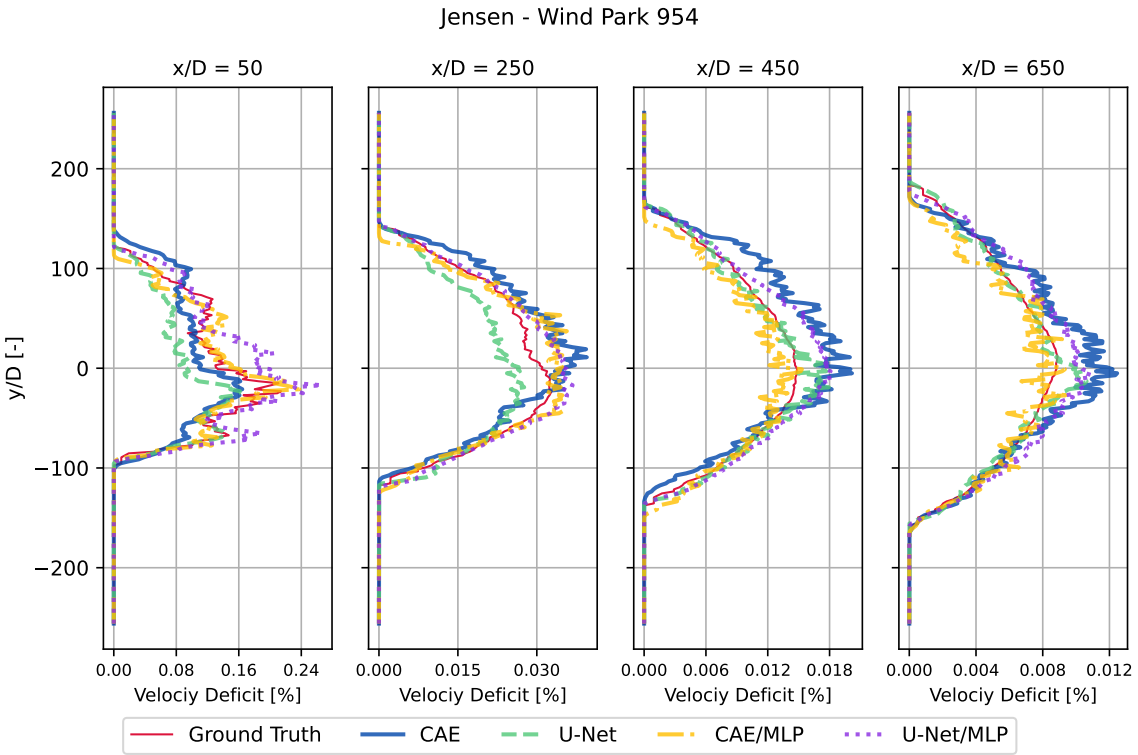


Figure 4.5: Wake deficit profiles for Wind Park 954 using the Jensen model. Profiles shown at multiple downstream distances (x/D) behind wind farms for all neural network architectures.

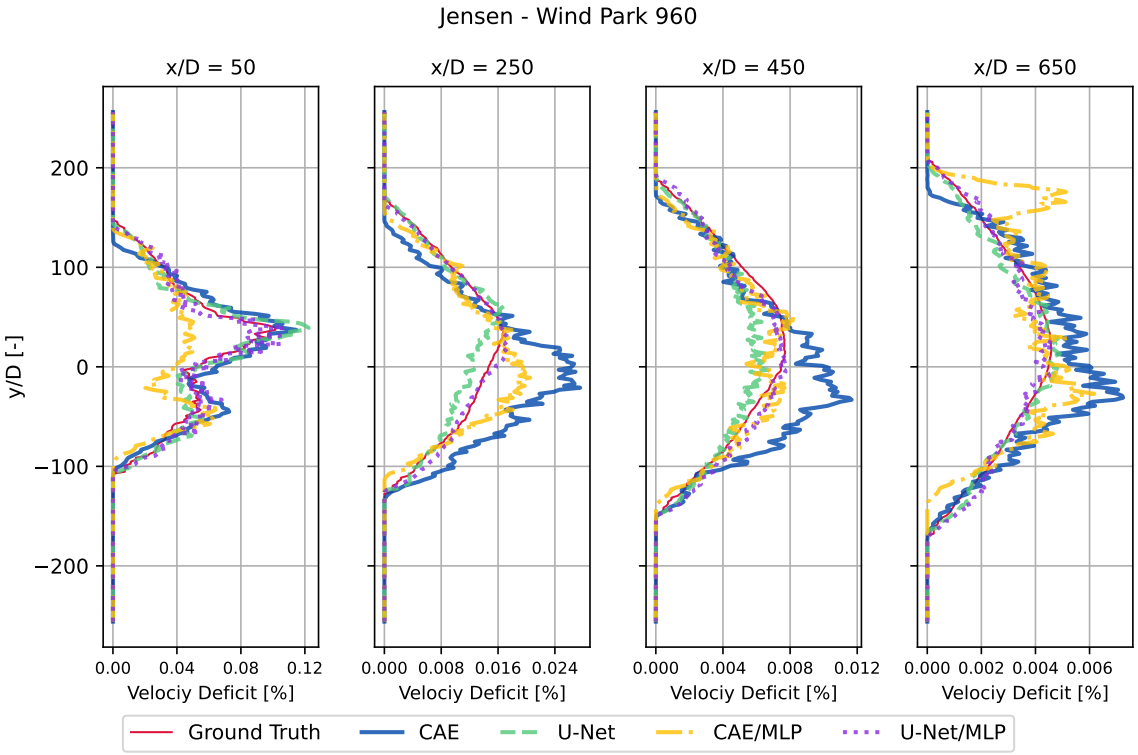


Figure 4.6: Wake deficit profiles for Wind Park 960 using the Jensen model. Profiles shown at multiple downstream distances (x/D) behind wind farms for all neural network architectures.

For the Jensen dataset, all neural network architectures show good agreement with the ground truth. The U-Net and U-Net/MLP models consistently outperform the CAE and CAE/MLP models across all wind parks. This is especially noticeable for Wind Park 960, where the U-Net variants capture the wake deficit more accurately at all downstream distances. For smaller wind parks (987 and 954), the differences between models are less pronounced, but U-Net and U-Net/MLP still show slightly better performance.

4.3.2 Bastankhah Dataset

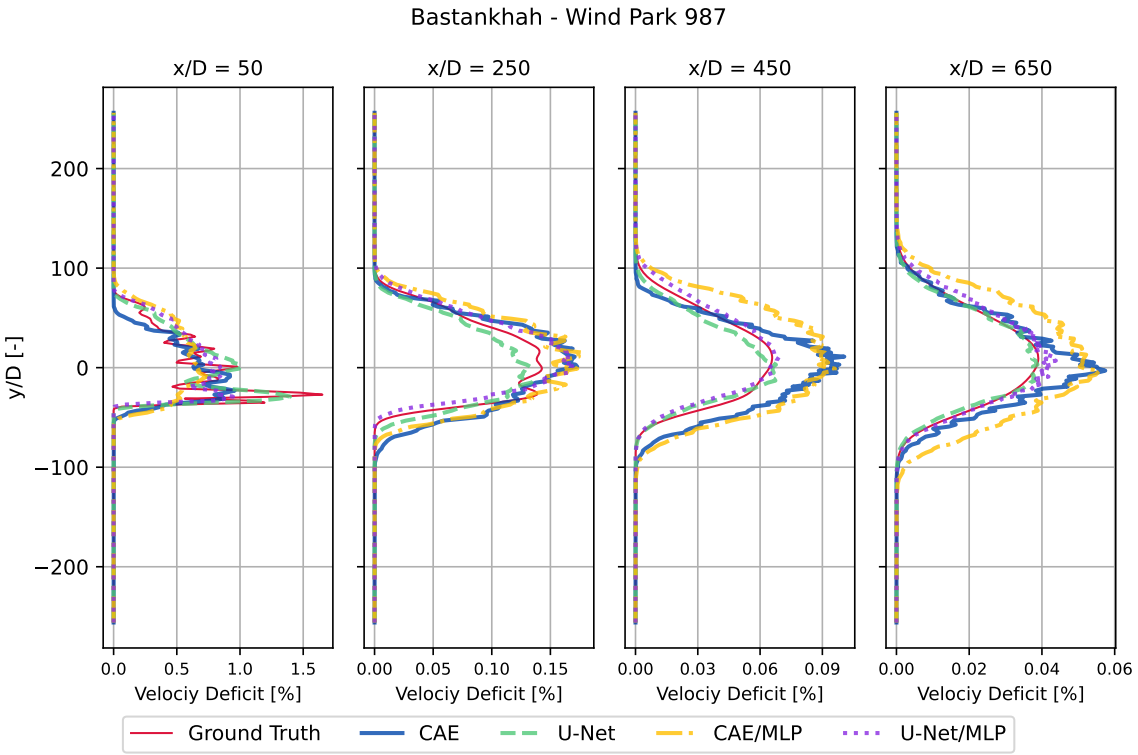


Figure 4.7: Wake deficit profiles for Wind Park 987 using the Bastankhah model. Profiles shown at multiple downstream distances (x/D) behind wind farms for all neural network architectures.

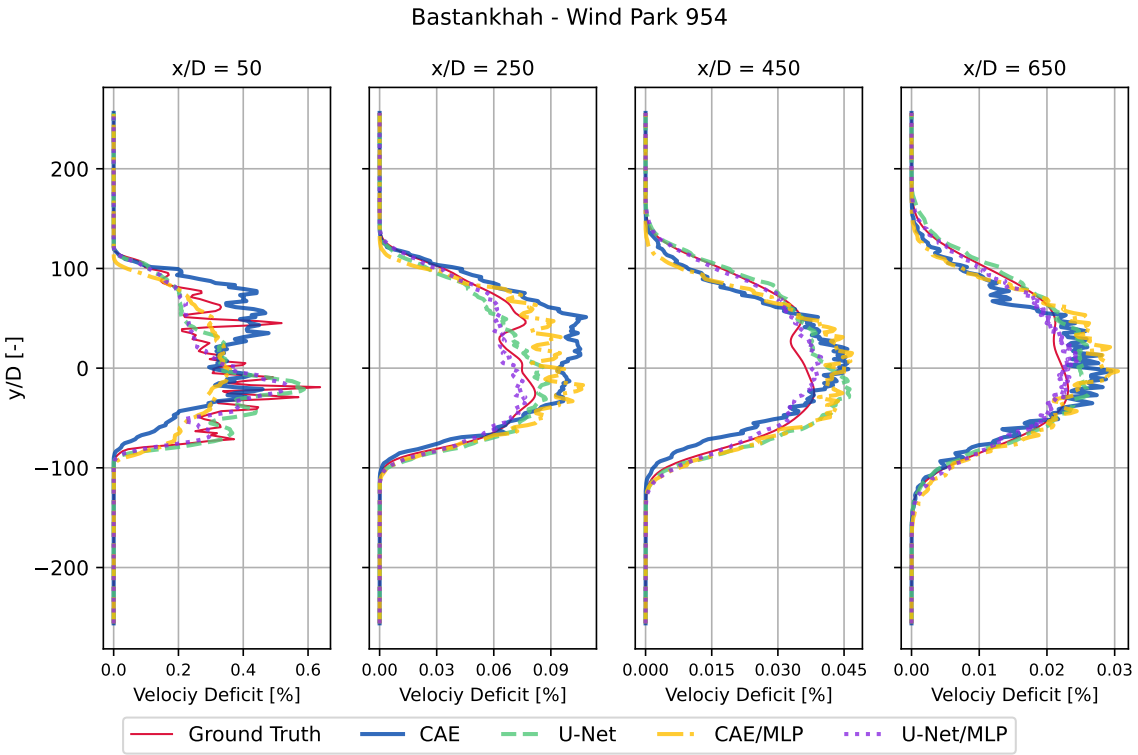


Figure 4.8: Wake deficit profiles for Wind Park 954 using the Bastankhah model. Profiles shown at multiple downstream distances (x/D) behind wind farms for all neural network architectures.

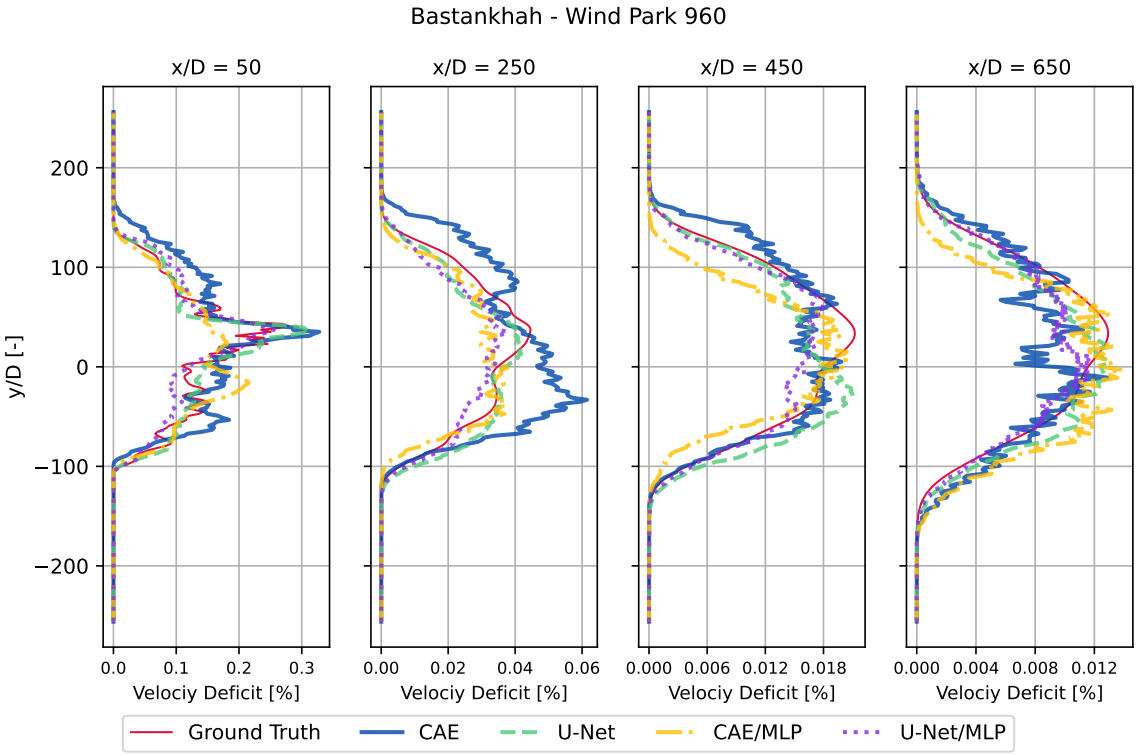


Figure 4.9: Wake deficit profiles for Wind Park 960 using the Bastankhah model. Profiles shown at multiple downstream distances (x/D) behind wind farms for all neural network architectures.

Similar trends are observed for the Bastankhah dataset. The U-Net and U-Net/MLP models again show superior performance, particularly for Wind Park 987. All models capture the smoother wake profiles characteristic of the Bastankhah model. For larger wind parks, the differences between models are less significant, but U-Net variants still maintain a slight edge in accuracy.

4.3.3 TurbOPark Dataset

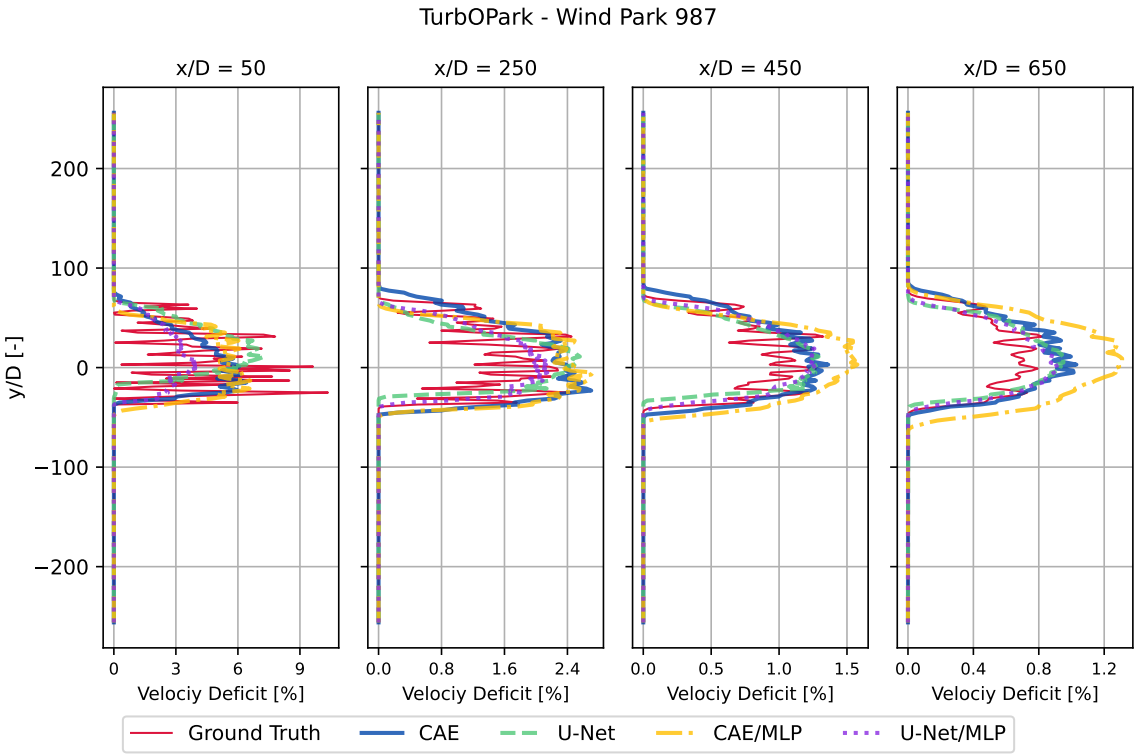


Figure 4.10: Wake deficit profiles for Wind Park 987 using the TurbOPark model. Profiles shown at multiple downstream distances (x/D) behind wind farms for all neural network architectures.

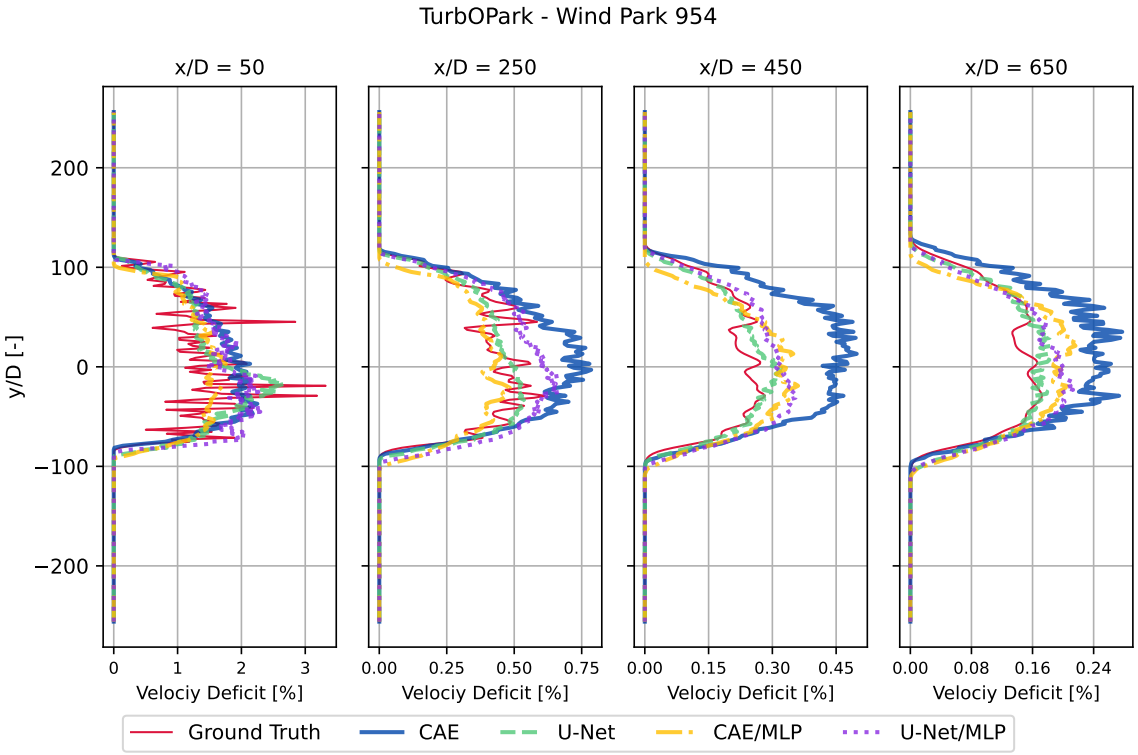


Figure 4.11: Wake deficit profiles for Wind Park 954 using the TurbOPark model. Profiles shown at multiple downstream distances (x/D) behind wind farms for all neural network architectures.

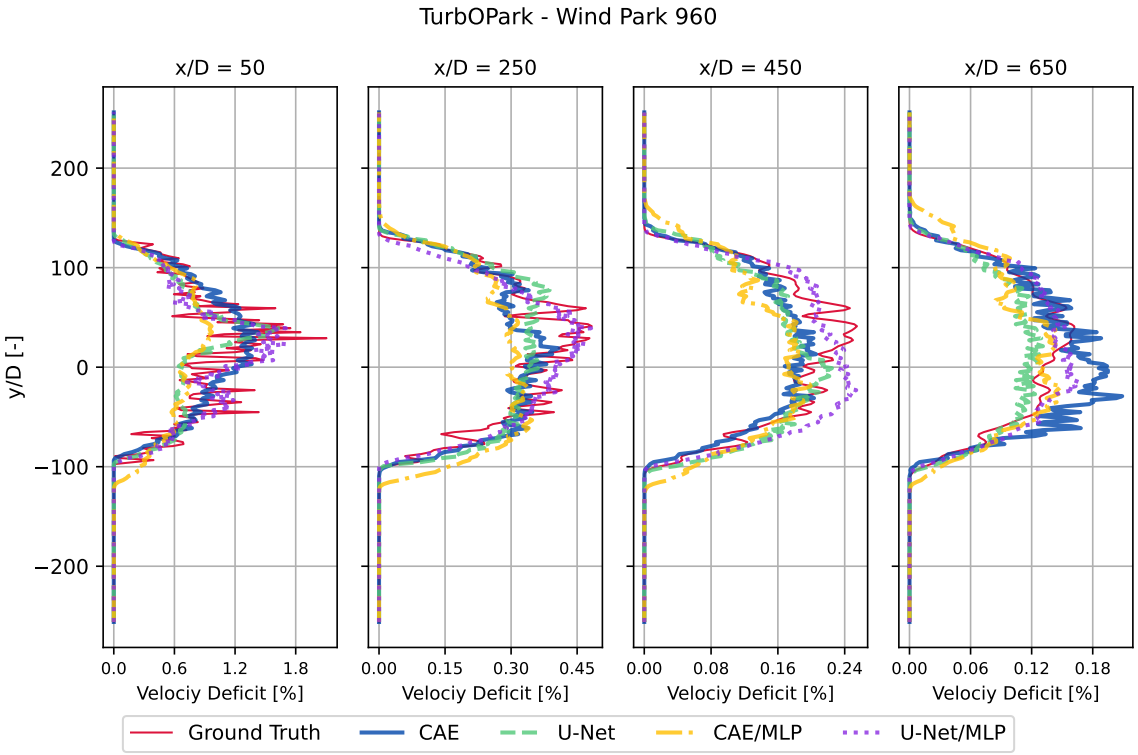


Figure 4.12: Wake deficit profiles for Wind Park 960 using the TurbOPark model. Profiles shown at multiple downstream distances (x/D) behind wind farms for all neural network architectures.

The TurbOPark dataset presents the greatest challenge for all neural network architectures. This is evident in the noticeably reduced prediction accuracy compared to the Jensen and Bastankhah datasets. The difficulty is most pronounced for Wind Park 987, where sharp wake deficit spikes occur directly behind turbines. These spikes are particularly prominent at closer distances (x/D = 50) and become less defined further downstream (x/D = 650) as individual wakes mix. U-Net and U-Net/MLP models still outperform their CAE counterparts, but the gap in performance is less significant. All models struggle to accurately capture the complex wake interactions predicted by TurbOPark, especially for smaller wind parks. This difficulty decreases for larger wind parks and at greater downstream distances, where wake profiles become smoother and more uniform. The reduced performance on the TurbOPark dataset likely stems from its higher complexity and variability compared to the Jensen and Bastankhah models. This suggests that more advanced neural network architectures or training strategies might be necessary to fully capture the intricacies of more sophisticated wake models like TurbOPark.

For a comprehensive comparison of 2D flow patterns across various neural network architectures, training datasets, and wind parks, refer to Appendix C.

4.3.4 Wake-generated Turbulence Prediction

As shown in Table 3.2, the three datasets (Jensen, Bastankhah, and TurbOPark) used for training the neural network models have also been set up to calculate the turbulence intensity of the wake using the turbulence models presented in subsubsection 2.2.4. The U-Net/MLP model presented in subsubsection 3.4.4 has additionally been trained on the Crespo-Hernández turbulence model, as well as the two versions of the Frandsen turbulence model, to test whether these CNN approaches to estimating the wake effects can also be applied to added turbulence estimations. The models are compared using the relative added turbulence intensity, which is calculated as follows:

$$I_{add,rel} = \frac{I - I_0}{I_0} \times 100\% \tag{38}$$

where $I_{add,rel}$ is the relative added turbulence intensity (in percentage), I is the total turbulence intensity at a given point, and I_0 is the ambient turbulence intensity.

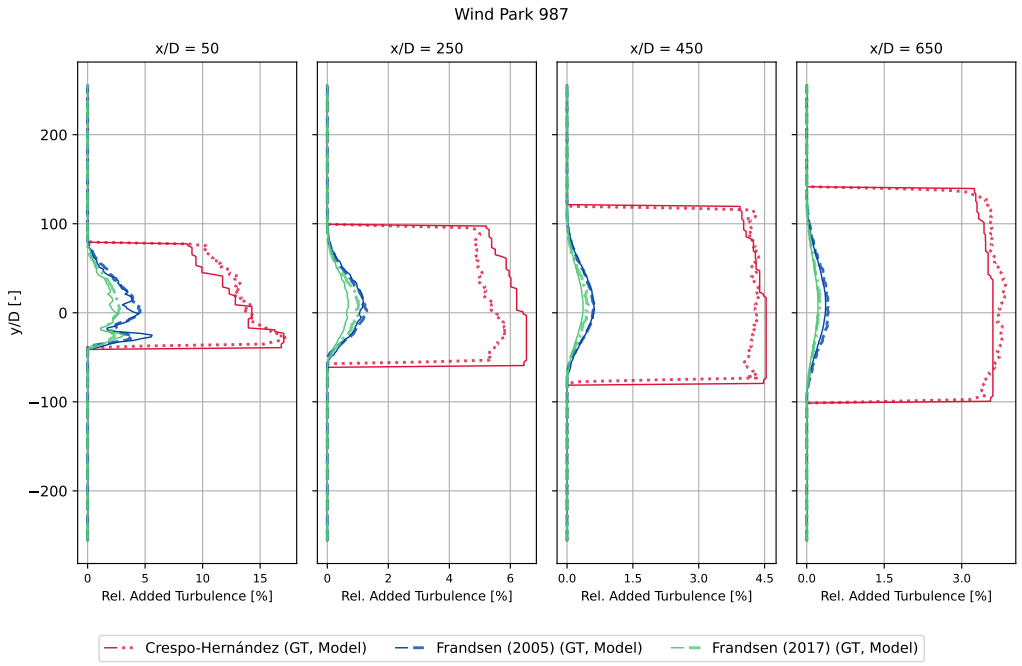


Figure 4.13: Comparison of turbulence models for Wind Park 987. The figure shows the relative added turbulence intensity for the Crespo-Hernández model, Frandsen (2005) model, and Frandsen (2017) model, alongside predictions from the U-Net/MLP neural network. Solid lines represent the ground truth data for each model, while dashed or dotted lines indicate the corresponding neural network predictions. The comparison is made at four different downstream distances (x/D) from the wind turbine.

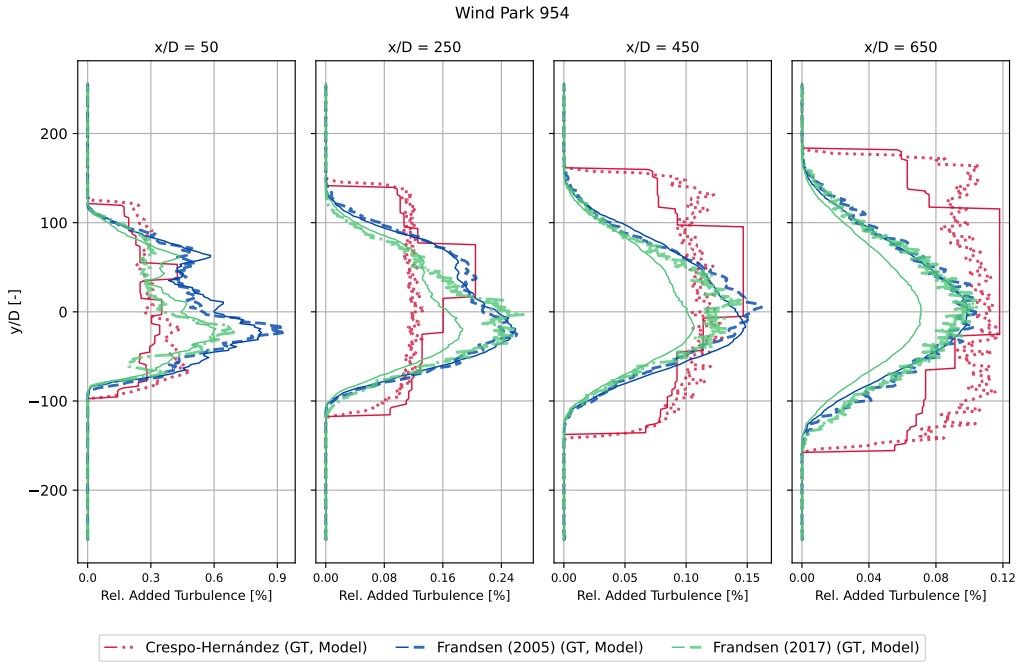


Figure 4.14: Comparison of turbulence models for Wind Park 954. The figure shows the relative added turbulence intensity for the Crespo-Hernández model, Frandsen (2005) model, and Frandsen (2017) model, alongside predictions from the U-Net/MLP neural network. Solid lines represent the ground truth data for each model, while dashed or dotted lines indicate the corresponding neural network predictions. The comparison is made at four different downstream distances (x/D) from the wind turbine.

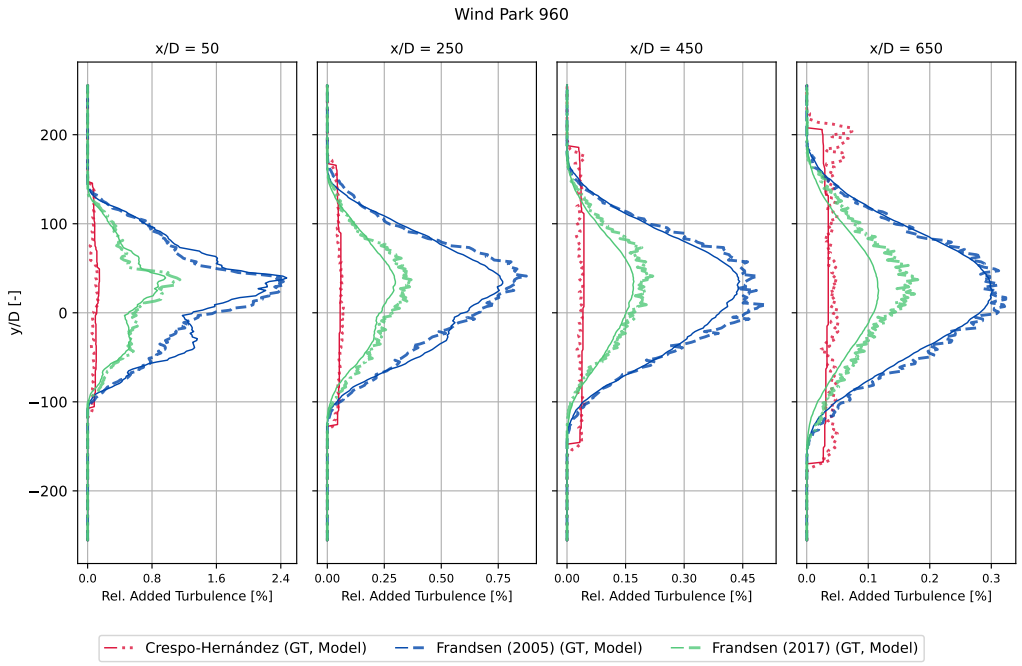


Figure 4.15: Comparison of turbulence models for Wind Park 960. The figure shows the relative added turbulence intensity for the Crespo-Hernández model, Frandsen (2005) model, and Frandsen (2017) model, alongside predictions from the U-Net/MLP neural network. Solid lines represent the ground truth data for each model, while dashed or dotted lines indicate the corresponding neural network predictions. The comparison is made at four different downstream distances (x/D) from the wind turbine.

Figure 4.13, Figure 4.14, and Figure 4.15 present comparisons of turbulence models for Wind Parks 987, 954, and 960, respectively. These figures illustrate the relative added turbulence intensity as predicted by the Crespo-Hernández model, Frandsen (2005) model, and Frandsen (2017) model, alongside predictions from the U-Net/MLP neural network. The solid lines represent ground truth data for each model, while dashed or dotted lines indicate the corresponding neural network predictions. These comparisons are made at four different downstream distances (x/D) from the wind turbine. A notable observation from these figures is the significant variation in turbulence estimates between the Crespo-Hernández model and the two Frandsen models. In Wind Park 987, characterized by the lowest ambient turbulence intensity and inflow wind speed, the Crespo-Hernández model predicts substantially higher relative added turbulence compared to the Frandsen models. Conversely, for the largest wind park (Wind Park 960), both Frandsen models predict significantly higher turbulence levels than the Crespo-Hernández model. This divergence highlights the importance of model selection in turbulence estimation and underscores the need for a versatile approach that can accommodate different models.

The U-Net/MLP model demonstrates promising performance in following the ground truth data across most scenarios. However, some limitations are evident, particularly in Wind Park 954 (Figure 4.14), where the neural network model trained on Crespo-Hernández data struggles to accurately replicate the square-shaped curve of the Crespo-Hernández ground truth. This challenge suggests that while the U-Net/MLP approach is generally effective, it may require further refinement to capture certain complex turbulence patterns.

Model	Crespo-Hernández		Frandsen (2005)		Frandsen (2017)	
	MAE [-]	MSE [-]	MAE [-]	MSE [-]	MAE [-]	MSE [-]
U-Net/MLP	2.21×10^{-4}	6.40×10^{-7}	1.87×10^{-4}	2.26×10^{-6}	1.30×10^{-4}	2.46×10^{-6}

Table 4.4: Performance evaluation of the U-Net/MLP neural network in predicting turbulence intensity for different turbulence models. The table shows the MAE and MSE values for the Crespo-Hernández, Frandsen (2005), and Frandsen (2017) models, indicating the accuracy of the neural network's predictions compared to the ground truth data from each model.

Table 4.4 provides a quantitative assessment of the U-Net/MLP neural network's performance across the three turbulence models. MAE and MSE values indicate that the neural network's predictions for the Crespo-Hernández model are slightly less accurate compared to those for the Frandsen models. However, the differences in MAE and MSE across all three models are relatively small, suggesting that the U-Net/MLP can model different turbulence patterns with comparable levels of accuracy. These results demonstrate the potential of CNN-based approaches in estimating wake-generated turbulence across various turbulence models. The ability to adapt to different models with similar levels of accuracy is a significant advantage, offering flexibility in wind farm modeling and analysis.

For a comprehensive comparison of 2D flow patterns across various neural network architectures, training datasets, and wind parks, refer to subsection C.5.

4.4 Computational Efficiency

To assess the computational efficiency of the Jensen, Bastankhah, and TurbOpark models as implemented in PyWake, a comprehensive benchmark study was conducted. The evaluation was conducted on a system featuring an AMD Ryzen 7 5800H CPU with 16 cores clocked at 4.463 GHz. The system includes 16 GB of DDR4 memory running at 3200 MT/s. For dedicated graphics processing, it is equipped with an NVIDIA GeForce RTX 3060 Max-Q GPU. All simulations utilized PyWake's engineering wind farm model, specifically the *PropagateDownwind* implementation. The benchmark tests were carried out on square wind farm layouts, ranging from a minimal 1x1 configuration (single turbine) to a maximum of 14x14 (196 turbines) with a resolution of 512 x 256. This upper limit was imposed due to memory constraints of the test system, which became unstable for larger layouts.

The models were evaluated under uniform inflow conditions, with a constant wind speed of $U_0 = 10\,\mathrm{m/s}$ and a turbulence intensity of $I_0 = 0.1$. To ensure statistical reliability and mitigate the impact of computational variability, each simulation was repeated 10 times for every wind farm configuration. This methodical approach allows for a robust comparison of the models' performance across a range of farm sizes, providing insights into their scalability and computational demands. The repeated trials for each configuration enhance the reliability of the results by accounting for potential variations in execution time.

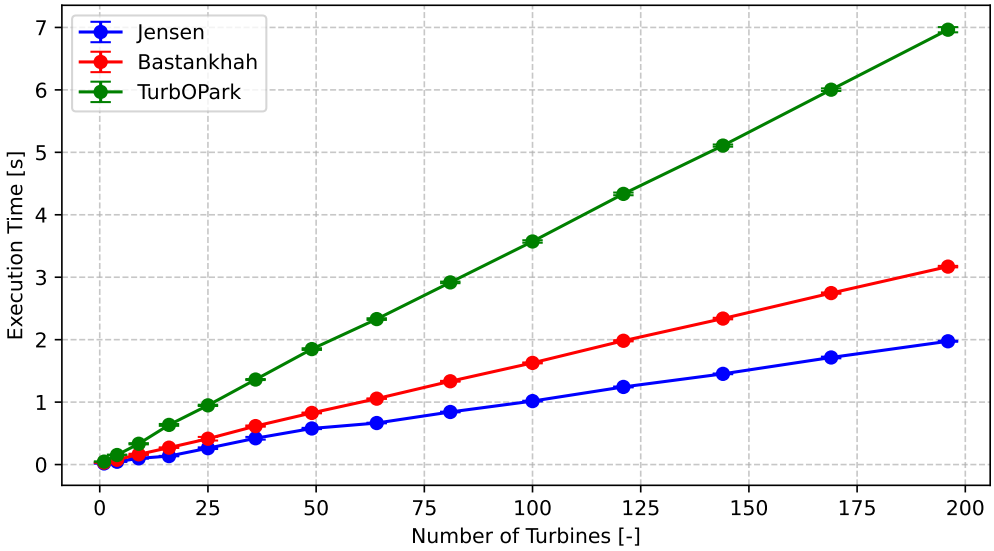


Figure 4.16: Comparison of execution time required for different engineering models and various wind farm sizes. The points represent mean execution times, with error bars indicating standard deviation.

The results of the computational efficiency simulations are presented in Figure 4.16. A clear linear relationship is observed between the execution time and the number of turbines in the wind farm for all three models. This linear trend aligns with expectations, given that the *PropagateDownwind* model iterates over all turbines in downstream order exactly once. The strong linearity of this relationship allows for the calculation of a slope, which can be used to estimate computational time for larger wind farms beyond the capabilities of the current test system. Table 4.5 presents the calculated slope for each engineering model, along with the coefficient of determination (R^2).

Model	**Slope [sec/turbine]**	$\mathbf{R^2}$
Jensen	1.00×10^{-2}	0.9973
Bastankhah	1.61×10^{-2}	0.9995
TurbOPark	3.53×10^{-2}	0.9992

Table 4.5: Linear fit results for execution time per number of turbines.

The high R^2 values (all > 0.997) confirm the strong linear relationship between execution time and turbine count. Among the three models, Jensen exhibits the lowest slope, indicating it is the most computationally efficient. Bastankhah shows moderate computational demands, while TurbOPark, with the highest slope, is the most computationally intensive. These differences in computational efficiency likely reflect the varying complexities of the wake modeling approaches employed by each model.

Neural network models present a stark contrast to traditional engineering models in terms of performance. After initial training, these networks can estimate wake deficits several orders of magnitude faster than their engineering counterparts. A key advantage of neural networks is their invariant calculation time regardless of wind park size; estimating the wake deficit behind a single turbine takes the same time as for a 200-turbine wind farm. Figure 4.17 illustrates the execution times of various neural networks, using the same wind park layouts as in Figure 4.16. A notable difference is observed between CPU and GPU performance, with GPU

implementations generally 3-5 times faster. This performance gap is attributed to the GPU's architecture, which features thousands of cores designed for parallel computation, making it ideal for the matrix operations prevalent in neural networks. GPUs are optimized for deep learning calculations, particularly floating-point operations and matrix multiplication.

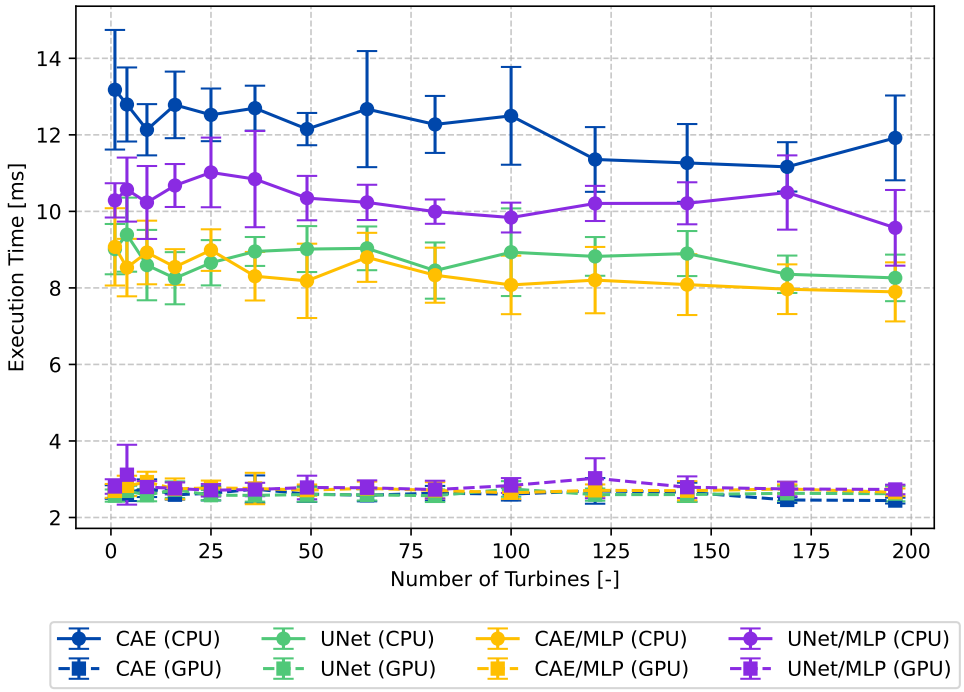


Figure 4.17: Execution time comparison for different neural network models across various wind farm sizes. Points represent mean execution times, with error bars indicating standard deviation.

As evident from Figure 4.17 and Table 4.6, GPU evaluations typically require 2.6-2.8 ms, while CPU evaluations take 8.4-12.2 ms. The CAE/MLP model performs fastest on CPU, while the CAE and UNet models tie for GPU execution speed. Notably, even for a single turbine, neural networks outpace PyWake models (see Table 4.5).

Model	**CPU Mean Time [ms]**	**GPU Mean Time [ms]**
CAE	12.24	**2.62**
U-Net	8.76	**2.62**
CAE/MLP	**8.42**	2.74
U-Net/MLP	10.32	2.81

Table 4.6: Mean execution times for different neural network models on CPU and GPU.

When comparing Figure 4.16 and Figure 4.17, it is evident that the execution time for the engineering models increases linearly ($\mathcal{O}(n)$), whereas the neural networks model maintains a constant execution time regardless of wind park size ($\mathcal{O}(1)$). This indicates that as the wind park size grows, the performance advantage of using neural networks will increase linearly. Consequently, larger wind farms, which significantly influence downwind wind speeds, would benefit most from neural network approaches for estimating wake deficits. Therefore, assessing the wake velocity impact of these larger wind parks becomes increasingly crucial.

4.4.1 Training Time

When evaluating the overall efficiency of machine learning models compared to standard engineering models, it is important to consider the initial computational cost associated with training. While machine learning models can be computationally efficient once trained (as discussed in subsection 4.4), they require a significant upfront investment in processing time and resources. GPU acceleration plays a vital role in reducing this initial overhead. GPUs excel at performing numerous simple calculations in parallel, which is particularly beneficial for the matrix operations that form the core of many machine learning algorithms. For this study, we utilized the DTU High Performance Computing (HPC) cluster, specifically leveraging Tesla V100 GPUs. Each of these GPUs boasts 5120 CUDA cores, 640 tensor cores, and a peak single-precision performance of 14.13 TFLOPS.

Table 4.7 presents a comparison of the total training time for the four neural network models examined in this study. The reported training times represent an average across the three training datasets: Jensen, Bastankhah, and TurbOPark.

Table 4.7: Comparison of training time for the neural network models.

Model	**Training Time [hr]**	**Total Epochs**
CAE	**14.81**	1000
U-Net	15.46	1000
CAE/MLP	16.00	1000
U-Net/MLP	15.38	1000

As seen from Table 4.7, the basic CAE neural network shows the shortest training time. However, it's important to note that, as discussed in subsection 4.1, this model also demonstrates the poorest performance among all tested models. The differences in training time between the models are minimal, with variations of 8% between the fastest and slowest models. Such small differences could be attributed to factors beyond the model architecture itself, such as temporary fluctuations in GPU performance or other system-level variations. These results suggest that, for the models and datasets considered in this study, the choice between these architectures should be primarily driven by their performance characteristics rather than differences in training time. The similar training times across models indicate that the computational cost of training is not a significant differentiating factor in this comparison.

The computational feasibility of neural network models must also account for the time required to generate training data. As shown in Table 4.5, engineering wake deficit models have low computational costs, enabling the generation of a 1000 wind park dataset (like the one used in this study) within hours, depending on available resources. However, using higher-fidelity simulations such as RANS or LES for training data could significantly increase upfront computational costs. Conversely, if suitable data already exists from previous simulations or large-scale wind farm measurements, the initial computational investment may be limited to data preprocessing.

5 Discussion

This section examines the implications, limitations, and future directions of the neural network-based wake deficit models presented in this study. The discussion is structured into three main subsections:

The first subsection explores the implications of the findings for wind farm design. It analyzes the potential impact of these models on large-scale wind farm planning, using the N-09 development area in Germany as a case study. The computational efficiency gains offered by the neural network models compared to traditional engineering approaches are evaluated, and the accuracy of power predictions based on the wake deficit estimations is assessed. The second subsection addresses the limitations of the study. It critically examines the constraints of the current model, including its reliance on a single wind turbine type, the 2D nature of the wake predictions, and the architectural limitations of the neural networks. The challenges posed by the training data generation process and the assumption of uniform inflow conditions are also discussed.

The final subsection outlines potential future research directions. Several avenues for enhancing the versatility and accuracy of the models are proposed, including the incorporation of turbine-specific information, expansion to 3D wake calculations, and exploration of alternative neural network architectures such as GNNs. Novel approaches to wind farm representation that could potentially improve computational efficiency and model generalization are also considered. This discussion aims to place the findings in the context of the broader field of wind farm wake modeling, highlight the study's potential impact, and suggest directions for future research in this important area of renewable energy.

5.1 Implications for Wind Farm Design

The results presented in section 4 demonstrate that all models exhibit a relatively high level of accuracy compared to the training data. In its current state, such models can be valuable for estimating wake deficit downwind when planning new wind parks. This is particularly crucial for large tenders like the one depicted in Figure 1.3, which is estimated to have a total nameplate capacity of 5.5 GW with an area of 421 km^2 [17]. The performance of existing wind parks, such as Deutsche Bucht, Veja Mate, and BARD Offshore 1, located south of the N-9 wind development area, is expected to decrease once the new wind farms are commissioned. This underscores the importance of incorporating future neighboring wind parks into economic yield analyses during the planning phase. EnBW He Dreiht, one of Germany's largest offshore wind farms [29], provides a useful reference for estimating the scale of the N-09 development. Using its specifications, we can estimate that approximately 367 turbines ($5500\,\mathrm{MW}/15\,\mathrm{MW} \approx 367$) would be needed in the N-09 area, assuming similar turbine models are suitable.

As discussed in subsection 4.4, the trained neural network models significantly outperform traditional engineering models in terms of computational efficiency, especially as wind farm size increases. Using data from Table 4.5, we can estimate that a single 256×512 resolution TurbOPark simulation for the N-09 area should take:

$$\begin{aligned} &367\,\text{turbines} \times 3.53 \times 10^{-2}\,\text{s/turbine} \\ &= 12.96\,\text{s} \end{aligned} \tag{39}$$

While this timeframe is reasonable for a single simulation, real-world wind farm development often requires multiple simulations. Considering 100 different layouts, 36 inflow wind directions,

and 100 inflow conditions, the total calculation time becomes:

$$\begin{aligned} &367\,\text{turbines} \cdot 3.53 \times 10^{-2}\,\text{s/turbine} \cdot 100\,\text{layouts} \cdot 36\,\text{sectors} \cdot 100\,\text{inflows} \\ &= 1295.5\,\text{hours} \\ &\approx 54\,\text{days} \end{aligned} \tag{40}$$

This comprehensive calculation reveals a much more time-consuming process. However, by utilizing a pre-trained neural network, this computational time can be greatly reduced. As seen from Table 4.6, the mean time per simulation when using the Graphics Processing Unit (GPU) is around 3 ms. Applying this timeframe to the previous simulation scenario yields:

$$\begin{aligned} &3 \times 10^{-3}\,\text{s/simulation} \cdot 100\,\text{layouts} \cdot 36\,\text{sectors} \cdot 100\,\text{inflows} \\ &= 0.3\,\text{hours} \end{aligned} \tag{41}$$

While this is a rough calculation, it highlights the potential of using neural network-based flow prediction methods. Implementing a well-trained and documented model into the industry could allow for a scale of wake deficit calculations that is simply not feasible with current methods, significantly enhancing the efficiency and accuracy of wind farm design and optimization processes.

As shown in Table 4.3, the MAE of the U-Net based models ranges from 4.75×10^{-4} m/s to 1.44×10^{-2} m/s, depending on the training dataset used. The power curve for the Vestas V80 wind turbine employed in this study is illustrated in Figure 3.7b. As discussed in subsubsection 3.3.2, in region 2 between the cut-in wind speed and the rated power wind speed, the power curve's slope is steep, indicating significant power changes with small variations in wind speed. For the Vestas V80 data used in this study, the steepest slope occurs at 9.5 m/s with a gradient of $345\,\frac{\text{kW}}{\text{m/s}}$. Consequently, if one of the U-Net architecture models were used to estimate the wake deficit from an upstream wind farm approaching a Vestas V80 wind turbine, the mean error in power prediction for a single turbine would range between:

$$\begin{aligned} &4.75 \times 10^{-4}\,\text{m/s} \cdot 345\,\frac{\text{kW}}{\text{m/s}} = 0.16\,\text{kW} \\ &\text{and} \\ &1.44 \times 10^{-2}\,\text{m/s} \cdot 345\,\frac{\text{kW}}{\text{m/s}} = 4.97\,\text{kW} \end{aligned} \tag{42}$$

It's important to note that this value depends on the individual wind turbine's position relative to the wake. As observed in subsection 4.3, predictions at the edge of the wake-affected area show higher accuracy. The models generally perform well in applying the ambient wind speed outside the wake. However, within the wake, the discrepancy between the engineering model and the neural network model appears to be greatest. This suggests that if a turbine is located within the wake-affected area, the error in estimated inflow wind speed is likely higher than the overall MAE.

The plots in Appendix C suggest that the neural network models are unbiased, with an equal likelihood of over- and under-predicting the wake deficit, resulting in an average error close to 0 m/s. However, further investigation is needed to conclusively determine whether the models are truly unbiased or if this appearance is coincidental.

5.2 Limitations of the Study

This study's neural network model exhibits several limitations that restrict its immediate applicability in the wind energy industry. Primarily, the model's training is confined to a single wind turbine model, the Vestas V80, which limits its predictive capabilities to wind farms containing this specific turbine or those with very similar thrust curves. While the model's framework could potentially accommodate training on various wind turbines, each would require its own unique training dataset and individual model training, significantly increasing the overall computational demand. The current model's scope is further constrained by its ability to estimate only the 2D wake deficit at the hub height of the Vestas V80 used for training. This limitation poses challenges when predicting wind speeds for downwind turbines with different hub heights or when a more comprehensive understanding of the wind speed variance across the entire rotor disk is required. The model's assumption of extending the hub height wind speed laterally across the rotor disk may introduce errors compared to a more nuanced representation of lateral wind speed variations.

Another technical constraint stems from the neural network architecture, specifically the MLP in the latent space, which currently supports only a 256x512 input and output dimension. While FCNs typically accommodate variable input sizes, the MLP's presence in the latent space imposes this restriction. A potential solution involving input scaling before neural network processing has been identified but remains untested. The wind farm layouts used in the training data generation process, while providing a solid foundation, require further diversification. Future iterations should incorporate a wider array of configurations, including string formations, curved layouts (as observed in Figure 3.2), and randomized grid-based designs with strategically removed turbines to create structural gaps. These enhancements would improve the model's ability to handle more complex and realistic wind farm scenarios.

Lastly, the uniform inflow assumption in the training datasets limits the model's applicability to scenarios with non-uniform inflow conditions. This restriction limits the model from accurately calculating wake deficits in cascading wind park arrangements, where the inflow to subsequent parks in the chain would be inherently non-uniform due to upstream wake effects. Addressing these limitations in future research will significantly enhance the model's versatility and practical applicability in the wind energy sector, paving the way for more comprehensive and accurate wind farm wake deficit predictions.

5.3 Future Research Directions

The accuracy of neural networks depends on multiple factors, including network architecture, proper convergence during training, and the quality and quantity of training data. In this study, where engineering wake deficit models are used to generate training data, the neural networks' accuracy is inherently linked to the accuracy of these underlying models. To surpass the performance of current industry-standard engineering models, future iterations of these neural networks should be trained on higher-fidelity data. This approach could potentially offer both computational efficiency and improved accuracy compared to analytical wake deficit models. However, it's important to note that the immediate implementation of higher-fidelity data is not crucial while the models are still in the development phase. The current approach of comparing neural network outputs with test datasets, as demonstrated in this study, provides a sufficient overview of the models' relative accuracy. This allows for continued refinement of the neural network architectures and training processes before investing in more computationally expensive, high-fidelity training data.

The neural network models presented in subsection 3.4 are designed with the potential to incorporate wind turbine-specific information, such as thrust curves, into the latent space representation. This feature could significantly enhance the model's versatility and industry applicability. However, the effectiveness of this approach in accurately adjusting wake predictions based on individual thrust curves remains to be validated. Implementing such a feature would likely require more extensive training data, potentially increasing the initial computational cost. A logical progression for these models would be to expand from 2D to 3D wake calculations, similar to the approach taken by Asmuth and Korb [7], but on a wind farm scale. This advancement would eliminate the current limitation of single hub height predictions. However, this transition would exponentially increase the complexity of both the training data and the neural network architecture. While this would significantly enhance the model's utility, it would also substantially increase computational demands for training and operation.

The application of Graph Neural Networks (GNNs) in wind farm power output estimation, as demonstrated by Bleeg [13] and Duthé et al. [27], presents an interesting avenue for further research. Extending these models to account for variable thrust curves and non-uniform inflow conditions could potentially enable the prediction of wake effects between neighboring wind parks, expanding beyond their current focus on intra-park effects. Additionally, exploring whether GNNs or similar flexible data structures can be encoded for decoding by CNN-based decoders (like those used in this study) could significantly improve computational efficiency. If flexible data structures prove challenging to encode effectively, an alternative approach could involve representing wind farms as polygons rather than as collections of individual turbines. While this method might sacrifice some fine-grained detail, especially in areas close to the wind parks where individual turbine wakes are most distinct, it could potentially reduce the volume of training data required for long-distance wake field predictions. This approach would model the overall shape of the wind farm and incorporate factors such as average turbine spacing and thrust curves to determine a "denseness" score, offering a more generalized representation of wind farm effects.

These proposed research directions aim to enhance the model's versatility, accuracy, and computational efficiency, addressing current limitations and expanding its potential applications in wind farm wake modeling.

6 Conclusion

This thesis aimed to explore long-distance wind farm flow modeling using Convolutional Neural Networks (CNNs), with a specific focus on the often underestimated wake effects between neighboring wind parks. By utilizing three distinct wake deficit models (Jensen, Bastankhah, and TurbOPark) to generate training datasets for various neural network architectures, the study has demonstrated that CNNs can closely match the accuracy of these models while significantly improving the computational efficiency of wake modeling.

The novel method for random wind park layout generation, coupled with extensive simulations using the Vestas V80 wind turbine model, has provided a robust framework for evaluating the performance of different neural networks across a range of wind park configurations. Among the tested architectures, U-Net and U-Net/MLP consistently outperformed the CAE approaches, delivering the lowest Mean Absolute Errors (MAE) and proving particularly effective in capturing the complex wake interactions modeled by the Jensen and Bastankhah methods. The TurbOPark model, while more challenging due to its slower wake mixing and greater spanwise velocity variations, also benefited from these CNN approaches, albeit with notably higher MAEs.

A key finding of this research is the computational advantage offered by neural networks over traditional engineering models. Unlike conventional methods, where computational time increases linearly with the number of turbines, the CNN-based models maintain a constant execution time of approximately 3 ms, regardless of the wind park size. This efficiency positions CNNs as a promising tool for large-scale wind farm modeling and optimization.

This thesis has laid the groundwork for more accurate and efficient wind farm flow modeling using advanced neural network architectures. The CNN-based approaches developed here have the potential to enhance wind farm design and operation by enabling rapid assessments of different layouts. Moving forward, further refinement of these models—along with the incorporation of additional environmental factors and validation against real-world data—will be crucial in ensuring their practical applicability in the wind energy industry.

Nomenclature

Acronyms

Notation	Description	Page List
ABL	Atmospheric Boundary Layer	9
Adam	Adaptive Moment Estimation	19, 20
AEP	Annual Energy Production	37
ANN	Artificial Neural Network	16
CAE	Convolutional Autoencoder	5, 23, 25, 27, 29, 39, 48–51, 64, 88–90, 94–96
CFD	Computational Fluid Dynamics	2, 3, 32, 37, 42
cGAN	Conditional Generative Adversarial Network	25–27
CNN	Convolutional Neural Network	1, 3, 5, 6, 20–22, 26, 27, 29, 30, 35, 58, 61, 68
DNN	Deep Neural Network	22, 26
EEZ	Exclusive Economic Zone	4, 5
FC	Fully Connected	39
FCN	Fully Convolutional Neural Network	23, 67
GNN	Graph Neural Network	27, 28, 65, 68
GPU	Graphics Processing Unit	66
HPC	High Performance Computing	64
IEC	International Electrotechnical Commission	14
LES	Large Eddy Simulation	2, 26, 64
LiDAR	Light Detection and Ranging	26

Notation	Description	Page List
MAE	Mean Absolute Error	41, 42, 50, 51, 61, 66
MLP	Multilayer Perceptron	5, 24, 25, 27, 29, 48–51, 58, 60, 61, 67, 94–102
MSE	Mean Squared Error	41–43, 46, 48–50, 61
MSP	Marine Spatial Planning	4
RANS	Reynolds-averaged Navier-Stokes	2, 26, 37, 64
ReLU	Rectified Linear Unit	18, 19, 21, 39
RF	Random Forest	26
RMSProp	Root Mean Square Propagation	19, 20
SAR	Synthetic Aperture Radar	2
SCADA	Supervisory Control and Data Acquisition	26, 42
SOS	Sum of Squares	11, 13, 15, 26
SSAA	Supersampling Anti-Aliasing	35, 36
TI	Turbulence Intensity	44
TurbOPark	Turbulence Optimized Park	7, 10, 11, 13, 29, 65
WAsP	Wind Atlas Analysis and Application Program	11
XGBoost	eXtreme Gradient Boosting	26

Symbols

Symbol	Description	Unit	Page List
A	Wake Expansion Calibration Parameter	-	12, 13
C_T	Thrust Coefficient	-	11–13
C	Maximum Normalized Velocity Deficit	-	12, 13
D_0	Rotor Diameter	m	12–14
I_0	Ambient Turbulence Intensity	-	11–15, 58, 61

Symbol	Description	Unit	Page List
I_{add}	Added Turbulence Intensity	-	12–14
I	Turbulence Intensity	-	12, 14, 58
U_0	Initial Wake Velocity	$\mathrm{m\ s^{-1}}$	11, 15, 44, 61
U_w	Wake Velocity	$\mathrm{m\ s^{-1}}$	11
V_{in}	Inflow Wind Speed	$\mathrm{m\,s^{-1}}$	13, 14
α	Parameter for Non-linear Wake Expansion	-	13
β	Parameter for Non-linear Wake Expansion	-	13
δ	Normalized Wind Speed Deficit	-	11, 12, 15
ϵ	Initial Normalized Characteristic Wake Width	-	13
$\mathcal{O}$	Big O Notation	-	63
$\mathcal{U}$	Continuous Uniform Distribution	-	32
σ_w	Characteristic Wake Width	m	12, 13
σ	Gaussian Velocity Deficit Profile Std. Dev.	m	12
a	Axial Induction Factor	-	11, 13, 14
c_1	Empirical Constant	-	13
c_2	Empirical Constant	-	13
k^*	Wake Growth Rate	-	12
k_w	Wake Decay Constant	-	11
r_0	Rotor Radius	m	11
r	Wake Radius	m	11, 12
x	Downwind Distance	m	11–15
y	Spanwise Distance	m	12
z_H	Turbine Hub Height	m	12
z	Vertical Distance	m	12

References

[1] Afshar, Yaser, Bhatnagar, Saakaar, Pan, Shaowu, Duraisamy, Karthik, and Kaushik, Shailendra. "Prediction of Aerodynamic Flow Fields Using Convolutional Neural Networks". en. In: *Computational Mechanics* 64.2 (Aug. 2019). arXiv:1905.13166 [physics], pp. 525–545. ISSN: 0178-7675, 1432-0924. DOI: 10.1007/s00466-019-01740-0. URL: http://arxiv.org/abs/1905.13166 (visited on 03/20/2024).

[2] Ainslie, J. F. "Calculating the flowfield in the wake of wind turbines". In: *Journal of Wind Engineering and Industrial Aerodynamics* 27.1 (Jan. 1988), pp. 213–224. ISSN: 0167-6105. DOI: 10.1016/0167-6105(88)90037-2. URL: https://www.sciencedirect.com/science/article/pii/0167610588900372 (visited on 01/17/2024).

[3] Ajit, Arohan, Acharya, Koustav, and Samanta, Abhishek. "A Review of Convolutional Neural Networks". In: *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*. Vellore, India: IEEE, Feb. 2020, pp. 1–5. ISBN: 978-1-72814-142-8. DOI: 10.1109/ic-ETITE47903.2020.049. URL: https://ieeexplore.ieee.org/document/9077735/ (visited on 07/11/2024).

[4] Ali, Naseem, Calaf, Marc, and Cal, Raúl Bayoán. "Cluster-based probabilistic structure dynamical model of wind turbine wake". In: *Journal of Turbulence* 22.8 (Aug. 2021). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/14685248.2021.1925125, pp. 497–516. ISSN: null. DOI: 10.1080/14685248.2021.1925125. URL: https://doi.org/10.1080/14685248.2021.1925125 (visited on 01/25/2024).

[5] Anagnostopoulos, Sokratis J., Bauer, Jens, Clare, Mariana C. A., and Piggott, Matthew D. "Accelerated wind farm yaw and layout optimisation with multi-fidelity deep transfer learning wake models". In: *Renewable Energy* 218 (Dec. 2023), p. 119293. ISSN: 0960-1481. DOI: 10.1016/j.renene.2023.119293. URL: https://www.sciencedirect.com/science/article/pii/S0960148123012089 (visited on 01/24/2024).

[6] Andersen, S. J., Sørensen, J. N., Ivanell, S., and Mikkelsen, R. F. "Comparison of Engineering Wake Models with CFD Simulations". en. In: *Journal of Physics: Conference Series* 524.1 (June 2014), p. 012161. ISSN: 1742-6596. DOI: 10.1088/1742-6596/524/1/012161. URL: https://dx.doi.org/10.1088/1742-6596/524/1/012161 (visited on 01/15/2024).

[7] Asmuth, Henrik and Korb, Henry. "WakeNet 0.1 - A Simple Three-dimensional Wake Model Based on Convolutional Neural Networks". en. In: *Journal of Physics: Conference Series* 2265.2 (May 2022). Publisher: IOP Publishing, p. 022066. ISSN: 1742-6596. DOI: 10.1088/1742-6596/2265/2/022066. URL: https://dx.doi.org/10.1088/1742-6596/2265/2/022066 (visited on 01/24/2024).

[8] Barthelmie, R.J., Folkerts, L., Larsen, Gunner Chr., Rados, K., Pryor, S.C., Frandsen, Sten Tronæs, Lange, B., and Schepers, G. "Comparison of wake model simulations with offshore wind turbine wake profiles measured by sodar". In: *Journal of Atmospheric and Oceanic Technology* 23 (2006), pp. 888–901. ISSN: 0739-0572. DOI: 10.1175/JTECH1886.1.

[9] Bastankhah, Majid and Porté-Agel, Fernando. "A new analytical model for wind-turbine wakes". In: *Renewable Energy*. Special issue on aerodynamics of offshore wind energy systems and wakes 70 (Oct. 2014), pp. 116–123. ISSN: 0960-1481. DOI: 10.1016/j.renene.2014.01.002. URL: https://www.sciencedirect.com/science/article/pii/S0960148114000317 (visited on 12/11/2023).

[10] Bastankhah, Majid, Welch, Bridget L., Martínez-Tossas, Luis A., King, Jennifer, and Fleming, Paul. "Analytical solution for the cumulative wake of wind turbines in wind farms". en. In: *Journal of Fluid Mechanics* 911 (Mar. 2021). Publisher: Cambridge University Press, A53. ISSN: 0022-1120, 1469-7645. DOI: 10.1017/jfm.2020.1037. URL: https://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/analytical-solution-for-the-cumulative-wake-of-wind-turbines-in-wind-farms/014B76C0605168E6648AA9B3A4D4EDFE (visited on 01/14/2024).

[11] Bertolani, Luca. "Fourier Neural Operator-Driven Multi-Fidelity Surrogate of RANS-based Wind Farm Wake Simulations". en. PhD thesis. Lyngby, Denmark: Technical Univeristy of Denmark, June 2024.

[12] Binsbergen, Diederik van, Daems, Pieter-Jan, Verstraeten, Timothy, R. Nejad, Amir, and Helsen, Jan. "Performance comparison of analytical wake models calibrated on a large offshore wind cluster". In: *Journal of Physics: Conference Series* 2767 (June 2024), p. 092059. DOI: 10.1088/1742-6596/2767/9/092059.

[13] Bleeg, James. "A Graph Neural Network Surrogate Model for the Prediction of Turbine Interaction Loss". en. In: *Journal of Physics: Conference Series* 1618.6 (Sept. 2020). Publisher: IOP Publishing, p. 062054. ISSN: 1742-6596. DOI: 10.1088/1742-6596/1618/6/062054. URL: https://dx.doi.org/10.1088/1742-6596/1618/6/062054 (visited on 11/28/2023).

[14] Blondel, Frédéric and Cathelain, Marie. "An alternative form of the super-Gaussian wind turbine wake model". English. In: *Wind Energy Science* 5.3 (Sept. 2020). Publisher: Copernicus GmbH, pp. 1225–1236. ISSN: 2366-7443. DOI: 10.5194/wes-5-1225-2020. URL: https://wes.copernicus.org/articles/5/1225/2020/ (visited on 12/11/2023).

[15] Bokharaie, Vahid S., Bauweraerts, Pieter, and Meyers, Johan. "Wind-farm layout optimisation using a hybrid Jensen–LES approach". English. In: *Wind Energy Science* 1.2 (Dec. 2016). Publisher: Copernicus GmbH, pp. 311–325. ISSN: 2366-7443. DOI: 10.5194/wes-1-311-2016. URL: https://wes.copernicus.org/articles/1/311/2016/ (visited on 01/24/2024).

[16] Bossuyt, Ottelien. "Modelling and validation of wind turbine wake superposition: Using wind farm data". en. PhD thesis. 2018. URL: https://repository.tudelft.nl/islandora/object/uuid%3A8a9ff2f3-d4d4-470b-a5b8-11a08e31eb16 (visited on 01/12/2024).

[17] Bundesamt für Seeschifffahrt und Hydrographie. *Data Hub Preliminary Investigation of Sites (PINTA)*. URL: https://pinta.bsh.de/?lang=en (visited on 07/07/2024).

[18] Cai, Zemin, Fan, Zhengyuan, and Liu, Tianshu. *Efficient aerodynamic coefficients prediction with a long sequence neural network*. arXiv:2403.14979 [physics]. Mar. 2024. DOI: 10.48550/arXiv.2403.14979. URL: http://arxiv.org/abs/2403.14979 (visited on 07/02/2024).

[19] Cañadillas, Beatriz, Foreman, Richard, Steinfeld, Gerald, and Robinson, Nick. "Cumulative Interactions between the Global Blockage and Wake Effects as Observed by an Engineering Model and Large-Eddy Simulations". en. In: *Energies* 16.7 (Jan. 2023). Number: 7 Publisher: Multidisciplinary Digital Publishing Institute, p. 2949. ISSN: 1996-1073. DOI: 10.3390/en16072949. URL: https://www.mdpi.com/1996-1073/16/7/2949 (visited on 01/17/2024).

[20] Chen, Hai, He, Lei, Qian, Weiqi, and Wang, Song. "Multiple Aerodynamic Coefficient Prediction of Airfoils Using a Convolutional Neural Network". In: *Symmetry* 12 (Apr. 2020), p. 544. DOI: 10.3390/sym12040544.

[21] Chen, Junfeng, Viquerat, Jonathan, and Hachem, Elie. "U-net architectures for fast prediction in fluid mechanics". Dec. 2019. URL: https://hal.science/hal-02401465 (visited on 02/15/2024).

[22] Continuum Labs. *Learning Rate Scheduler | Continuum Labs*. en. Apr. 2024. URL: https://training.continuumlabs.ai/training/the-fine-tuning-process/hyperparameters/learning-rate-scheduler (visited on 07/29/2024).

[23] Crespo, A. and Hernández, J. "Turbulence characteristics in wind-turbine wakes". In: *Journal of Wind Engineering and Industrial Aerodynamics* 61.1 (June 1996), pp. 71–85. ISSN: 0167-6105. DOI: 10.1016/0167-6105(95)00033-X. URL: https://www.sciencedirect.com/science/article/pii/016761059500033X (visited on 07/14/2024).

[24] DataCamp. *Multilayer Perceptrons in Machine Learning: A Comprehensive Guide*. en. July 2024. URL: https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning (visited on 07/27/2024).

[25] Dogger Bank Wind Farm. *About The Dogger Bank Wind Farm Projects*. 2024. URL: https://doggerbank.com/about/ (visited on 07/30/2024).

[26] Donglin, Chen, Gao, Xiang, Xu, Chuanfu, Chen, Shizhao, Fang, Jianbin, and Wang, Zhenghua. *FlowGAN: A Conditional Generative Adversarial Network for Flow Prediction in Various Conditions*. Sept. 2020. DOI: 10.1109/ICTAI50040.2020.00057.

[27] Duthé, Gregory, Santos, Francisco de Nolasco, Abdallah, Imad, Réthore, Pierre-Élouan, Weijtjens, Wout, Chatzi, Eleni, and Devriendt, Christof. "Local flow and loads estimation on wake-affected wind turbines using graph neural networks and PyWake". In: *Journal of Physics: Conference Series* 2505.1 (May 2023). Publisher: IOP Publishing, p. 012014. DOI: 10.1088/1742-6596/2505/1/012014. URL: https://dx.doi.org/10.1088/1742-6596/2505/1/012014.

[28] Ember. *Electricity Data Explorer | Open Source Global Electricity Data*. en-US. 2023. URL: https://ember-climate.org/data/data-tools/data-explorer/ (visited on 07/28/2024).

[29] EnBW. *EnBW wind farm He Dreiht*. en. URL: https://www.enbw.com/company/topics/wind-power/offshore-wind-farm-he-dreiht/ (visited on 07/28/2024).

[30] European Marine Observation and Data Network. *EMODnet Map Viewer*. URL: https://emodnet.ec.europa.eu/geoviewer/ (visited on 02/28/2024).

[31] Fischereit, Jana, Schaldemose Hansen, Kurt, Larsén, Xiaoli Guo, Laan, Maarten Paul van der, Réthoré, Pierre-Elouan, and Murcia Leon, Juan Pablo. "Comparing and validating intra-farm and farm-to-farm wakes across different mesoscale and high-resolution wake models". English. In: *Wind Energy Science* 7.3 (May 2022). Publisher: Copernicus GmbH, pp. 1069–1091. ISSN: 2366-7443. DOI: 10.5194/wes-7-1069-2022. URL: https://wes.copernicus.org/articles/7/1069/2022/ (visited on 01/09/2024).

[32] Frandsen, Sten. "On the wind speed reduction in the center of large clusters of wind turbines". In: *Journal of Wind Engineering and Industrial Aerodynamics* 39.1 (Jan. 1992), pp. 251–265. ISSN: 0167-6105. DOI: 10.1016/0167-6105(92)90551-K. URL: https://www.sciencedirect.com/science/article/pii/016761059290551K (visited on 07/30/2024).

[33] Frandsen, Sten, Barthelmie, Rebecca, Pryor, Sara, Rathmann, Ole, Larsen, Søren, Højstrup, Jørgen, and Thøgersen, Morten. "Analytical modelling of wind speed deficit in large offshore wind farms". en. In: *Wind Energy* 9.1-2 (2006). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/we.189, pp. 39–53. ISSN: 1099-1824. DOI: 10.1002/we.189. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/we.189 (visited on 01/08/2024).

[34] Frandsen, Sten Tronæs. "Turbulence and turbulence-generated structural loading in wind turbine clusters". ISBN: 9788755034587 Publication Title: Turbulence and turbulence-generated structural loading in wind turbine clusters. Doctoral thesis. 2007.

[35] Fukami, Kai, Fukagata, Koji, and Taira, Kunihiko. *Assessment of supervised machine learning methods for fluid flows*. Jan. 2020.

[36] Fukushima, Kunihiko. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". en. In: *Biological Cybernetics* 36.4 (Apr. 1980), pp. 193–202. ISSN: 0340-1200, 1432-0770. DOI: 10.1007/BF00344251. URL: http://link.springer.com/10.1007/BF00344251 (visited on 07/14/2024).

[37] Global Wind Energy Council. *Global Wind Report 2023*. en-US. Tech. rep. Feb. 2023. URL: https://gwec.net/globalwindreport2023/ (visited on 07/28/2024).

[38] Göçmen, Tuhfe, Laan, Paul Van Der, Réthoré, Pierre-Elouan, Diaz, Alfredo Peña, Larsen, Gunner Chr., and Ott, Søren. "Wind turbine wake models developed at the technical university of Denmark: A review". en. In: *Renewable and Sustainable Energy Reviews* 60 (July 2016), pp. 752–769. ISSN: 13640321. DOI: 10.1016/j.rser.2016.01.113. URL: https://linkinghub.elsevier.com/retrieve/pii/S136403211600143X (visited on 07/30/2024).

[39] Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep learning*. MIT Press, 2016.

[40] Guo, Xiaoxiao, Li, Wei, and Iorio, Francesco. "Convolutional Neural Networks for Steady Flow Approximation". en. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco California USA: ACM, Aug. 2016, pp. 481–490. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939738. URL: https://dl.acm.org/doi/10.1145/2939672.2939738 (visited on 01/24/2024).

[41] Hamilton, Nicholas, Viggiano, Bianca, Calaf, Marc, Tutkun, Murat, and Cal, Raúl Bayoán. "A generalized framework for reduced-order modeling of a wind turbine wake". en. In: *Wind Energy* 21.6 (2018). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/we.2167, pp. 373–390. ISSN: 1099-1824. DOI: 10.1002/we.2167. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/we.2167 (visited on 01/25/2024).

[42] Harrison-Atlas, Dylan, Glaws, Andrew, King, Ryan N., and Lantz, Eric. "Artificial intelligence-aided wind plant optimization for nationwide evaluation of land use and economic benefits of wake steering". en. In: *Nature Energy* 9.6 (June 2024). Publisher: Nature Publishing Group, pp. 735–749. ISSN: 2058-7546. DOI: 10.1038/s41560-024-01516-8. URL: https://www.nature.com/articles/s41560-024-01516-8 (visited on 07/08/2024).

[43] Hasager, Charlotte, Rasmussen, Leif, Peña, Alfredo, Jensen, Leo, and Réthoré, Pierre-Elouan. "Wind Farm Wake: The Horns Rev Photo Case". en. In: *Energies* 6.2 (Feb. 2013), pp. 696–716. ISSN: 1996-1073. DOI: 10.3390/en6020696. URL: https://www.mdpi.com/1996-1073/6/2/696 (visited on 07/30/2024).

[44] Hasager, Charlotte Bay, Vincent, Pauline, Badger, Jake, Badger, Merete, Di Bella, Alessandro, Peña, Alfredo, Husson, Romain, and Volker, Patrick J. H. "Using Satellite SAR to Characterize the Wind Flow around Offshore Wind Farms". en. In: *Energies* 8.6 (June 2015). Number: 6 Publisher: Multidisciplinary Digital Publishing Institute, pp. 5413–5439. ISSN: 1996-1073. DOI: 10.3390/en8065413. URL: https://www.mdpi.com/1996-1073/8/6/5413 (visited on 07/28/2024).

[45] International Renewable Energy Agency (IRENA). *Renewable capacity statistics 2023*. en. Tech. rep. Mar. 2023. URL: https://www.irena.org/Publications/2023/Mar/Renewable-capacity-statistics-2023 (visited on 07/28/2024).

[46] Jensen, Niels Otto. "A note on wind generator interaction". en. In: *Risø National Laboratory* (1983), p. 16. ISSN: 87-550-0971-9. URL: https://orbit.dtu.dk/en/publications/a-note-on-wind-generator-interaction.

[47] Katic, I., Højstrup, J., and Jensen, N.O. "A Simple Model for Cluster Efficiency: European Wind Energy Association Conference and Exhibition". In: *EWEC'86. Proceedings. Vol. 1* (1987). Ed. by W. Palz and E. Sesto. Place: Rome Publisher: A. Raguzzi, pp. 407–410.

[48] Kingma, Diederik P. and Ba, Jimmy. *Adam: A Method for Stochastic Optimization*. en. arXiv:1412.6980 [cs]. Jan. 2017. URL: http://arxiv.org/abs/1412.6980 (visited on 07/11/2024).

[49] Kipf, Thomas N. and Welling, Max. *Semi-Supervised Classification with Graph Convolutional Networks*. en. arXiv:1609.02907 [cs, stat]. Feb. 2017. URL: http://arxiv.org/abs/1609.02907 (visited on 07/31/2024).

[50] Krutova, Maria, Paskyabi, Mostafa Bakhoday, Nielsen, Finn Gunnar, and Reuder, Joachim. "Evaluation of Gaussian wake models under different atmospheric stability conditions: Comparison with large eddy simulation results". en. In: *Journal of Physics: Conference Series* 1669.1 (Oct. 2020), p. 012016. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/1669/1/012016. URL: https://iopscience.iop.org/article/10.1088/1742-6596/1669/1/012016 (visited on 07/30/2024).

[51] Larsen, Gunner Chr. *A Simple Wake Calculation Procedure*. Report 87-550-1484-4. Publication Title: A Simple Wake Calculation Procedure. Roskilde: Risø National Laboratory, 1988.

[52] LeCun, Yann, Bottou, Leon, Bengio, Yoshua, and Ha, Patrick. "Gradient-Based Learning Applied to Document Recognition". en. In: (1998).

[53] Leer, Michael and Kempf, Andreas. "Fast Flow Field Estimation for Various Applications with A Universally Applicable Machine Learning Concept". en. In: *Flow, Turbulence and Combustion* 107.1 (June 2021), pp. 175–200. ISSN: 1573-1987. DOI: 10.1007/s10494-020-00234-x. URL: https://doi.org/10.1007/s10494-020-00234-x (visited on 01/25/2024).

[54] Lorena, Ana C., Maciel, Aron I., Miranda, Péricles B. C. de, Costa, Ivan G., and Prudêncio, Ricardo B. C. "Data complexity meta-features for regression problems". en. In: *Machine Learning* 107.1 (Jan. 2018), pp. 209–246. ISSN: 1573-0565. DOI: 10.1007/s10994-017-5681-1. URL: https://doi.org/10.1007/s10994-017-5681-1 (visited on 07/26/2024).

[55] Magnusson, M. and Smedman, A. -S. "Air flow behind wind turbines". In: *Journal of Wind Engineering and Industrial Aerodynamics* 80.1 (Mar. 1999), pp. 169–189. ISSN: 0167-6105. DOI: 10.1016/S0167-6105(98)00126-3. URL: https://www.sciencedirect.com/science/article/pii/S0167610598001263 (visited on 07/30/2024).

[56] Memija, Adnan. *Germany Kicks Off 5.5 GW Offshore Wind Tender*. en-US. Feb. 2024. URL: https://www.offshorewind.biz/2024/02/28/germany-kicks-off-5-5-gw-offshore-wind-tender/ (visited on 07/07/2024).

[57] Minaee, Shervin, Boykov, Yuri, Porikli, Fatih, Plaza, Antonio, Kehtarnavaz, Nasser, and Terzopoulos, Demetri. *Image Segmentation Using Deep Learning: A Survey*. en. arXiv:2001.05566 [cs]. Nov. 2020. URL: http://arxiv.org/abs/2001.05566 (visited on 07/17/2024).

[58] Nai-Zhi, Guo, Ming-Ming, Zhang, and Bo, Li. "A data-driven analytical model for wind turbine wakes using machine learning method". In: *Energy Conversion and Management* 252 (Jan. 2022), p. 115130. ISSN: 0196-8904. DOI: 10.1016/j.enconman.2021.115130. URL: https://www.sciencedirect.com/science/article/pii/S0196890421013066 (visited on 01/08/2024).

[59] Nakhchi, M. E., Win Naung, S., and Rahmati, M. "Wake and power prediction of horizontal-axis wind farm under yaw-controlled conditions with machine learning". In: *Energy Conversion and Management* 296 (Nov. 2023), p. 117708. ISSN: 0196-8904. DOI: 10.1016/j.enconman.2023.117708. URL: https://www.sciencedirect.com/science/article/pii/S0196890423010543 (visited on 01/16/2024).

[60] Nygaard, Nicolai Gayle. "Wakes in very large wind farms and the effect of neighbouring wind farms". en. In: *Journal of Physics: Conference Series* 524.1 (June 2014), p. 012162. ISSN: 1742-6596. DOI: 10.1088/1742-6596/524/1/012162. URL: https://dx.doi.org/10.1088/1742-6596/524/1/012162 (visited on 01/05/2024).

[61] Nygaard, Nicolai Gayle. *OrstedRD/TurbOPark*. original-date: 2022-01-05T11:02:40Z. Sept. 2023. URL: https://github.com/OrstedRD/TurbOPark (visited on 12/11/2023).

[62] Nygaard, Nicolai Gayle and Hansen, Sidse Damgaard. "Wake effects between two neighbouring wind farms". en. In: *Journal of Physics: Conference Series* 753.3 (Sept. 2016). Publisher: IOP Publishing, p. 032020. ISSN: 1742-6596. DOI: 10.1088/1742-6596/753/3/032020. URL: https://dx.doi.org/10.1088/1742-6596/753/3/032020 (visited on 01/05/2024).

[63] Nygaard, Nicolai Gayle and Newcombe, Alexander Christian. "Wake behind an offshore wind farm observed with dual-Doppler radars". en. In: *Journal of Physics: Conference Series* 1037.7 (June 2018). Publisher: IOP Publishing, p. 072008. ISSN: 1742-6596. DOI: 10.1088/1742-6596/1037/7/072008. URL: https://dx.doi.org/10.1088/1742-6596/1037/7/072008 (visited on 01/10/2024).

[64] Nygaard, Nicolai Gayle, Steen, Søren Trads, Poulsen, Lina, and Pedersen, Jesper Grønnegaard. "Modelling cluster wakes and wind farm blockage". en. In: *Journal of Physics: Conference Series* 1618.6 (Sept. 2020). Publisher: IOP Publishing, p. 062072. ISSN: 1742-6596. DOI: 10.1088/1742-6596/1618/6/062072. URL: https://dx.doi.org/10.1088/1742-6596/1618/6/062072 (visited on 12/11/2023).

[65] Oppermann, Artem. *Optimization in Deep Learning: AdaGrad, RMSProp, ADAM*. en-US. Oct. 2021. URL: https://artemoppermann.com/optimization-in-deep-learning-adagrad-rmsprop-adam/ (visited on 07/11/2024).

[66] Ørsted. *Ørsted presents update on its long-term financial targets*. en. Oct. 2019. URL: https://orsted.com/en/company-announcement-list/2019/10/1937002 (visited on 07/30/2024).

[67] Ørsted. *Our offshore wind farms*. en. 2024. URL: https://orsted.co.uk/energy-solutions/offshore-wind/our-wind-farms (visited on 07/30/2024).

[68] Paszke, Adam, Gross, Sam, Massa, Francisco, Lerer, Adam, Bradbury, James, Chanan, Gregory, Killeen, Trevor, Lin, Zeming, Gimelshein, Natalia, Antiga, Luca, Desmaison, Alban, Kopf, Andreas, Yang, Edward, DeVito, Zachary, Raison, Martin, Tejani, Alykhan, Chilamkurthy, Sasank, Steiner, Benoit, Fang, Lu, Bai, Junjie, and Chintala, Soumith. "PyTorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[69] Pedersen, J G, Svensson, E, Poulsen, L, and Nygaard, N G. "Turbulence Optimized Park model with Gaussian wake profile". en. In: *Journal of Physics: Conference Series* 2265.2 (May 2022), p. 022063. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/2265/2/022063. URL: https://iopscience.iop.org/article/10.1088/1742-6596/2265/2/022063 (visited on 07/28/2024).

[70] Pedersen, Mads M., Forsting, Alexander Meyer, Laan, Paul van der, Riva, Riccardo, Risco, Javier Criado, Friis-Møller, Mikkel, Quick, Julian, Christiansen, Jens Peter Schøler, Rodrigues, Rafael Valotta, Olsen, Bjarke Tobias, and Réthoré, Pierre-Elouan. "PyWake 2.5.0: An open-source wind farm simulation tool". In: (Feb. 2023). Publisher: DTU Wind, Technical University of Denmark. URL: https://gitlab.windenergy.dtu.dk/TOPFARM/PyWake.

[71] Platis, Andreas, Bange, Jens, Bärfuss, Konrad, Cañadillas, Beatriz, Hundhausen, Marie, Djath, Bughsin, Lampert, Astrid, Schulz-Stellenfleth, Johannes, Siedersleben, Simon, Neumann, Thomas, and Emeis, Stefan. "Long-range modifications of the wind field by offshore wind parks – results of the project WIPAFF". en. In: *Meteorologische Zeitschrift* 29.5 (Nov. 2020), pp. 355–376. ISSN: 0941-2948. DOI: 10.1127/metz/2020/1023. URL: http://www.schweizerbart.de/papers/metz/detail/29/93565/Long_range_modifications_of_the_wind_field_by_offs?af=crossref (visited on 01/09/2024).

[72] Portal-Porras, Koldo, Fernandez-Gamiz, Unai, Zulueta, Ekaitz, Ballesteros-Coll, Alejandro, and Zulueta, Asier. "CNN-based flow control device modelling on aerodynamic airfoils". In: *Scientific Reports* 12 (May 2022). DOI: 10.1038/s41598-022-12157-w.

[73] Premanand, S. *A Comprehensive Guide to UNET Architecture | Mastering Image Segmentation*. en. Aug. 2023. URL: https://www.analyticsvidhya.com/blog/2023/08/unet-architecture-mastering-image-segmentation/ (visited on 07/27/2024).

[74] PyWake development team. "Automated validation report for PyWake". en. In: *DTU Wind Energy* (2024). URL: https://topfarm.pages.windenergy.dtu.dk/PyWake/_images/validation_report.pdf.

[75] Renganathan, Ashwin, Maulik, Romit, Letizia, Stefano, and Iungo, Giacomo Valerio. "Data-driven wind turbine wake modeling via probabilistic machine learning". en. In: *Neural Computing and Applications* 34.8 (Apr. 2022), pp. 6171–6186. ISSN: 1433-3058. DOI: 10.1007/s00521-021-06799-6. URL: https://doi.org/10.1007/s00521-021-06799-6 (visited on 01/05/2024).

[76] Ribeiro, Mateus Dias, Rehman, Abdul, Ahmed, Sheraz, and Dengel, Andreas. *DeepCFD: Efficient Steady-State Laminar Flow Approximation with Deep Convolutional Neural Networks*. en. arXiv:2004.08826 [physics]. Nov. 2021. URL: http://arxiv.org/abs/2004.08826 (visited on 04/24/2024).

[77] Ronneberger, Olaf, Fischer, Philipp, and Brox, Thomas. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. arXiv:1505.04597 [cs]. May 2015. DOI: 10.48550/arXiv.1505.04597. URL: http://arxiv.org/abs/1505.04597 (visited on 07/18/2024).

[78] Sanderse, Benjamin. "Aerodynamics of wind turbine wakes Literature review". In: 2009. URL: https://www.semanticscholar.org/paper/Aerodynamics-of-wind-turbine-wakes-Literature-Sanderse/25d768ff87346b263be5a9523264de9ab00f606c (visited on 07/01/2024).

[79] Schneemann, Jörge, Rott, Andreas, Dörenkämper, Martin, Steinfeld, Gerald, and Kühn, Martin. "Cluster wakes impact on a far-distant offshore wind farm's power". English. In: *Wind Energy Science* 5.1 (Jan. 2020). Publisher: Copernicus GmbH, pp. 29–49. ISSN: 2366-7443. DOI: 10.5194/wes-5-29-2020. URL: https://wes.copernicus.org/articles/5/29/2020/ (visited on 01/05/2024).

[80] Sessarego, Matias, Shen, Wen Zhong, Van der Laan, Maarten Paul, Hansen, Kurt Schaldemose, and Zhu, Wei Jun. "CFD Simulations of Flows in a Wind Farm in Complex Terrain and Comparisons to Measurements". en. In: *Applied Sciences* 8.5 (May 2018). Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, p. 788. ISSN: 2076-3417. DOI: 10.3390/app8050788. URL: https://www.mdpi.com/2076-3417/8/5/788 (visited on 07/28/2024).

[81] Siddique, Nahian, Paheding, Sidike, Elkin, Colin P., and Devabhaktuni, Vijay. "U-Net and Its Variants for Medical Image Segmentation: A Review of Theory and Applications". In: *IEEE Access* 9 (2021). Conference Name: IEEE Access, pp. 82031–82057. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3086020. URL: https://ieeexplore.ieee.org/abstract/document/9446143 (visited on 07/18/2024).

[82] Sofia Offshore Wind Farm. *Project*. en. 2024. URL: https://sofiawindfarm.com/project/ (visited on 07/30/2024).

[83] Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (June 2014), pp. 1929–1958.

[84] Stieren, Anja and Stevens, Richard J. A. M. "Evaluating wind farm wakes in large eddy simulations and engineering models". en. In: *Journal of Physics: Conference Series* 1934.1 (May 2021). Publisher: IOP Publishing, p. 012018. ISSN: 1742-6596. DOI: 10.1088/1742-6596/1934/1/012018. URL: https://dx.doi.org/10.1088/1742-6596/1934/1/012018 (visited on 07/28/2024).

[85] Thomsen, Kenneth and Sørensen, Poul. "Fatigue loads for wind turbines operating in wakes". In: *Journal of Wind Engineering and Industrial Aerodynamics* 80.1 (Mar. 1999), pp. 121–136. ISSN: 0167-6105. DOI: 10.1016/S0167-6105(98)00194-9. URL: https://www.sciencedirect.com/science/article/pii/S0167610598001949 (visited on 07/30/2024).

[86] Ti, Zilong, Deng, Xiao Wei, and Yang, Hongxing. "Wake modeling of wind turbines using machine learning". In: *Applied Energy* 257 (Jan. 2020), p. 114025. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2019.114025. URL: https://www.sciencedirect.com/science/article/pii/S030626191931712X (visited on 01/08/2024).

[87] Wilson, Brett, Wakes, Sarah, and Mayo, Michael. "Surrogate modeling a computational fluid dynamics-based wind turbine wake simulation using machine learning". In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. Nov. 2017, pp. 1–8. DOI: 10.1109/SSCI.2017.8280844. URL: https://ieeexplore.ieee.org/document/8280844 (visited on 01/17/2024).

[88] *Wind energy generation systems – Part 1: Design requirements*. en. Tech. rep. 61400-1. Geneva: IEC, 2005.

[89] *Wind energy generation systems – Part 1: Design requirements*. en. Tech. rep. 61400-1. Geneva: IEC, 2019.

[90] Wu, Yu-Ting and Porté-Agel, Fernando. "Modeling turbine wakes and power losses within a wind farm using LES: An application to the Horns Rev offshore wind farm". In: *Renewable Energy* 75 (Mar. 2015), pp. 945–955. ISSN: 0960-1481. DOI: 10.1016/j.renene.2014.06.019. URL: https://www.sciencedirect.com/science/article/pii/S0960148114003590 (visited on 01/09/2024).

[91] Xu, Jiayang and Duraisamy, Karthik. "Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics". In: *Computer Methods in Applied Mechanics and Engineering* 372 (Dec. 2020), p. 113379. ISSN: 0045-7825. DOI: 10.1016/j.cma.2020.113379. URL: https://www.sciencedirect.com/science/article/pii/S0045782520305648 (visited on 07/27/2024).

[92] Zhang, Jincheng and Zhao, Xiaowei. "A novel dynamic wind farm wake model based on deep learning". In: *Applied Energy* 277 (Nov. 2020), p. 115552. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2020.115552. URL: https://www.sciencedirect.com/science/article/pii/S0306261920310643 (visited on 01/14/2024).

[93] Zhang, Jincheng and Zhao, Xiaowei. "Wind farm wake modeling based on deep convolutional conditional generative adversarial network". In: *Energy* 238 (Jan. 2022), p. 121747. ISSN: 0360-5442. DOI: 10.1016/j.energy.2021.121747. URL: https://www.sciencedirect.com/science/article/pii/S0360544221019952 (visited on 01/22/2024).

[94] Zhang, Yao, Sung, Woong Je, and Mavris, Dimitri N. "Application of Convolutional Neural Network to Predict Airfoil Lift Coefficient". en. In: *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference* (Jan. 2018). Conference Name: 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference ISBN: 9781624105326 Place: Kissimmee, Florida Publisher: American Institute of Aeronautics and Astronautics. DOI: 10.2514/6.2018-1903. URL: https://arc.aiaa.org/doi/10.2514/6.2018-1903 (visited on 04/02/2024).

[95] Zhang, Zexia, Santoni, Christian, Herges, Thomas, Sotiropoulos, Fotis, and Khosronejad, Ali. "Time-Averaged Wind Turbine Wake Flow Field Prediction Using Autoencoder Convolutional Neural Networks". en. In: *Energies* 15.1 (Jan. 2022). Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, p. 41. ISSN: 1996-1073. DOI: 10.3390/en15010041. URL: https://www.mdpi.com/1996-1073/15/1/41 (visited on 01/17/2024).

A Graph Neural Networks: Foundations and Applications

Throughout this thesis, I experimented with various neural network architectures. Among them, Graph Neural Networks (GNNs) stood out for their effectiveness in capturing the structure and interactions between wind turbines in wind farms. This is primarily due to their ability to operate on graph-structured data, unlike traditional neural networks that process data in Euclidean space (e.g., grids for images or sequences for text). GNNs are a class of deep learning models specifically designed to process data represented as graphs. A graph $G = (V, E)$ consists of a set of nodes (or vertices) V and a set of edges E. Each node $v \in V$ can have associated feature vectors, and each edge $(u, v) \in E$ can carry information about the relationship between nodes u and v. In the context of the neural network presented in this thesis, the input data is represented as a graph, where each node is associated with specific features.

A fundamental component of many GNN architectures is the Graph Convolutional Network (GCN). GCNs extend the concept of convolution from grid-like data structures, as seen in Convolutional Neural Networks (CNNs), to graph-structured data. The core idea behind GCNs is to update node representations by aggregating information from neighboring nodes. This process can be mathematically formulated as follows [49]:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right) \tag{43}$$

In this equation, $H^{(l)}$ is the node feature matrix at layer l, $\tilde{A} = A + I$ represents the adjacency matrix of the graph with added self-loops (where A is the original adjacency matrix and I is the identity matrix), $\tilde{D}$ is the diagonal degree matrix of $\tilde{A}$ with $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $W^{(l)}$ is the learnable weight matrix for layer l, and σ is a non-linear activation function, such as ReLU. The core operation in GNNs is often referred to as "message passing." This process typically involves three steps: message generation, where each source node generates a message to be sent to its neighboring target nodes; message aggregation, where each node aggregates the messages it receives from its neighbors; and node update, where node representations are updated based on the aggregated messages.

In the architecture presented in Table A.1, the GCNConv layers in the encoder implement this message-passing mechanism, learning node embeddings that capture both local and global graph structures. For graph-level tasks, it is necessary to obtain a fixed-size representation of the entire graph. In this architecture, a global mean pooling (GMP) operation is employed:

$$x = \text{global\_mean\_pool}(x, \text{batch}) \tag{44}$$

This operation averages the node features across each graph in the batch, resulting in a graph-level representation. A significant property of GNNs, including the architecture presented here, is their ability to perform inductive learning. This allows the model to generalize to unseen graphs, provided they have the same node feature dimensions. The GCN layers learn to operate on local graph structures, making the model adaptable to graphs of varying sizes and structures.

A.1 Graph Convolutional Network / Convolutional Neural Network Architecture

As an experiment, I combined a Graph Neural Network (GNN) with a Convolutional Neural Network (CNN) decoder, similar to those used in Convolutional Autoencoder (CAE) and U-Net architectures. The GNN is used to extract features for all nodes in the graph. After applying

a global mean pooling operation to the node features, the resulting latent space feature map is fed into a CNN, which then decodes this feature map into a tensor representing the wake deficit of the wind farm. Table A.1 provides an overview of the neural network structure used to achieve this:

Table A.1: GCN-CNN Architecture.

Layer Group	**Layers**	**In Channels**	**Out Channels**	**Kernel**	**Stride**	**Padding**	**Out Padding**
Encoder	GCNConv 1	$in_{channels}$	16	–	–	–	–
	GCNConv 2	16	16	–	–	–	–
	GCNConv 3, GMP	16	256	–	–	–	–
MLP	FC 1	256 + [WS, TI]	1024	–	–	–	–
	FC 2	1024	256	–	–	–	–
Decoder	Decode 1-2, Lerp	128	64	3×3	1	1	–
	Decode 3-4, Lerp	64	32	3×3	1	1	–
	Decode 5-6, Lerp	32	16	5×5	1,2	2	–,1
	Decode 7-8, Lerp	16	8	5×5	1,2	2	–,1
	Decode 9-10, Lerp	8	4	5×5	1,2	2	–,1
	Decode 11	4	$out_{channels}$	5×5	1	2	–

A.2 Preliminary Results of the GCN-CNN Architecture

A single GCN-CNN model was trained using the TurbOPark dataset, as described in subsection 3.2. The training process, which followed the system setup outlined in subsection 4.4, ran for 1000 epochs. One notable advantage of the GCN-CNN architecture is its relatively low training cost. Completing all 1000 epochs took just over 1.5 hours, making the training process approximately ten times faster than the other models discussed in this thesis. However, despite its efficiency, the GCN-CNN model underperforms compared to the CAE and CAE/MLP networks, as shown in Table 4.3. The GCN-CNN model achieved a Mean Absolute Error (MAE) of $1.81 \times 10^{-2}\,\mathrm{m/s}$ and a Mean Squared Error (MSE) of $6.92 \times 10^{-3}\,\mathrm{m^2/s^2}$. The 1D and 2D results for the GCN-CNN architecture are illustrated in Figure A.1, Figure A.2, Figure A.3, and Figure A.4.

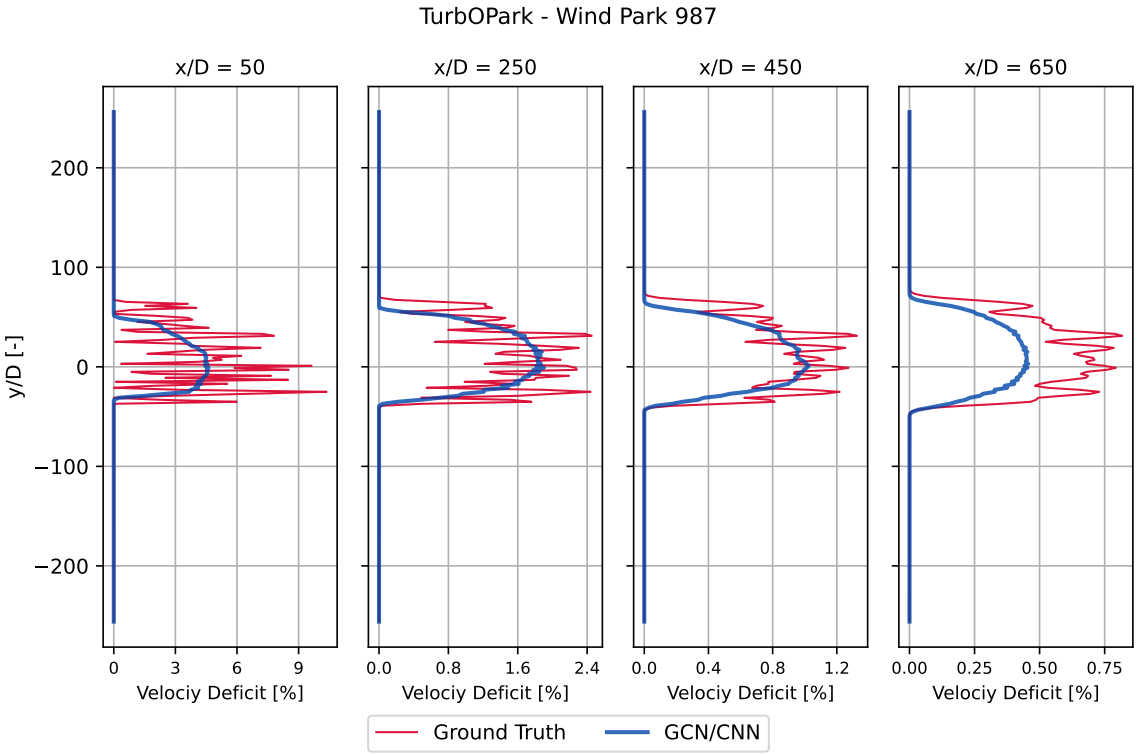


Figure A.1: Wake deficit profiles for Wind Park 987 using the TurbOPark model. Profiles shown at multiple downstream distances (x/D) behind wind farms for the GCN/CNN.

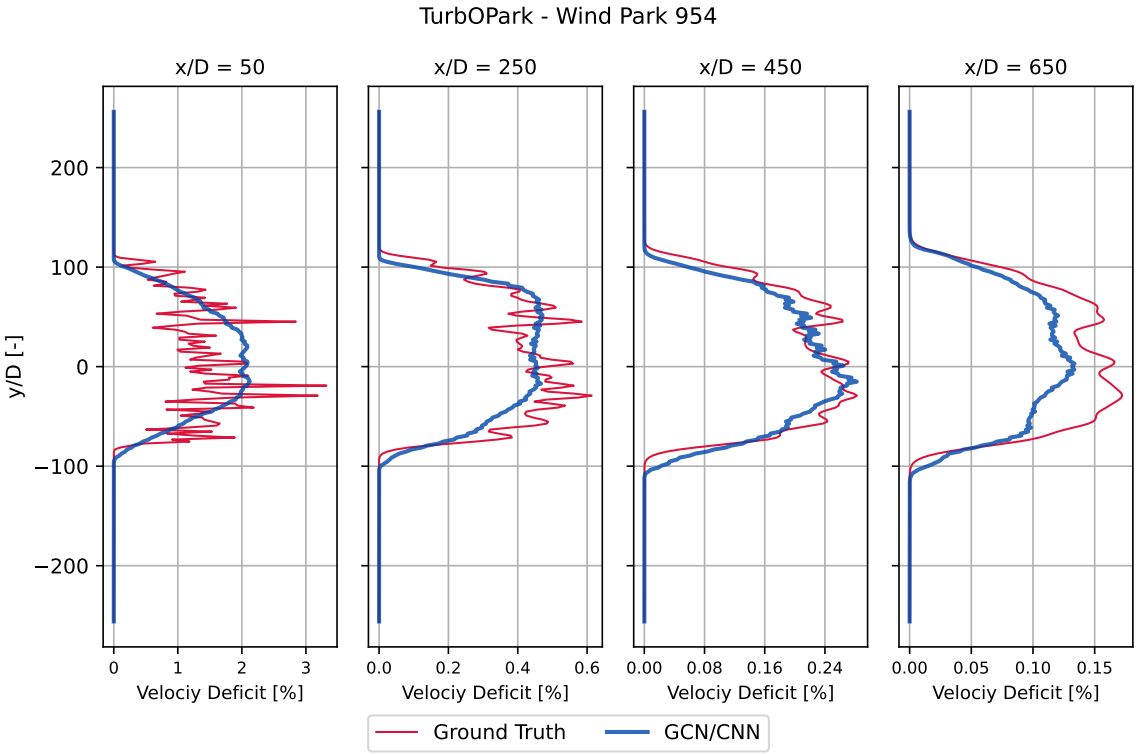


Figure A.2: Wake deficit profiles for Wind Park 954 using the TurbOPark model. Profiles shown at multiple downstream distances (x/D) behind wind farms for the GCN/CNN

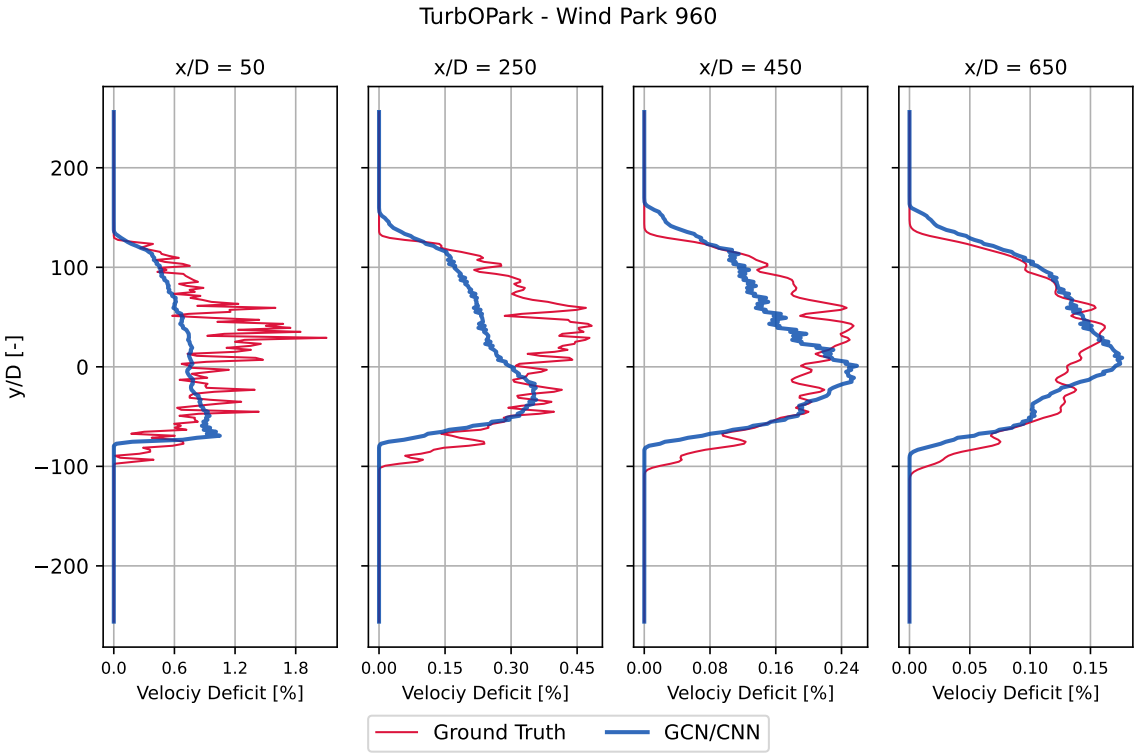


Figure A.3: Wake deficit profiles for Wind Park 960 using the TurbOPark model. Profiles shown at multiple downstream distances (x/D) behind wind farms for the GCN/CNN.

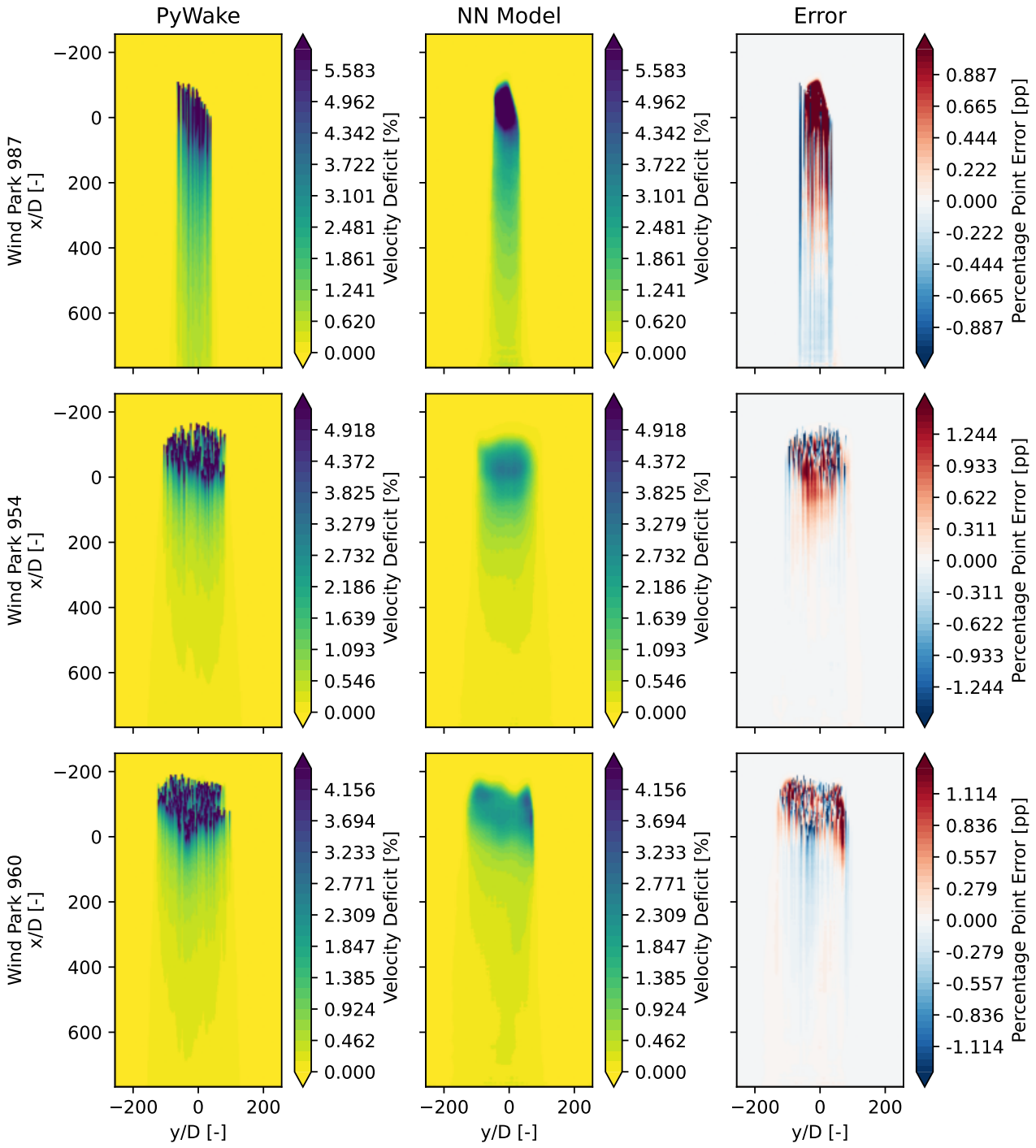


Figure A.4: Velocity deficit [%] comparison for test wind parks (987, 954, and 960). Left column: TurbOPark Wake Deficit Model outputs. Middle column: Predictions from a GCN/CNN trained on the TurbOPark model. Right column: Percentage Point Error [pp] between the TurbOPark model and the GCN/CNN predictions.

B Wind Park Generation

Figure B.1 illustrates the distribution and relationships between key input parameters used in the wind park generation algorithm. This pairplot combines scatter plots and histograms to visualize the distributions of turbulence intensity, turbine position noise, polygon irregularities, and polygon spikiness. These parameters were crucial in creating diverse and realistic wind

park layouts for both the training and test datasets used throughout this thesis. The diagonal of the plot shows histograms for each parameter, providing insight into their individual distributions. The off-diagonal scatter plots reveal potential correlations or patterns between pairs of parameters. This visualization helps to ensure a wide range of wind park configurations were considered, enhancing the robustness and applicability of the neural network models developed in this study.

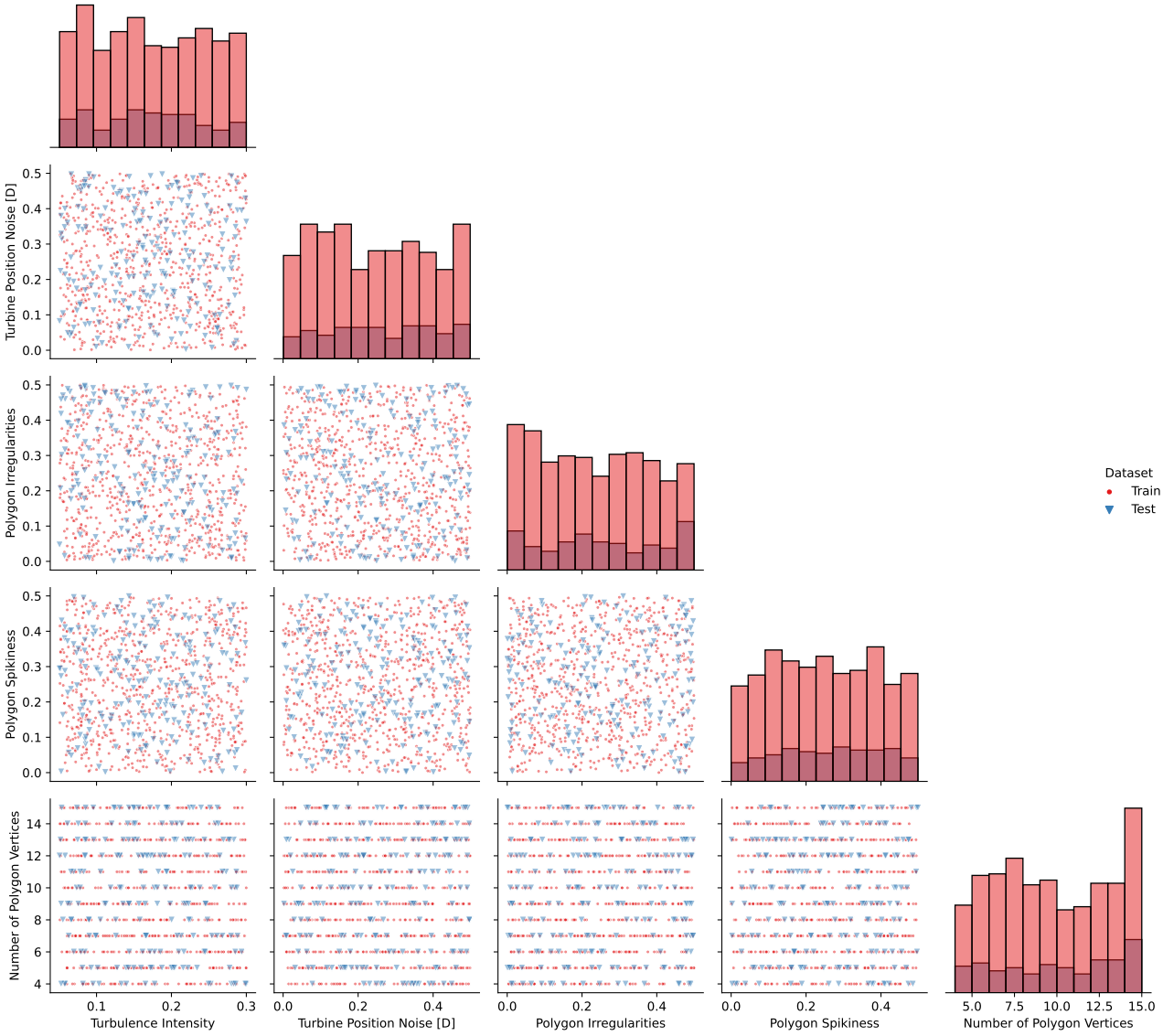


Figure B.1: Distribution of input parameters for wind park generation algorithm.

For the remain wind park generation parameters, see Figure 3.4.

C Wake Deficit Prediction in 2D

C.1 Convolutional Autoencoder

C.1.1 Jensen Wake Deficit Model

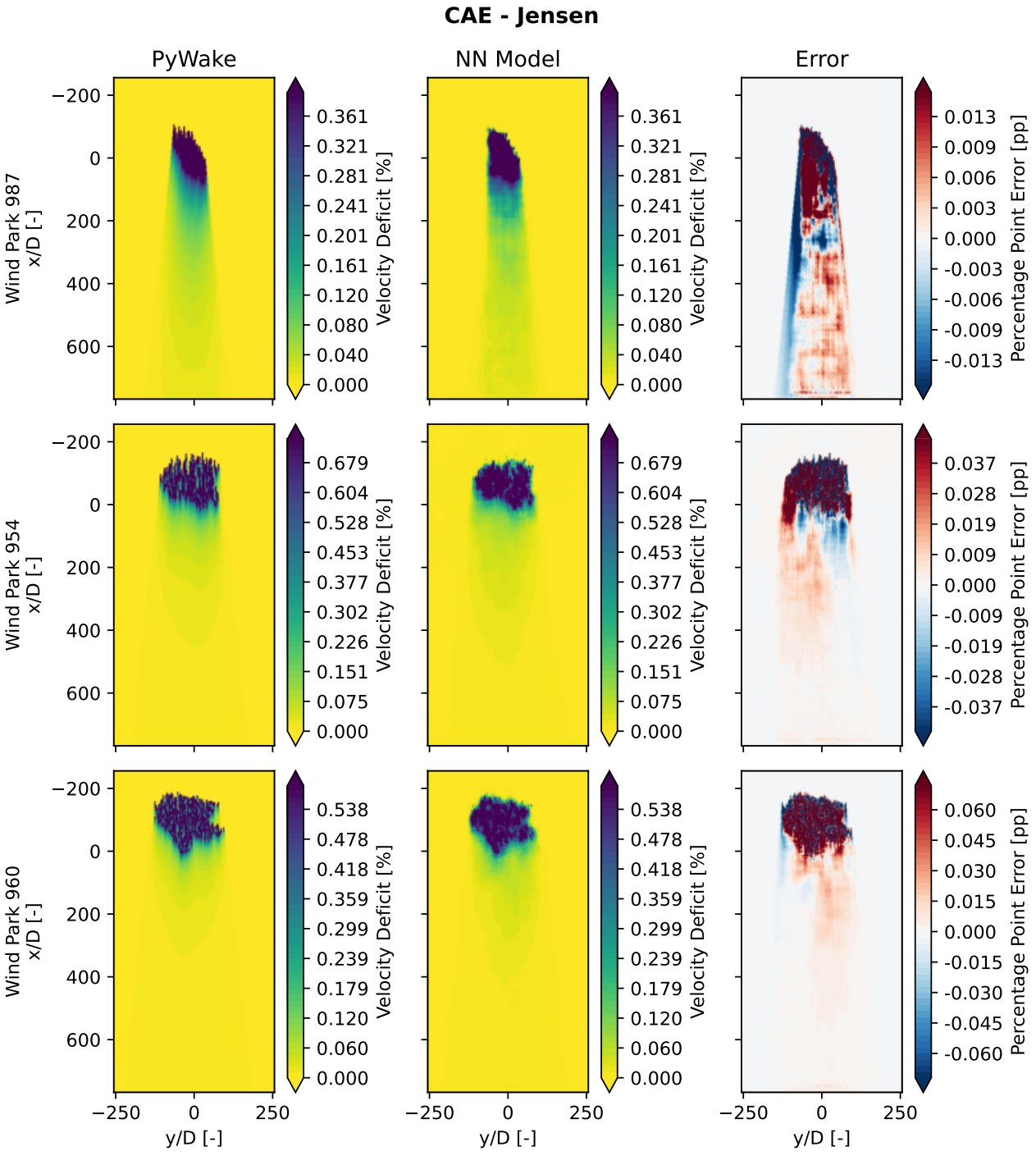


Figure C.1: Velocity deficit [%] comparison for test wind parks (987, 954, and 960). Left column: Jensen Wake Deficit Model outputs. Middle column: Predictions from a CAE trained on the Jensen model. Right column: Percentage Point Error [pp] between the Jensen model and the CAE predictions.

C.1.2 Bastankhah Wake Deficit Model

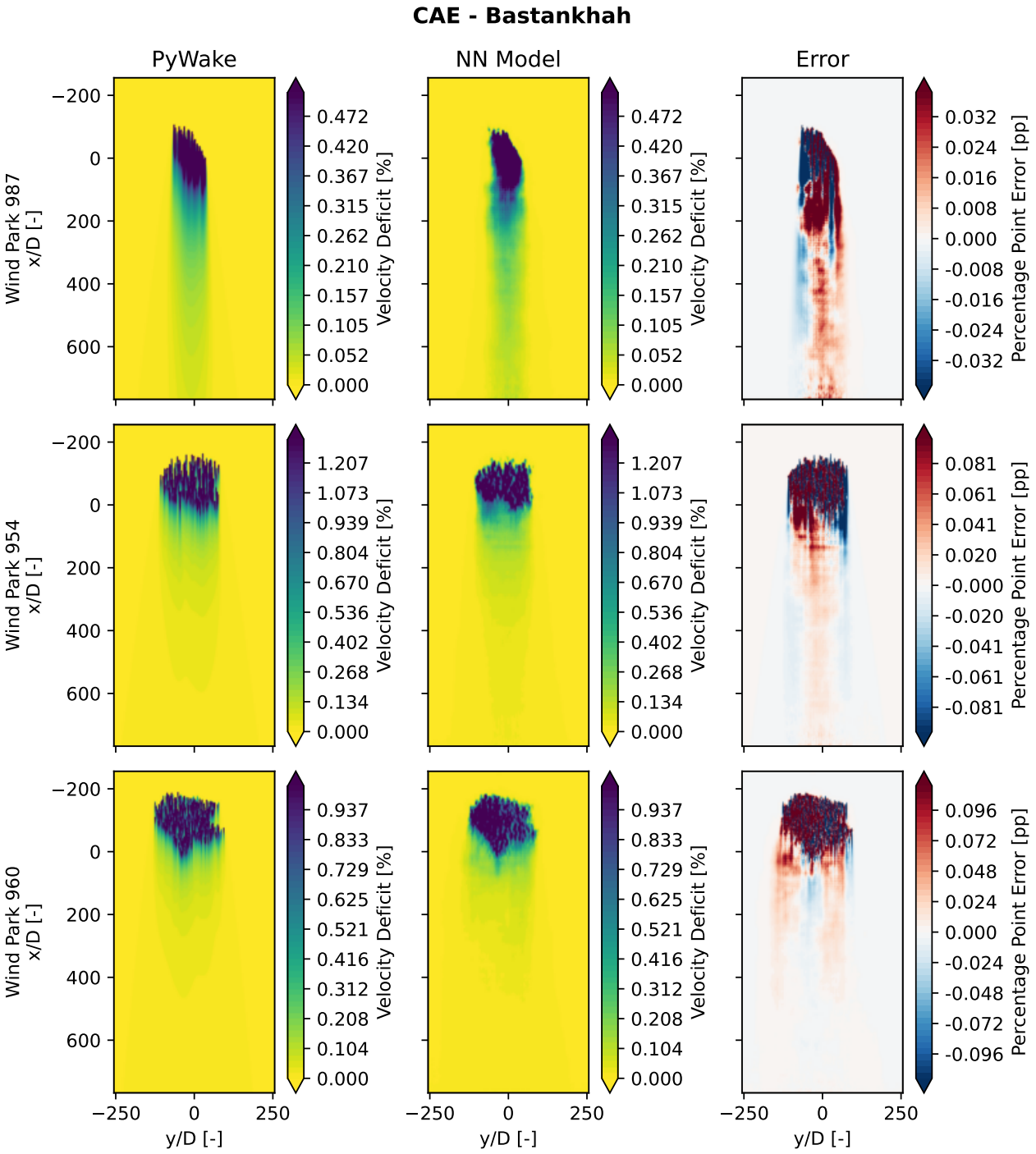


Figure C.2: Velocity deficit [%] comparison for test wind parks (987, 954, and 960). Left column: Bastankhah Wake Deficit Model outputs. Middle column: Predictions from a CAE trained on the Bastankhah model. Right column: Percentage Point Error [pp] between the Bastankhah model and the CAE predictions.

C.1.3 TurbOPark Wake Deficit Model

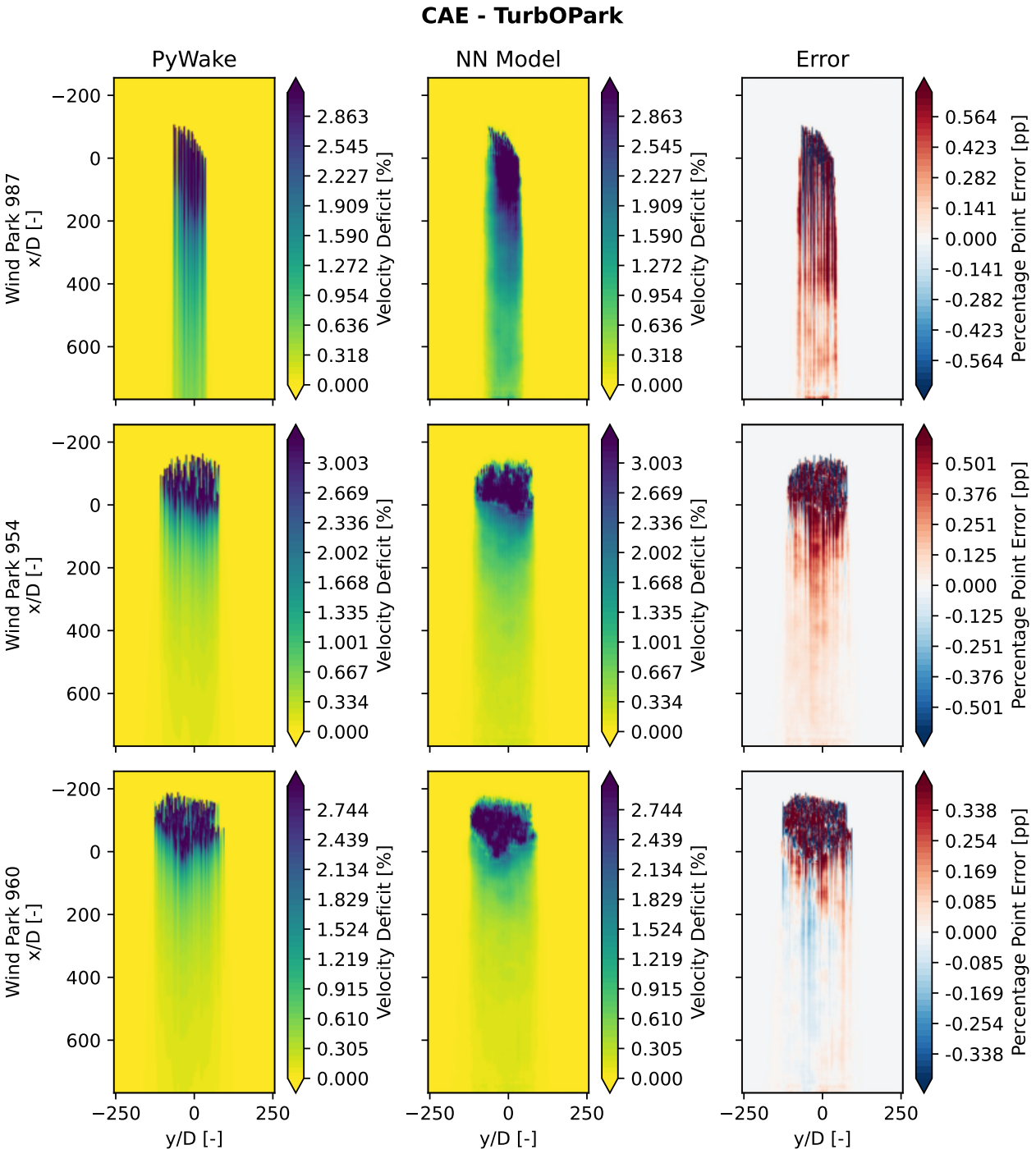


Figure C.3: Velocity deficit [%] comparison for test wind parks (987, 954, and 960). Left column: TurbOPark Wake Deficit Model outputs. Middle column: Predictions from a CAE trained on the TurbOPark model. Right column: Percentage Point Error [pp] between the TurbOPark model and the CAE predictions.

C.2 U-Net

C.2.1 Jensen Wake Deficit Model

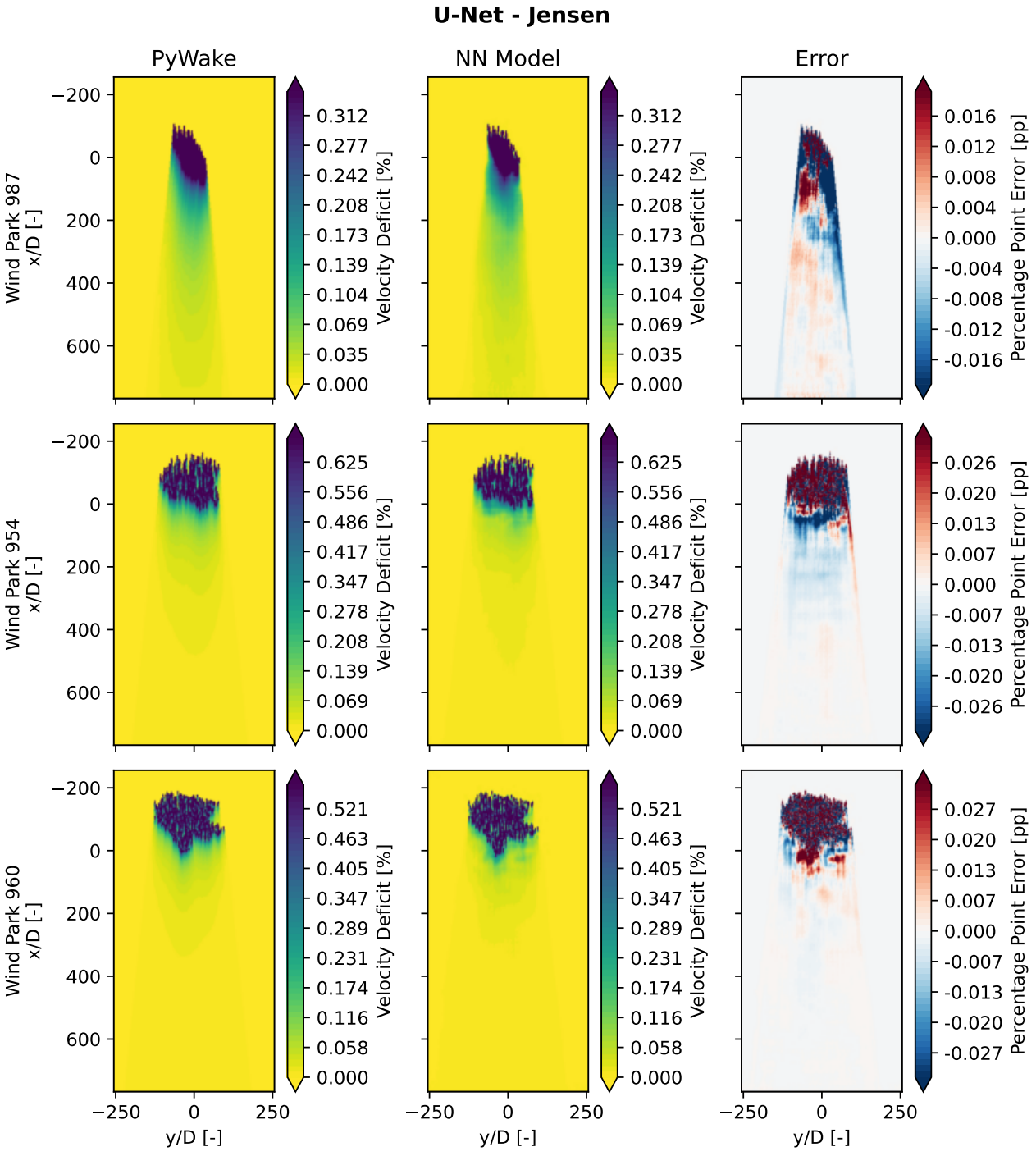


Figure C.4: Velocity deficit [%] comparison for test wind parks (987, 954, and 960). Left column: Jensen Wake Deficit Model outputs. Middle column: Predictions from a U-Net trained on the Jensen model. Right column: Percentage Point Error [pp] between the Jensen model and the U-Net predictions.

C.2.2 Bastankhah Wake Deficit Model

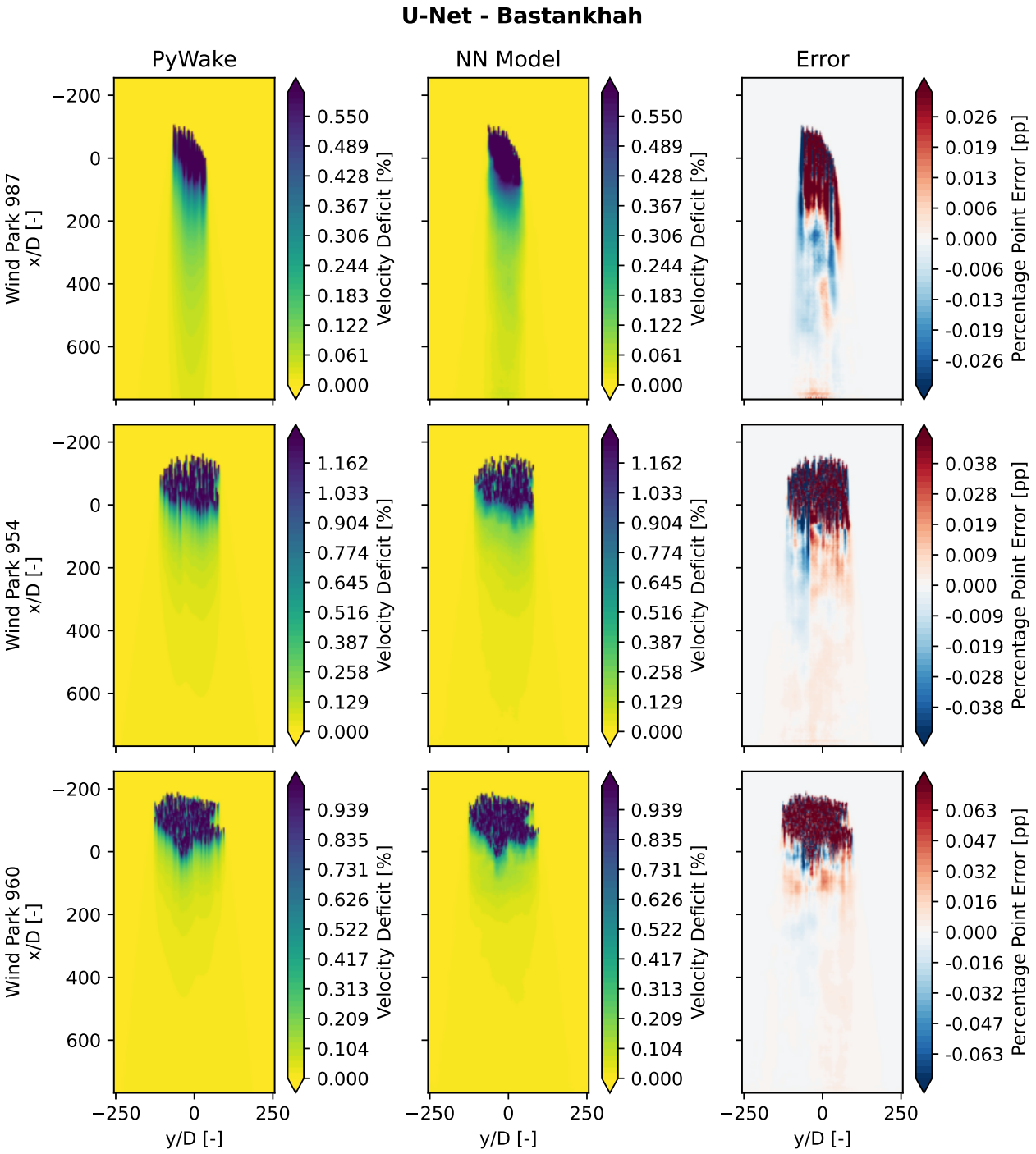


Figure C.5: Velocity deficit [%] comparison for test wind parks (987, 954, and 960). Left column: Bastankhah Wake Deficit Model outputs. Middle column: Predictions from a U-Net trained on the Bastankhah model. Right column: Percentage Point Error [pp] between the Bastankhah model and the U-Net predictions.

C.2.3 TurbOPark Wake Deficit Model

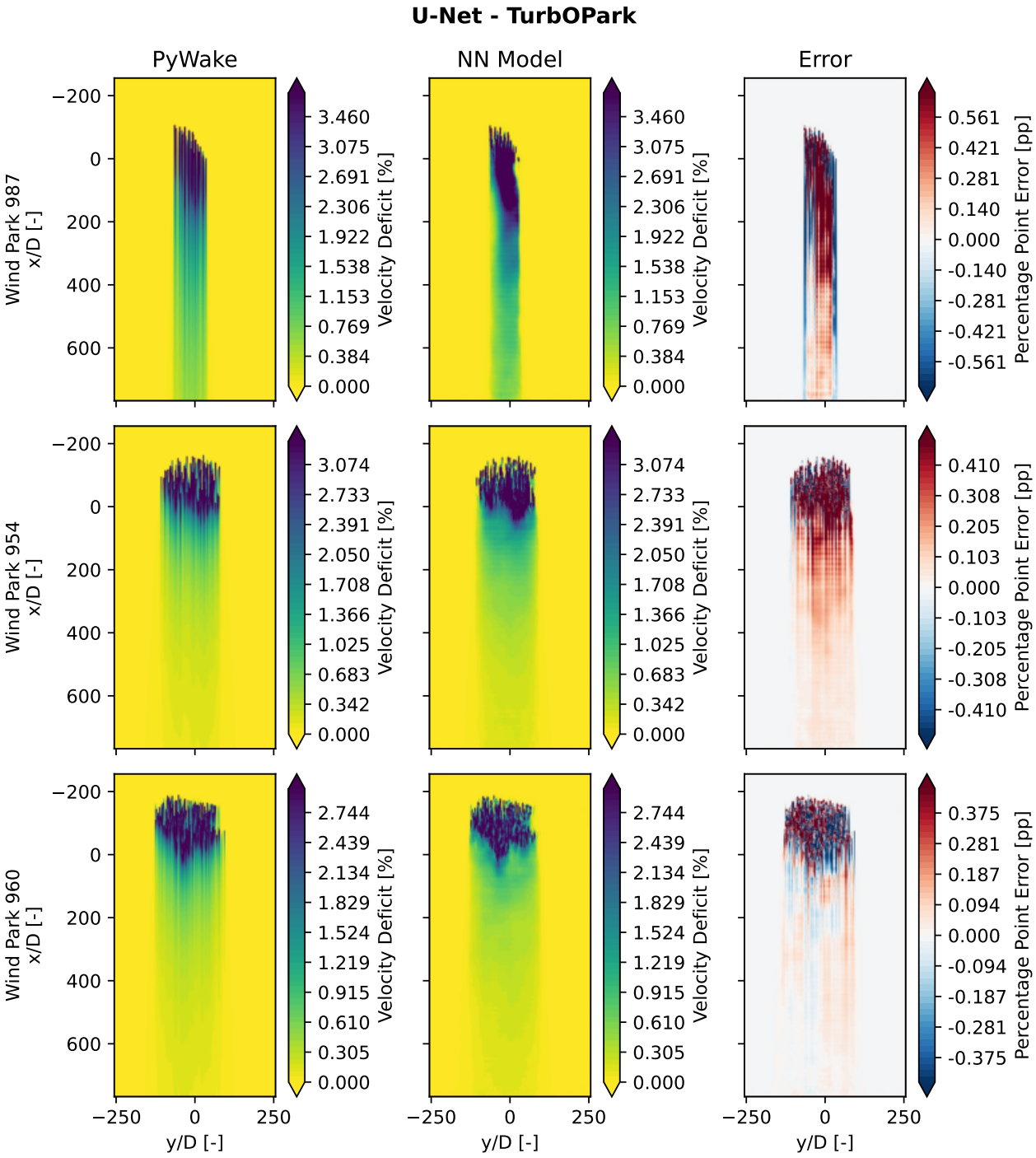


Figure C.6: Velocity deficit [%] comparison for test wind parks (987, 954, and 960). Left column: TurbOPark Wake Deficit Model outputs. Middle column: Predictions from a U-Net trained on the TurbOPark model. Right column: Percentage Point Error [pp] between the TurbOPark model and the U-Net predictions.

C.3 Convolutional Autoencoder with Multilayer Perceptron

C.3.1 Jensen Wake Deficit Model

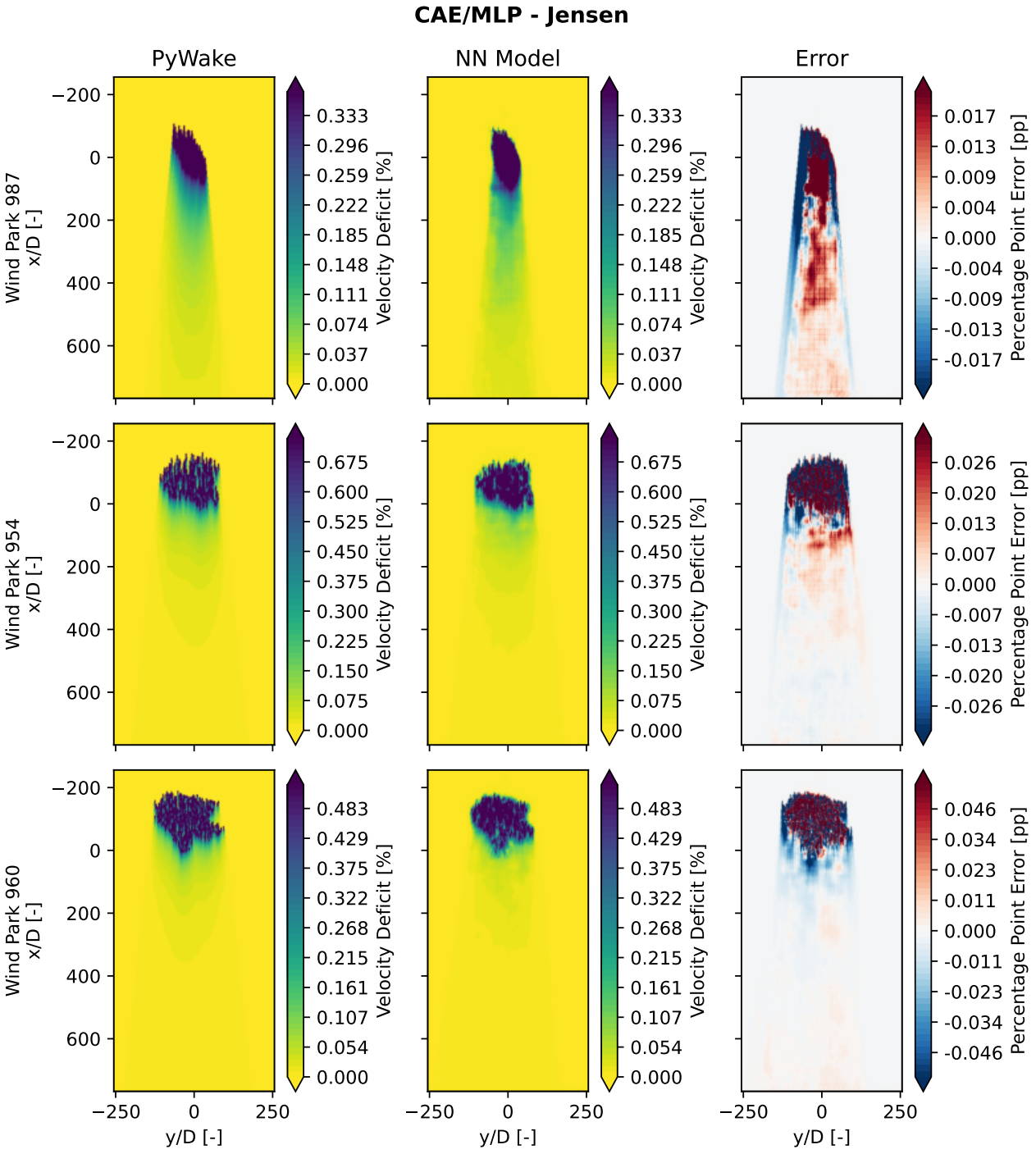


Figure C.7: Velocity deficit [%] comparison for test wind parks (987, 954, and 960). Left column: Jensen Wake Deficit Model outputs. Middle column: Predictions from a CAE/MLP trained on the Jensen model. Right column: Percentage Point Error [pp] between the Jensen model and the CAE/MLP predictions.

C.3.2 Bastankhah Wake Deficit Model

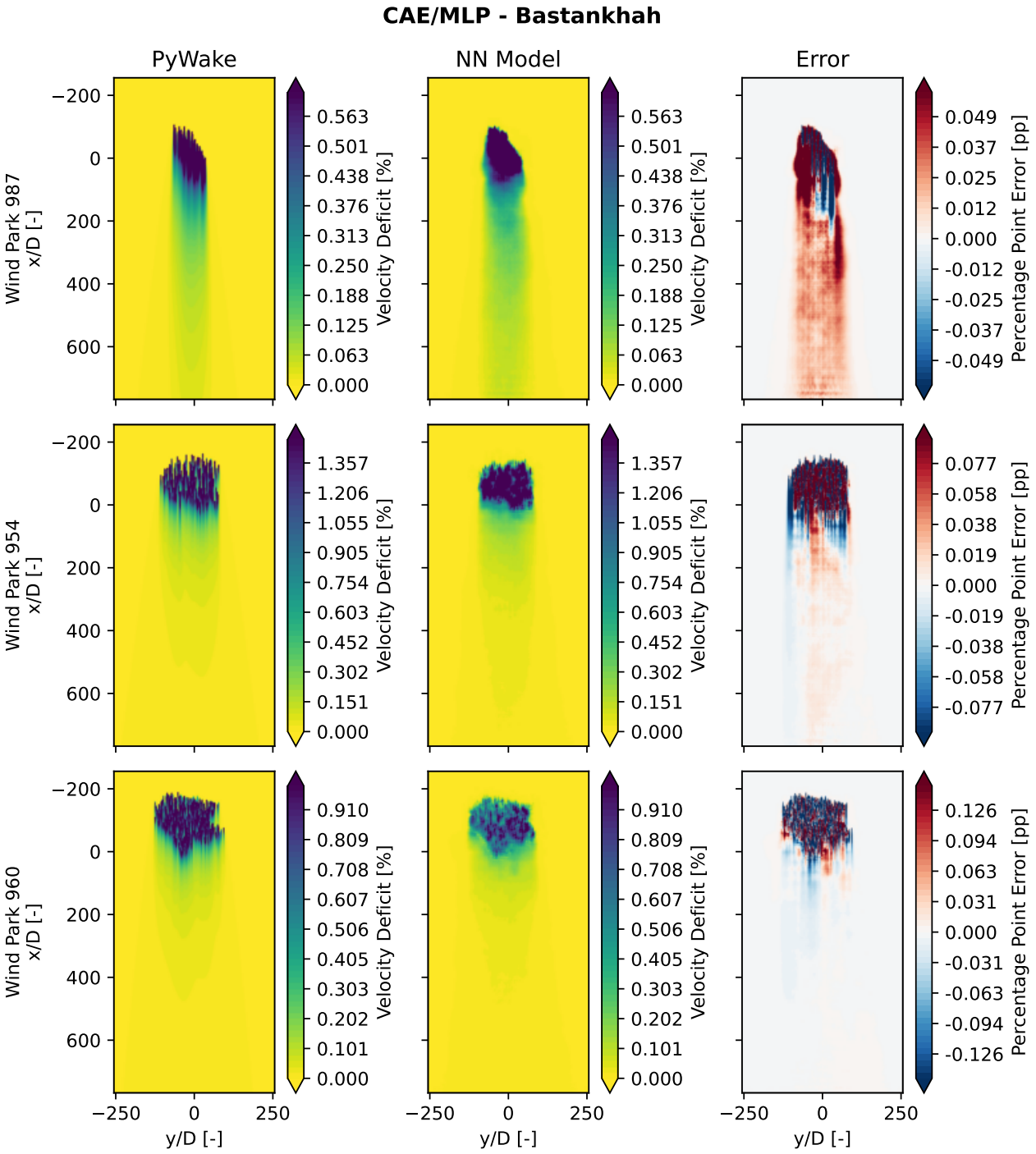


Figure C.8: Velocity deficit [%] comparison for test wind parks (987, 954, and 960). Left column: Bastankhah Wake Deficit Model outputs. Middle column: Predictions from a CAE/MLP trained on the Bastankhah model. Right column: Percentage Point Error [pp] between the Bastankhah model and the CAE/MLP predictions.

C.3.3 TurbOPark Wake Deficit Model

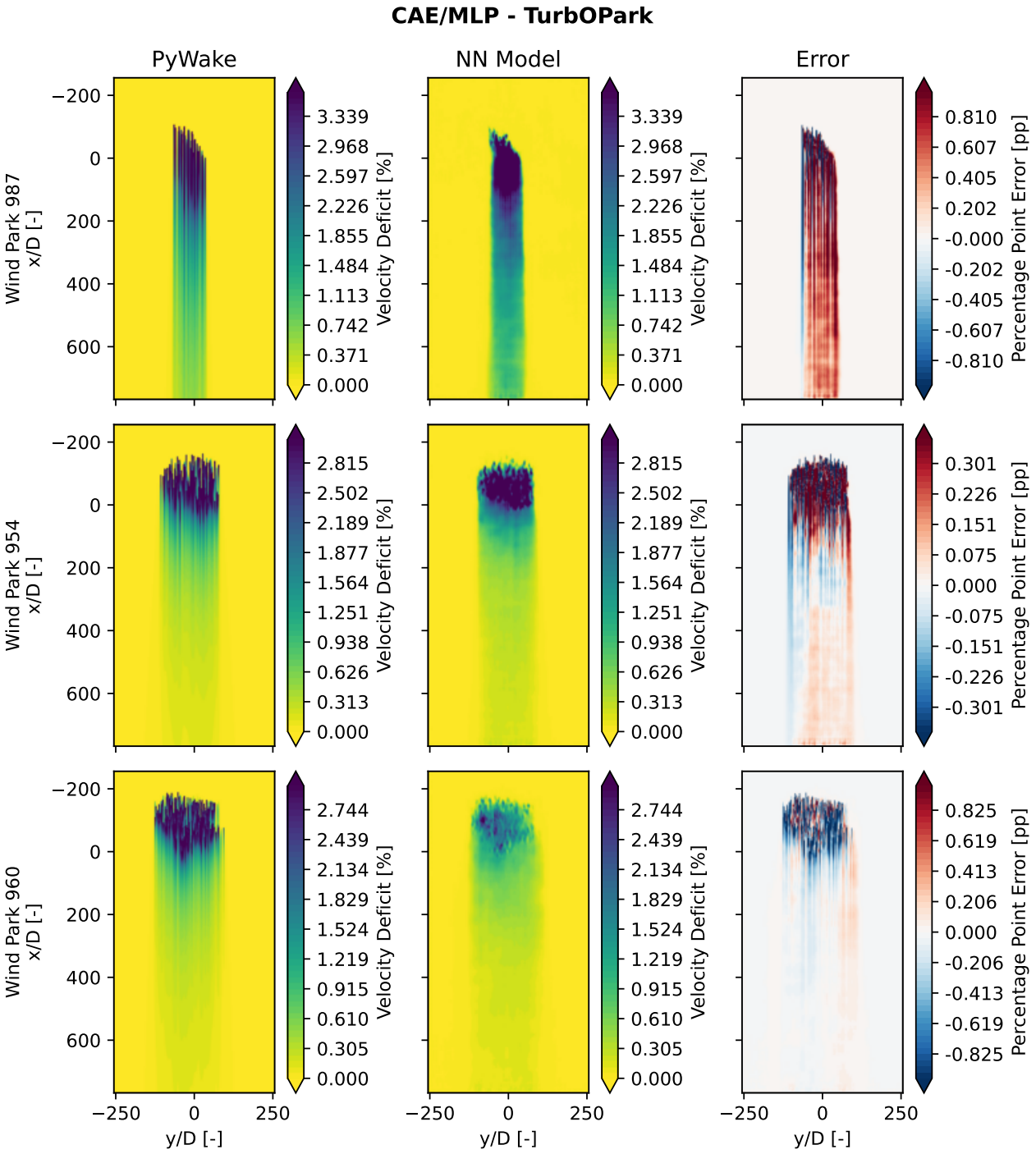


Figure C.9: Velocity deficit [%] comparison for test wind parks (987, 954, and 960). Left column: TurbOPark Wake Deficit Model outputs. Middle column: Predictions from a CAE/MLP trained on the TurbOPark model. Right column: Percentage Point Error [pp] between the TurbOPark model and the CAE/MLP predictions.

C.4 U-Net with Multilayer Perceptron

C.4.1 Jensen Wake Deficit Model

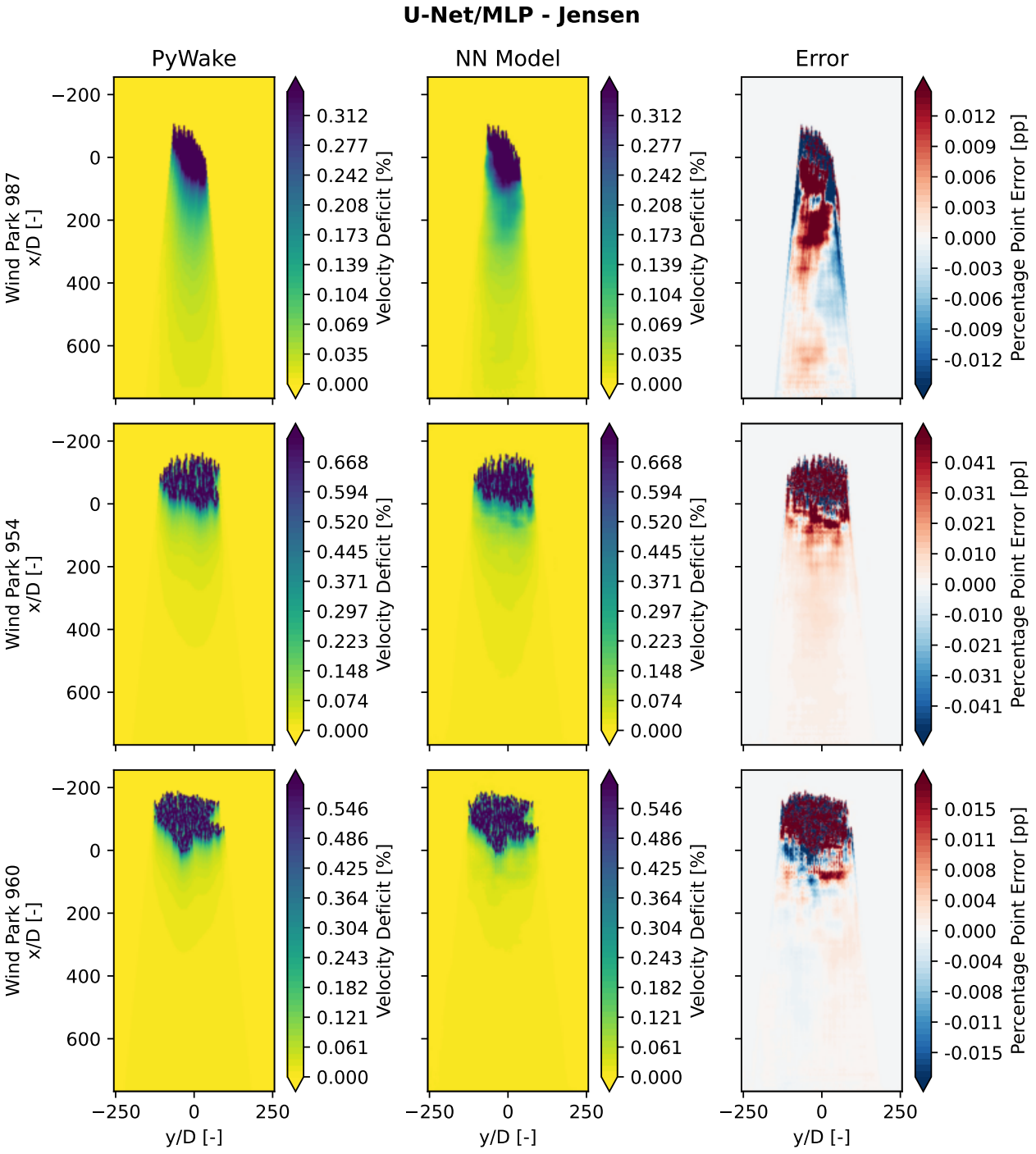


Figure C.10: Velocity deficit [%] comparison for test wind parks (987, 954, and 960). Left column: Jensen Wake Deficit Model outputs. Middle column: Predictions from a U-Net/MLP trained on the Jensen model. Right column: Percentage Point Error [pp] between the Jensen model and the U-Net/MLP predictions.

C.4.2 Bastankhah Wake Deficit Model

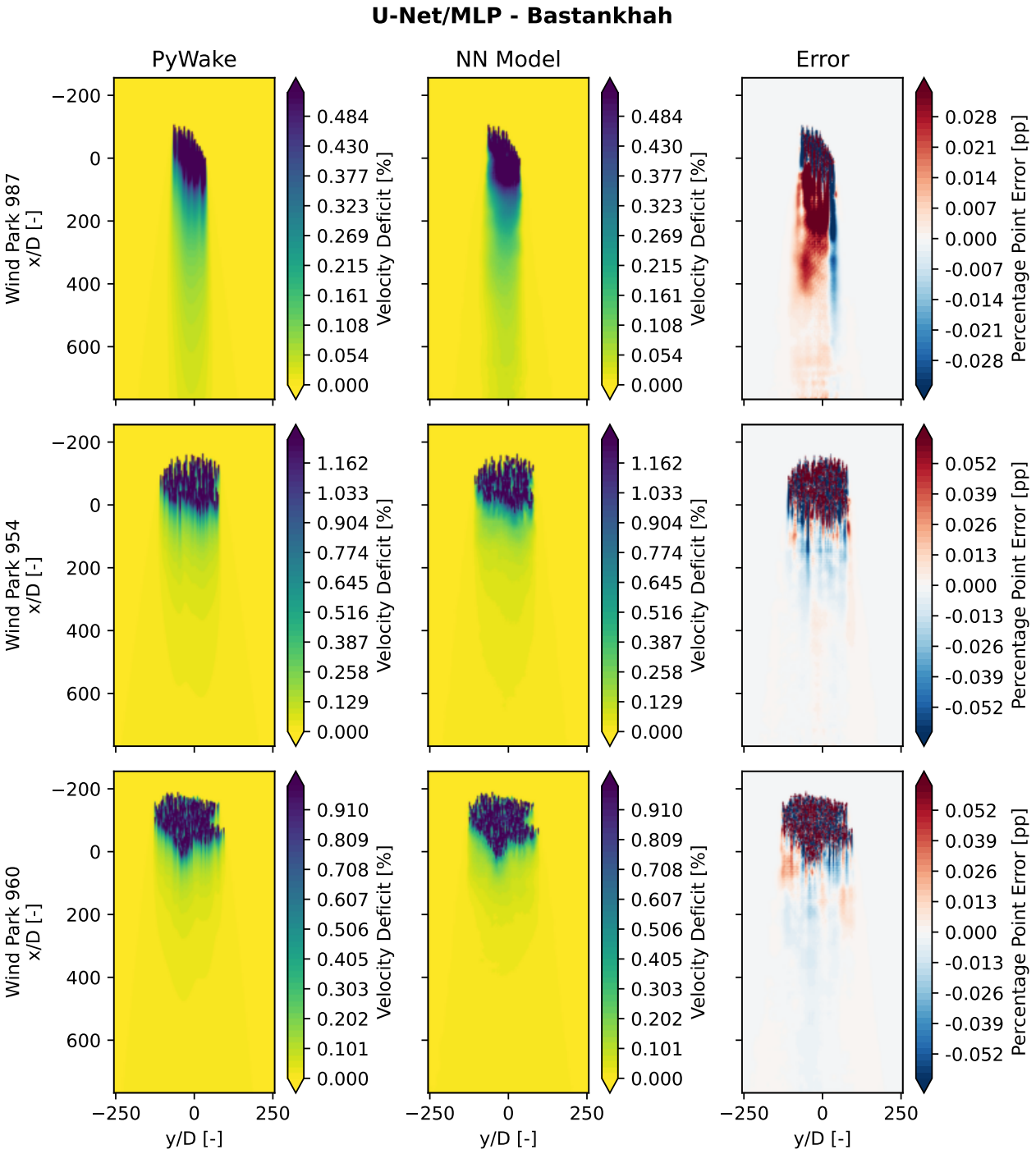


Figure C.11: Velocity deficit [%] comparison for test wind parks (987, 954, and 960). Left column: Bastankhah Wake Deficit Model outputs. Middle column: Predictions from a U-Net/MLP trained on the Bastankhah model. Right column: Percentage Point Error [pp] between the Bastankhah model and the U-Net/MLP predictions.

C.4.3 TurbOPark Wake Deficit Model

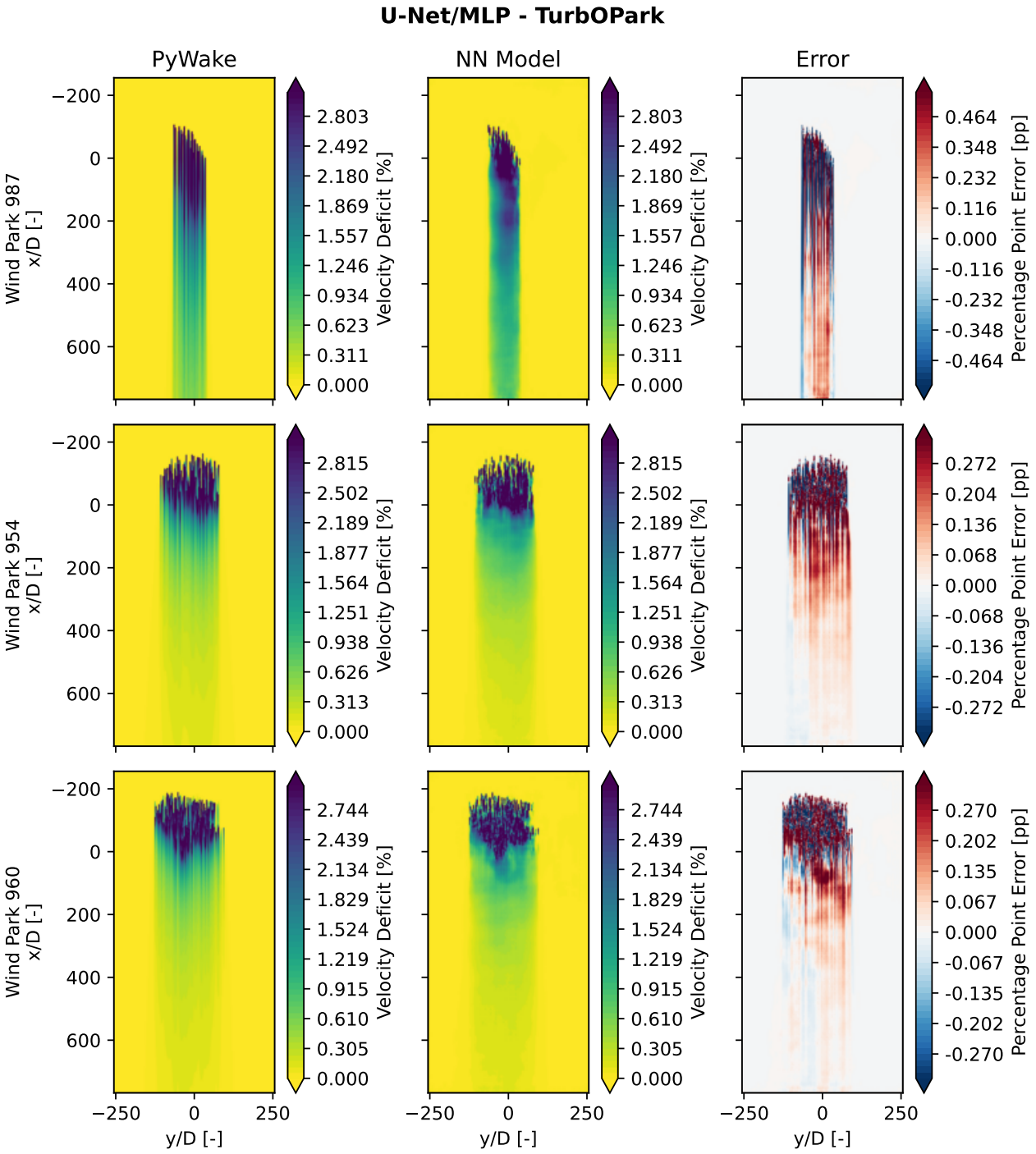


Figure C.12: Velocity deficit [%] comparison for test wind parks (987, 954, and 960). Left column: TurbOPark Wake Deficit Model outputs. Middle column: Predictions from a U-Net/MLP trained on the TurbOPark model. Right column: Percentage Point Error [pp] between the TurbOPark model and the U-Net/MLP predictions.

C.5 Wake-generated Turbulence Prediction

C.5.1 Crespo-Hernández Model

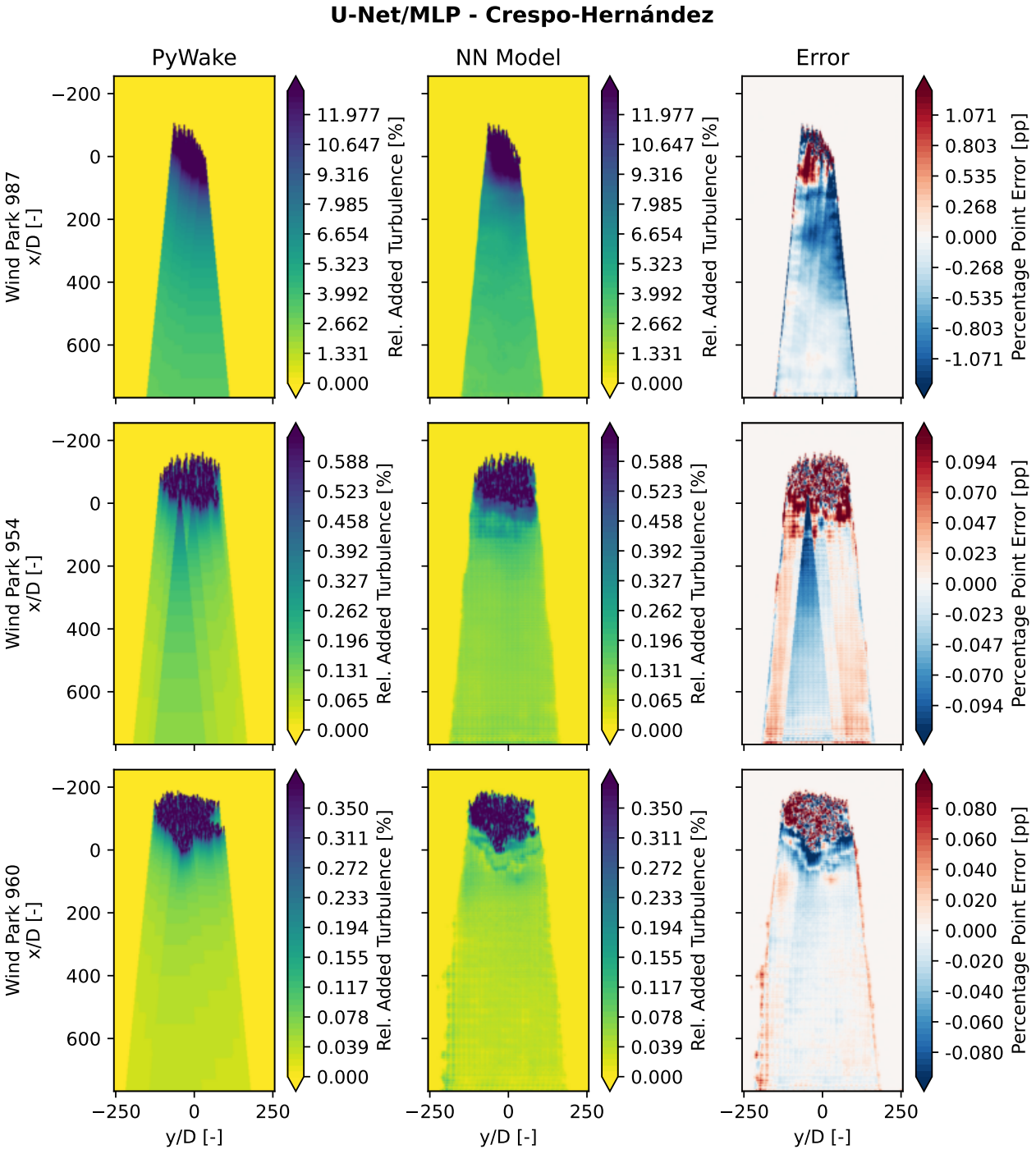


Figure C.13: Rel. Added Turbulence [%] comparison for test wind parks (987, 954, and 960). Left column: Crespo-Hernández Turbulence Model outputs. Middle column: Predictions from a U-Net/MLP trained on the Crespo-Hernández model. Right column: Percentage Point Error [pp] between the Crespo-Hernández model and the U-Net/MLP predictions.

C.5.2 Frandsen Model (2005)

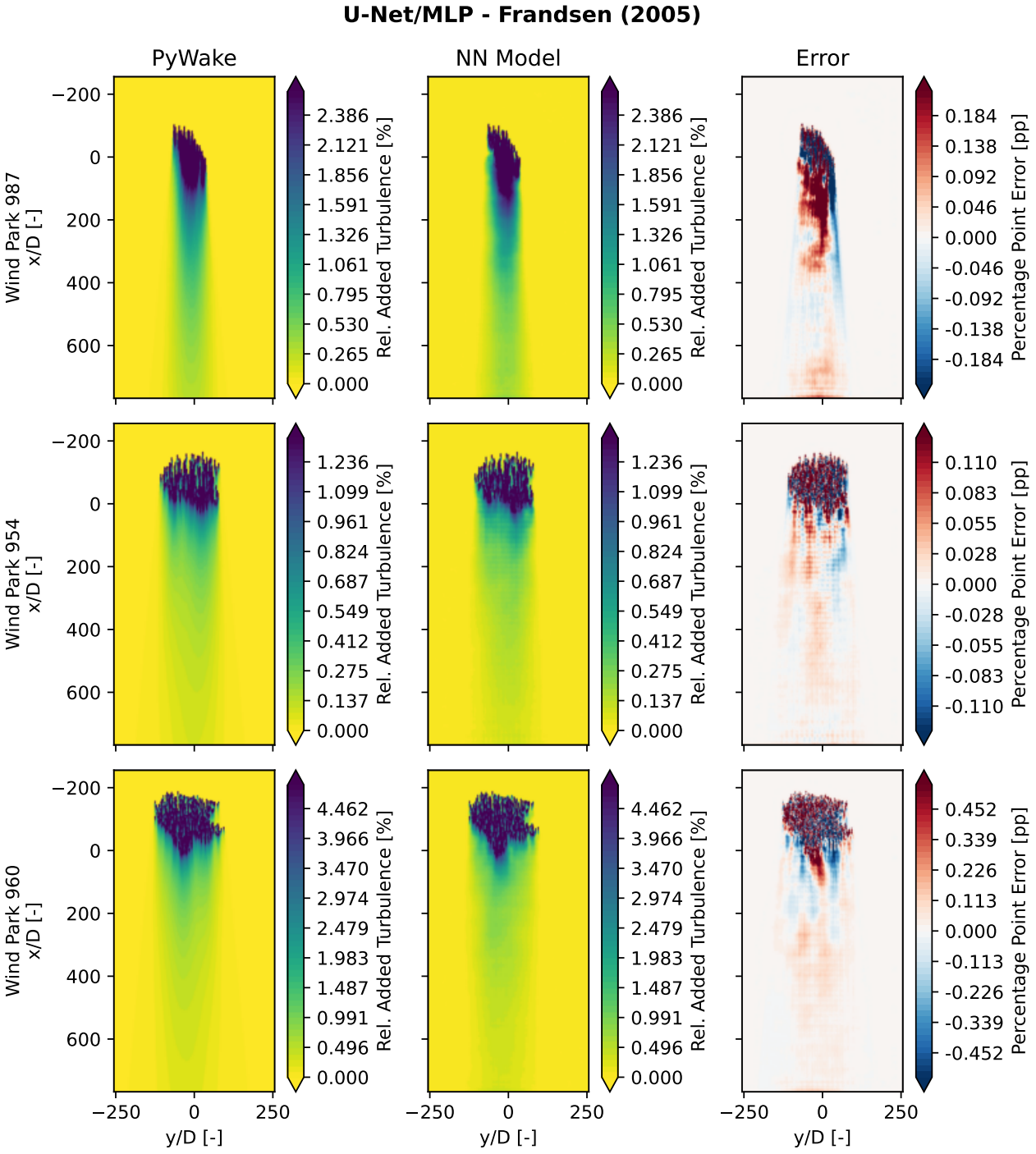


Figure C.14: Rel. Added Turbulence [%] comparison for test wind parks (987, 954, and 960). Left column: Frandsen (2005) Turbulence Model outputs. Middle column: Predictions from a U-Net/MLP trained on the Frandsen (2005) model. Right column: Percentage Point Error [pp] between the Frandsen (2005) model and the U-Net/MLP predictions.

C.5.3 Frandsen Model (2017)

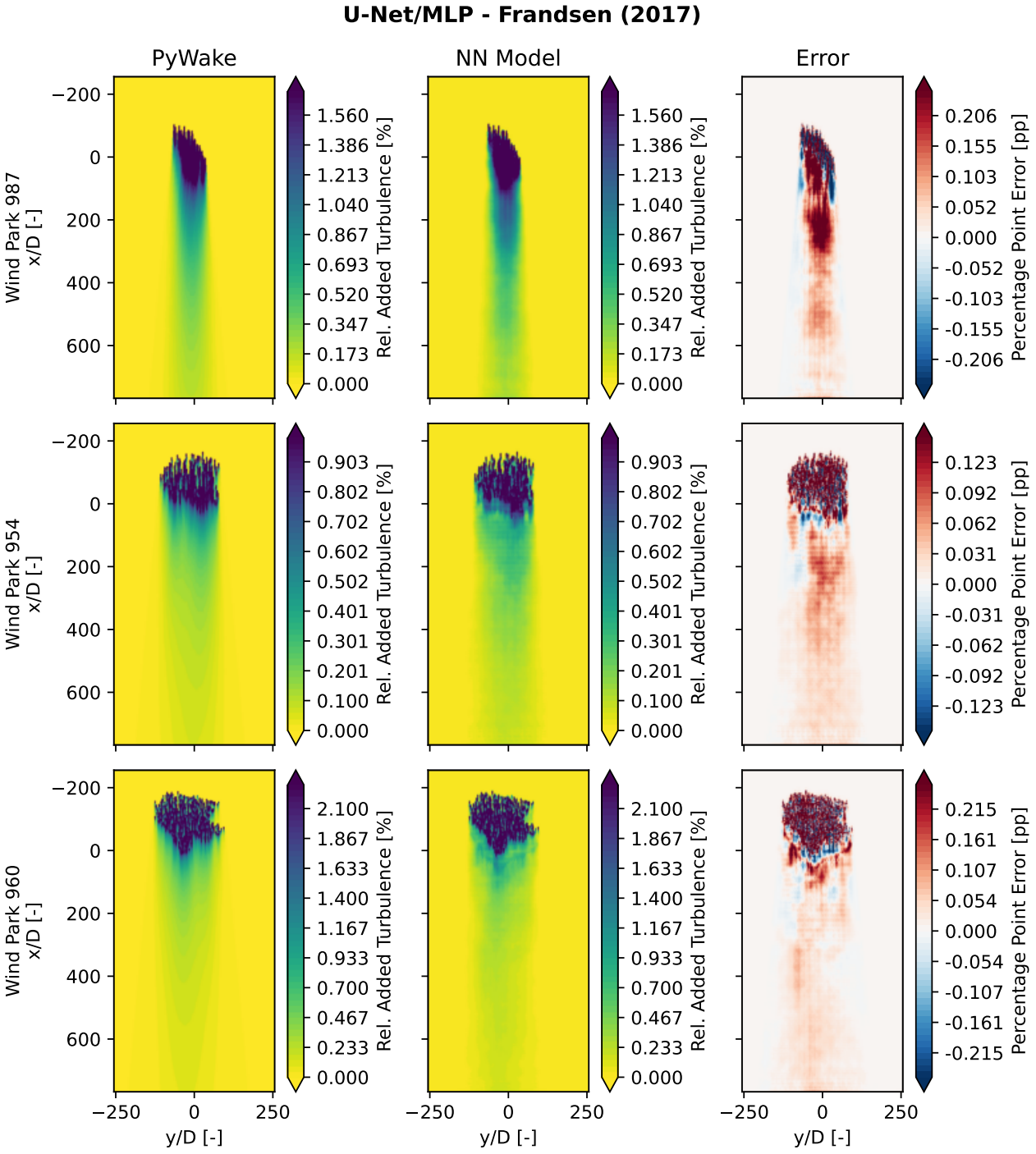


Figure C.15: Rel. Added Turbulence [%] comparison for test wind parks (987, 954, and 960). Left column: Frandsen (2005) Turbulence Model outputs. Middle column: Predictions from a U-Net/MLP trained on the Frandsen (2017) model. Right column: Percentage Point Error [pp] between the Frandsen (2017) model and the U-Net/MLP predictions.

D Wind Park Layouts

In the following section there is an overview over the 1000 wind park layouts used to generate the Jensen, Bastankhah and TurbOPark training data. The wind parks are split into a training

and test dataset, where 80% corresponding to 800 wind parks of the wind parks are used for training the models and 20% corresponding to 200 wind parks are used to validate the performance of the models.

D.1 Training Dataset Wind Parks

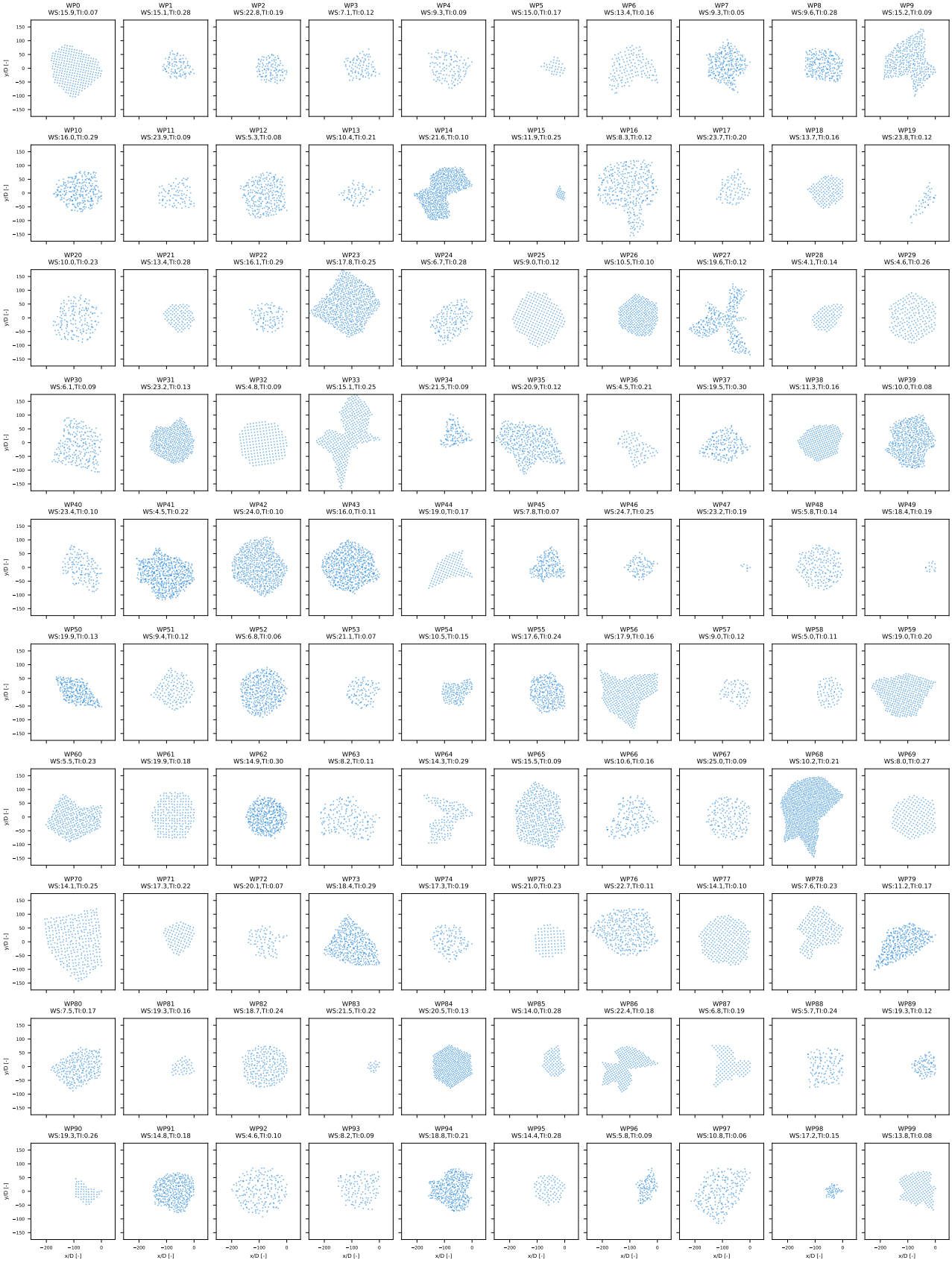


Figure D.1: 10x10 grid of wind park layouts (WP0-WP99) used for training. Wind speed (WS) and turbulence intensity (TI) are indicated for each park.

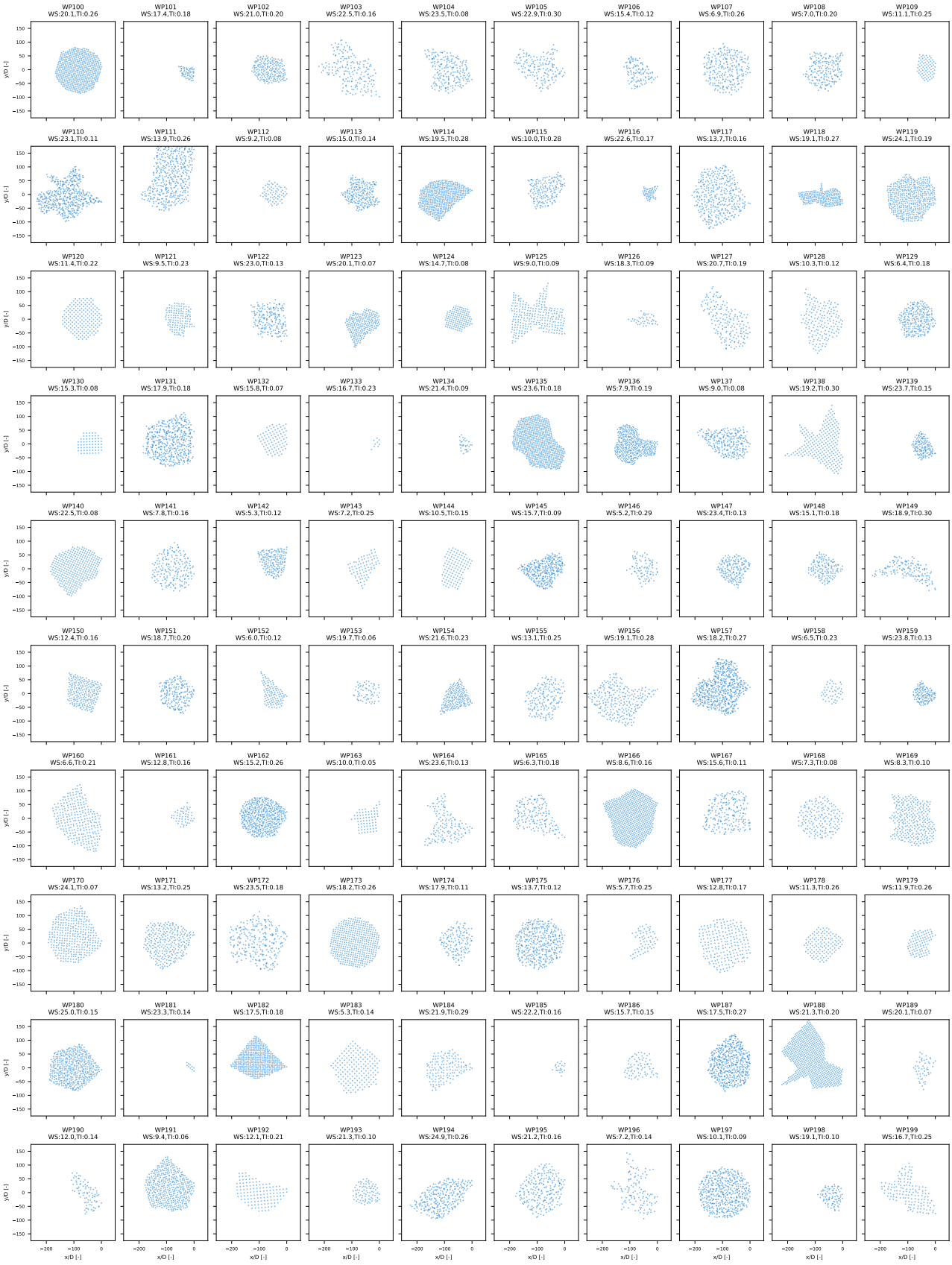


Figure D.2: 10x10 grid of wind park layouts (WP100-WP199) used for training. Wind speed (WS) and turbulence intensity (TI) are indicated for each park.

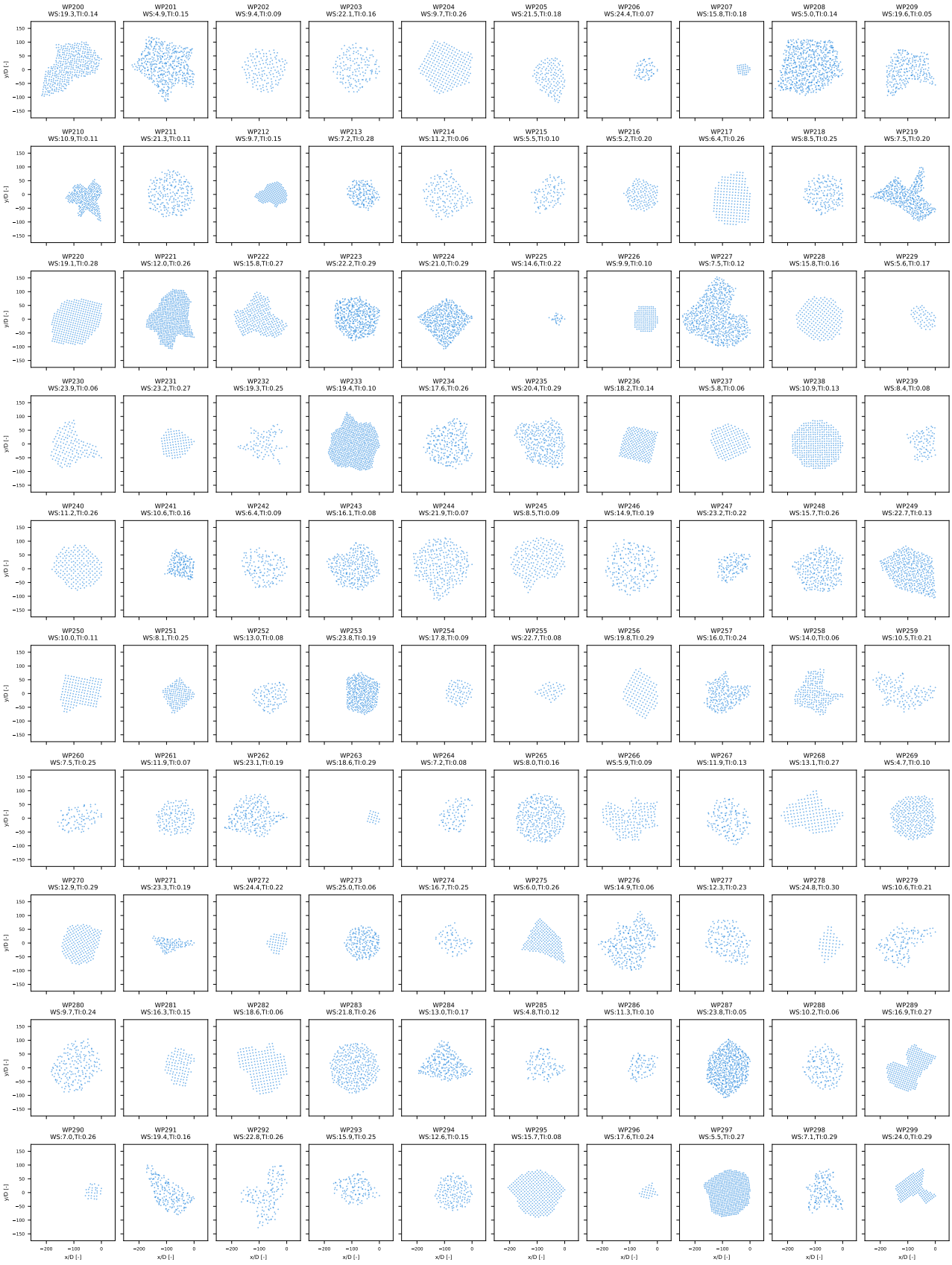


Figure D.3: 10x10 grid of wind park layouts (WP200-WP299) used for training. Wind speed (WS) and turbulence intensity (TI) are indicated for each park.

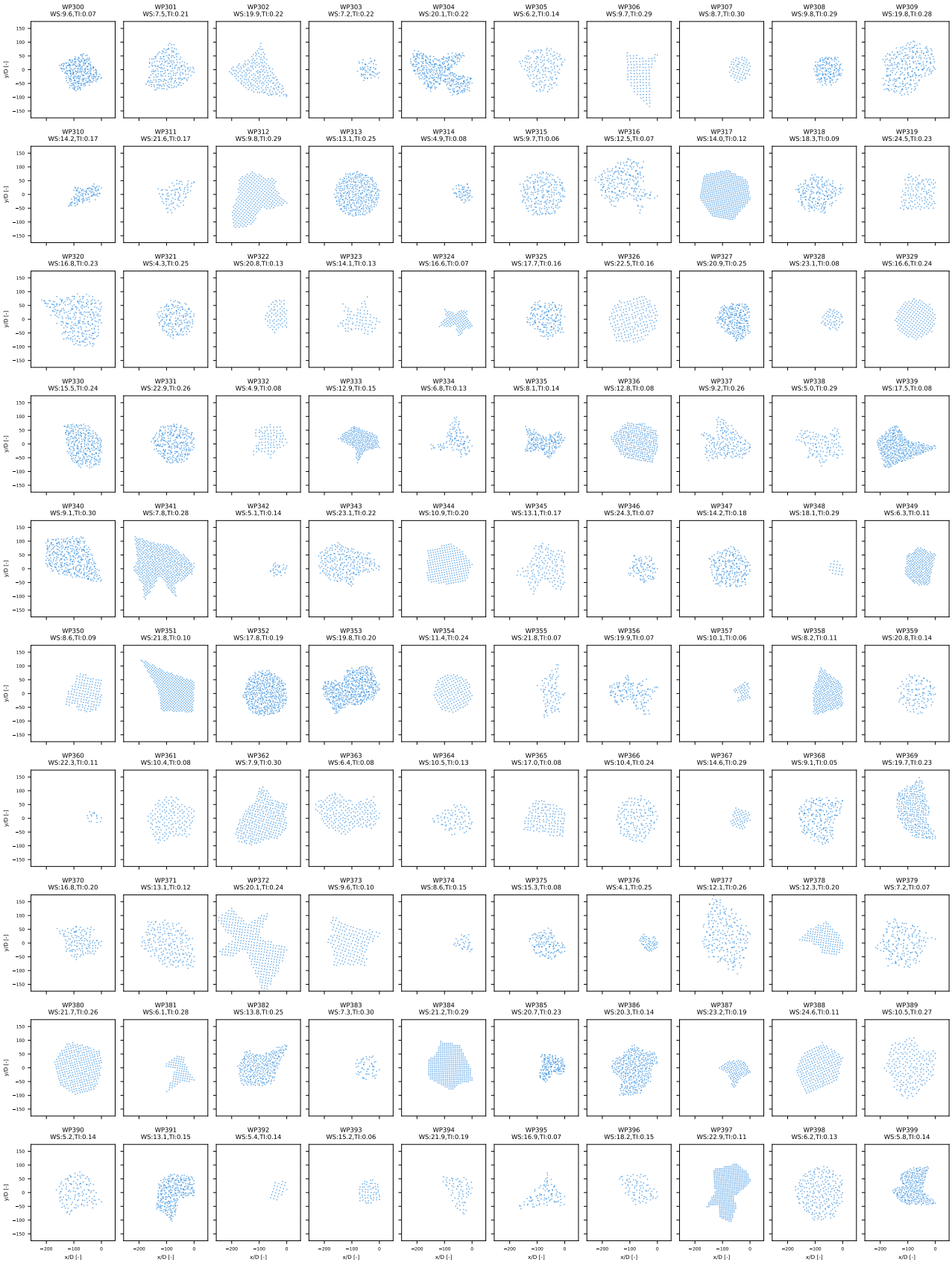


Figure D.4: 10x10 grid of wind park layouts (WP300-WP399) used for training. Wind speed (WS) and turbulence intensity (TI) are indicated for each park.

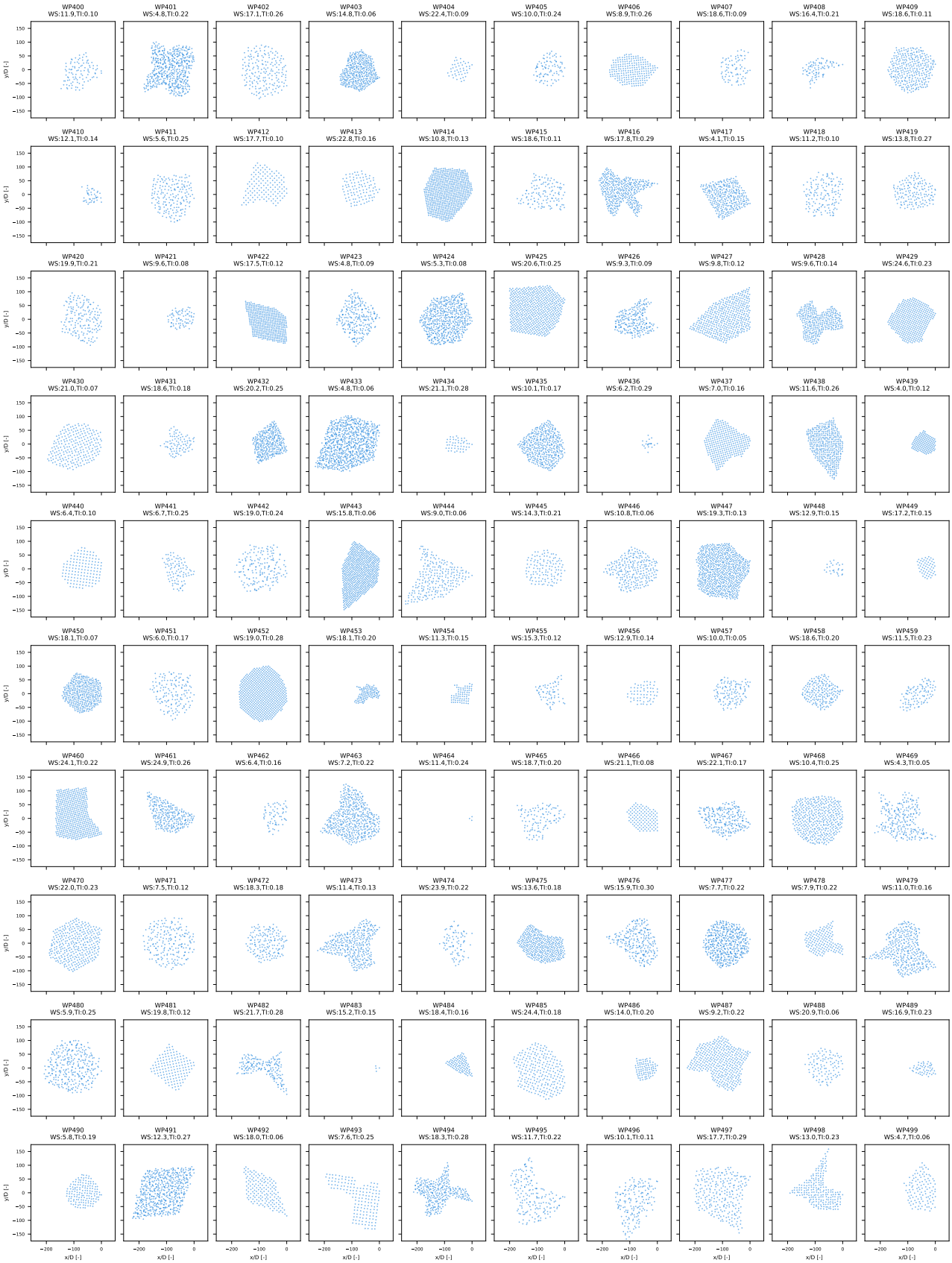


Figure D.5: 10x10 grid of wind park layouts (WP400-WP499) used for training. Wind speed (WS) and turbulence intensity (TI) are indicated for each park.

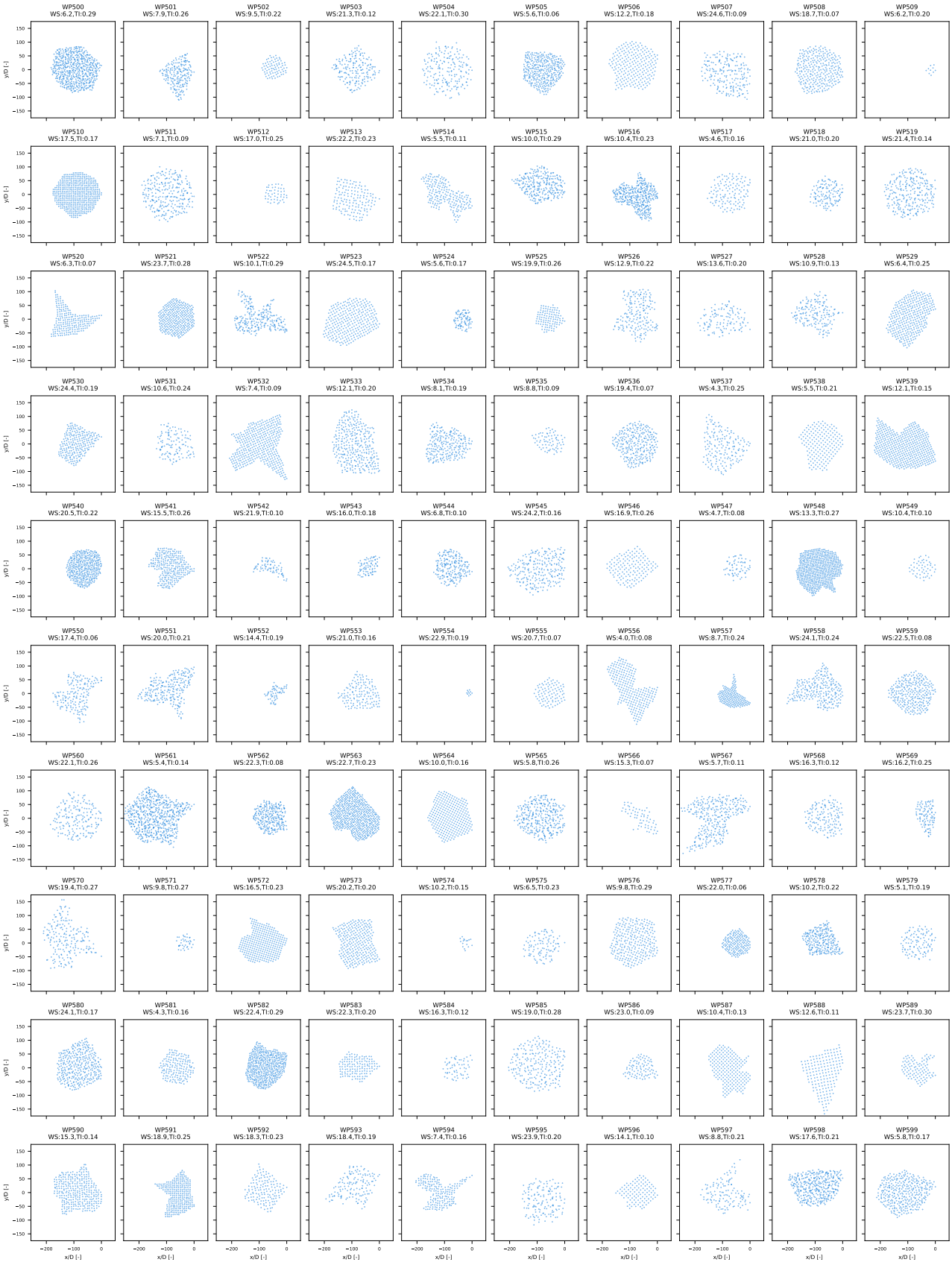


Figure D.6: 10x10 grid of wind park layouts (WP500-WP599) used for training. Wind speed (WS) and turbulence intensity (TI) are indicated for each park.

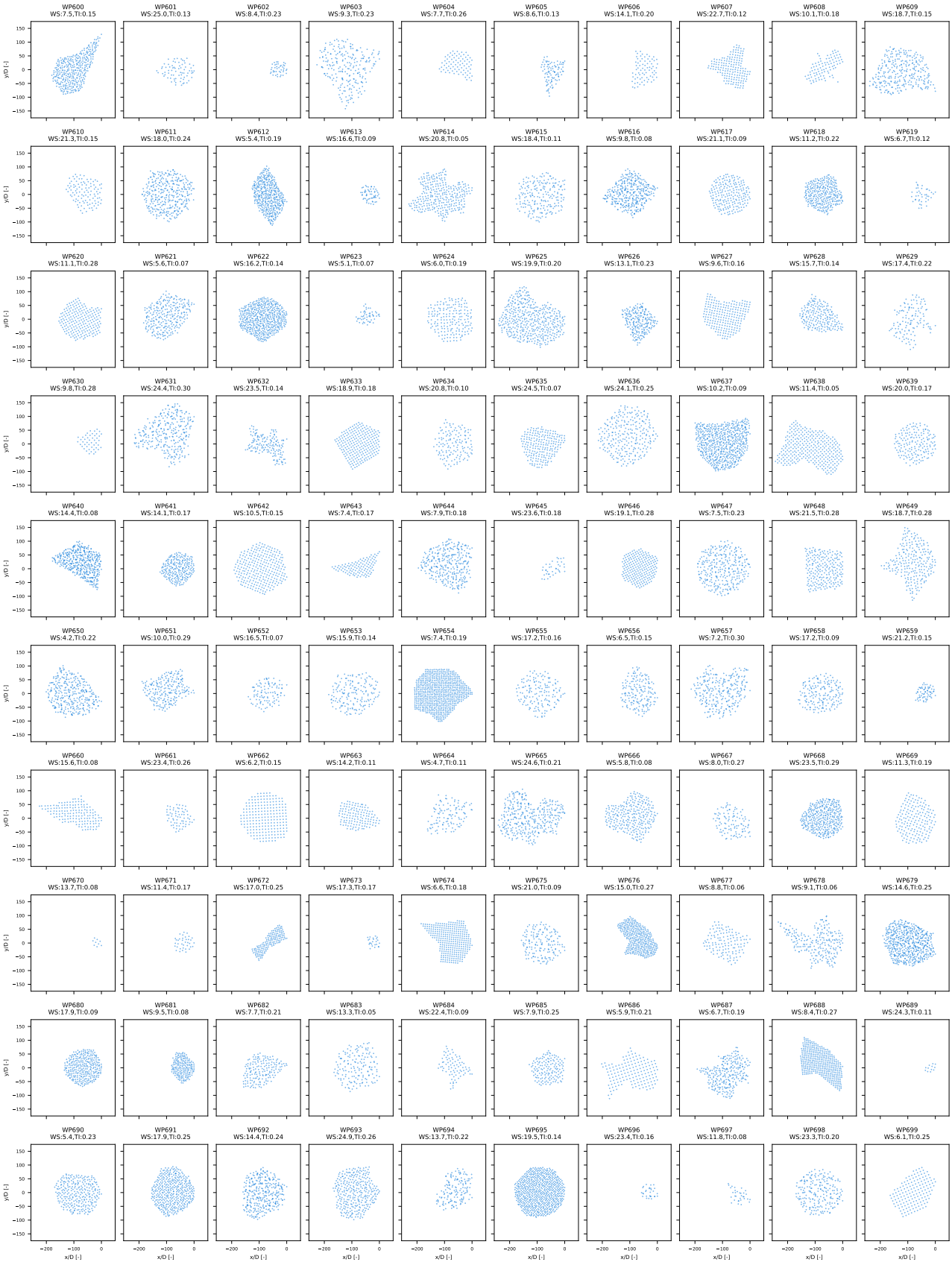


Figure D.7: 10x10 grid of wind park layouts (WP600-WP699) used for training. Wind speed (WS) and turbulence intensity (TI) are indicated for each park.

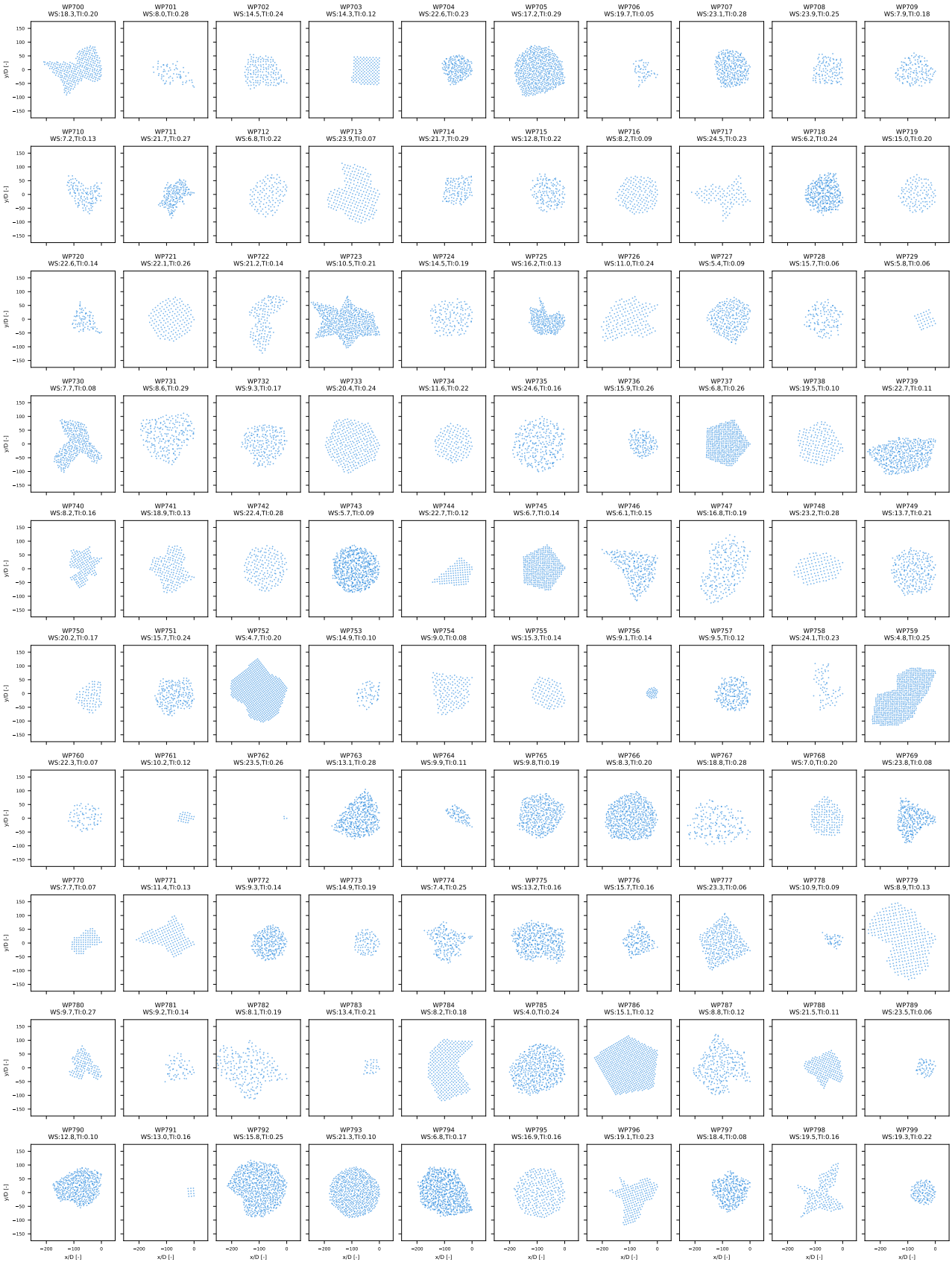


Figure D.8: 10x10 grid of wind park layouts (WP700-WP799) used for training. Wind speed (WS) and turbulence intensity (TI) are indicated for each park.

D.2 Test Dataset Wind Parks

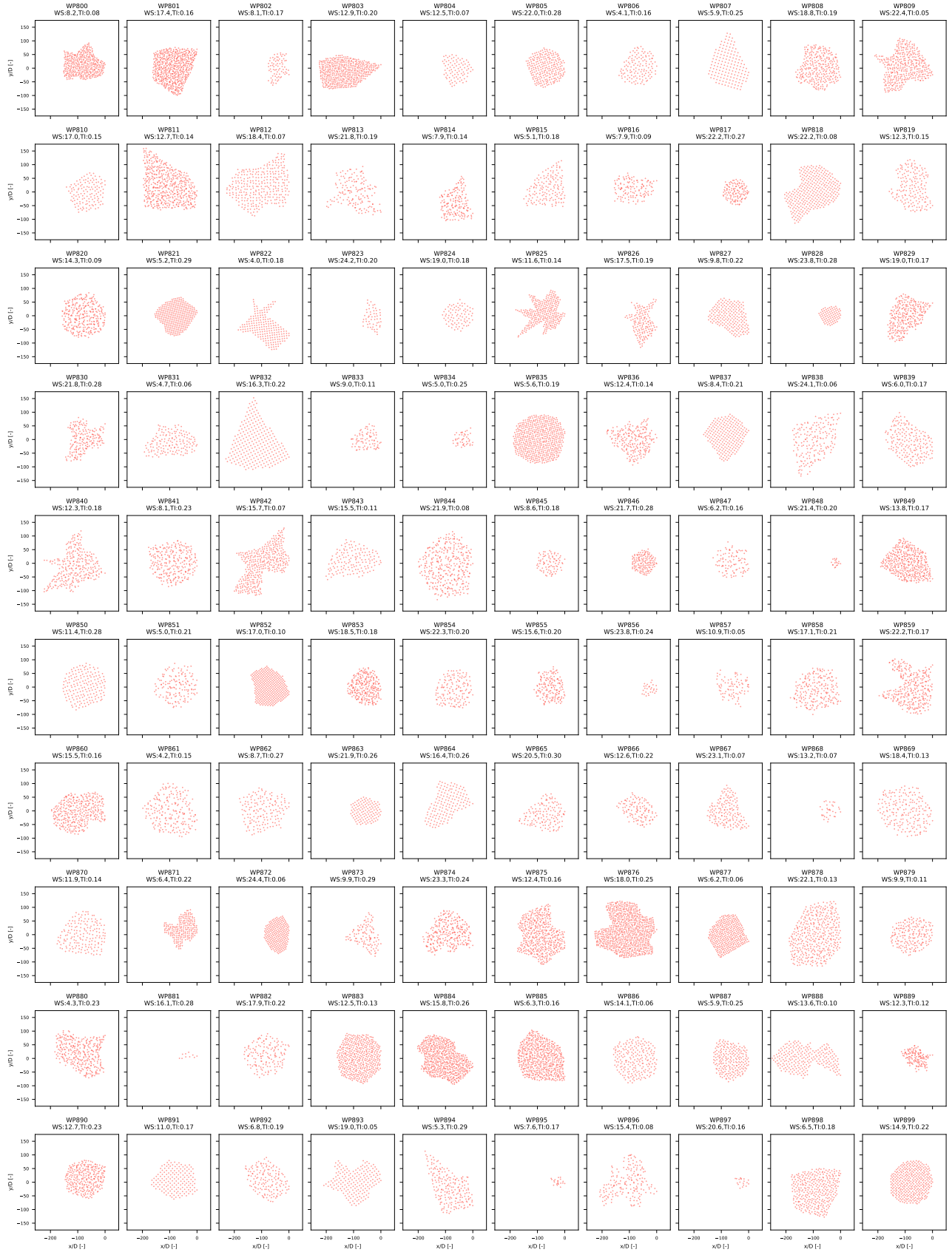


Figure D.9: 10x10 grid of wind park layouts (WP800-WP899) used for training. Wind speed (WS) and turbulence intensity (TI) are indicated for each park.

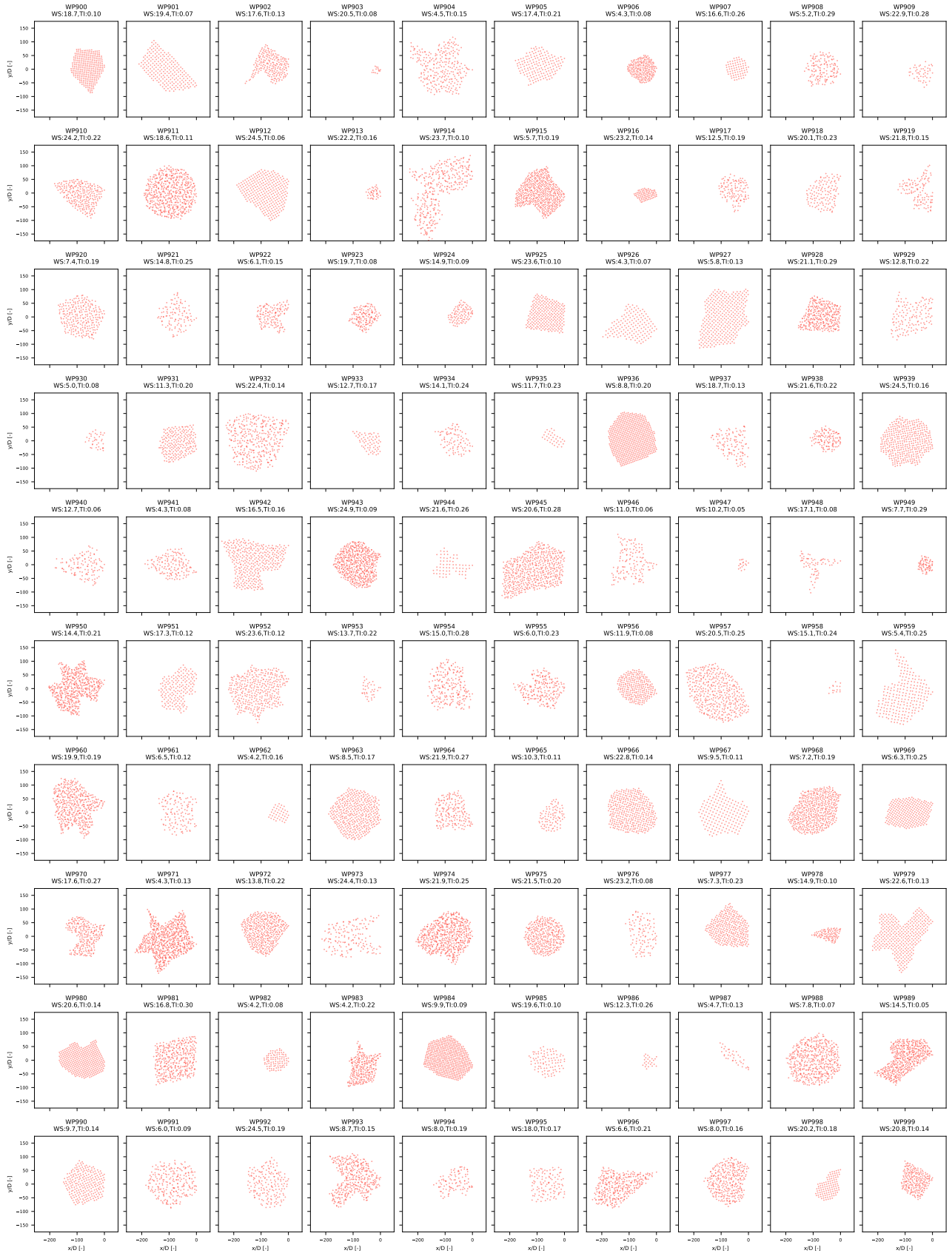


Figure D.10: 10x10 grid of wind park layouts (WP900-WP999) used for training. Wind speed (WS) and turbulence intensity (TI) are indicated for each park.