Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Eskild Gundersen Hansen

Quantifying performance variation in a deep learning model for wind turbine blade degradation

Master's thesis in Master of Science in Informatics
Supervisor: Özlem Özgöbek
Co-supervisor: Signe Riemer-Sørensen
June 2024

NTNU
Norwegian University of Science and Technology

SINTEF

Eskild Gundersen Hansen

Quantifying performance variation in a deep learning model for wind turbine blade degradation

Master's thesis in Master of Science in Informatics
Supervisor: Özlem Özgöbek
Co-supervisor: Signe Riemer-Sørensen
June 2024

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Performing maintenance on wind turbine blades at the right time is not an easy task. Material degradation is difficult to detect, and simulations that can estimate the degradation are difficult to tune and computationally too demanding to run in real time. The turbines are often situated in remote locations, so we only really want to perform maintenance on them when necessary.

Recently, a deep learning model has been developed to address part of this issue. It does this by predicting the forces acting on the turbine blades based on data we can normally measure using sensors mounted inside the turbine's nacelle. This model is trained on simulated data, but the goal is to eventually deploy it on real data. To do this we need to investigate how well it can be transferred to data that we know differs from the simulations. This thesis uses a simulated data set to investigate variations in the model's performance based on characteristics of the data it is given as input, and attempts to relate this to inherent differences within the data. The motivation of this is to be able to say something about how well a model will perform when deployed under conditions which are not fully represented in the training data.

We present our search for feasible candidate distance measures that can accurately represent the inherent differences in the data that cause the model to drop in performance. We find that wind turbulence around the turbine is so far the best candidate for such a measure. This is difficult, if impossible, to measure in practice. We therefore conclude that further exploration, and a stronger understanding of the data set, is needed to identify distance measures based entirely on quantities that are measurable in practice, and in the ideal case also be applied to other types of data.

Sammendrag

Det å utføre vedlikehold på vindturbinblader til riktig tid er ikke en enkel oppgave. Slitasje på materialet er vanskelig å oppdage, og simuleringer som kan estimere dette er vanskelige å innstille og for krevende til å kunne kjøres i samtid. Turbinene befinner seg ofte på fjerne områder, så vi vil gjerne bare utføre vedlikehold når det er nødvendig.

Nylig har en dyplæringsmodell blitt utviklet for å adressere deler av dette problemet. Den gjør dette ved å predikere kreftene som virker på turbinbladene basert på data vi vanligvis kan måle basert på sensorer som ligger inne i huset på turbinen. Denne modellen er trent på simulerte data, men målet er å til slutt sette den til verks på virkelige målinger. For å gjøre dette må vi undersøke hvor godt den kan overføres til data som vi vet er fravikende fra simulasjonene. Denne oppgaven bruker et simulert datasett til å undersøke variasjoner i modellens nøyaktighet basert på variasjoner i karakteristikkene til data den blir gitt som input, og forsøker å relatere dette til iboende forskjeller i dataene. Dette gjøres med motivasjonen av å kunne forutsi hvor godt en modell vil kunne operere når den settes til verks på turbiner under værforhold som ikke er fullstendig representerte i treningsdataene.

I denne oppgaven presenterer vi vårt søk for gode kandidater for avstandsmål som kan representere de iboende forskjellene i data som får modellen til å bli mindre nøyaktig i sine predikasjoner. Vi finner at vindturbulens rundt turbinen er så langt den beste kandidaten for et slikt mål. Dette er noe som er vanskelig, om ikke umulig, å måle i praksis. Vi konkluderer derfor at videre utforskning, og en sterkere forståelse av datasettet, er nødvendige for å identifisere avstandsmål som baserer seg kun på verdier som er målbare i praksis, og som i beste tenkelige scenario også kan anvendes på andre typer datasett.

Preface

Acknowledgements

This thesis is the final deliverable as part of my degree here at NTNU. The work has been supervised by Özlem Özgöbek, associate professor at the Department of Computer Science (IDI) at NTNU. The project was proposed and is done in cooperation with SINTEF, from which research leader Signe Riemer-Sørensen has provided additional supervision and assistance.

As such I would like to extend my deepest gratitude to both Özlem and Signe. Their support and guidance throughout the period of working on this project have been invaluable. Thank you so much, for everything you have done to get me to this point.

Note on reuse of material from the fall project

Parts of this thesis are fetched from or built upon parts of my delivery to the course IT3915 - Master in Informatics, Preparatory Project, named "Using deep learning as a surrogate for wind turbine simulations" [1]. Chapter 1 takes inspiration from chapter 1 in the project report, but is largely rewritten in the thesis. Chapter 2 has had some new additions moving into the thesis, namely a new introduction, the section on machine learning process, and the sections on data representation methods and statistical measures. Otherwise the contents of the chapter are complete copies from the fall report. Chapter 3 in the thesis is a complete copy from the fall report. The sections in chapter 4 are expansions upon the sections in chapter 4 in the fall report, the sections regarding data visualization and data preprocessing are new in the thesis, as well as the description of the turbine's internal components and the introduction to the chapter. In chapter 5 in the thesis, the experimental work that was done as part of the fall project is described with a highly abbreviated version of chapter 5 in the fall project. The code that was developed as part of the experiments in the thesis was partially developed during the fall project. The details of this transition can be found in chapter 5 in this thesis. The thesis' abstract is also an extension of the abstract from the fall project.

Contents

Figures

Chapter 1

Introduction

1.1 Motivation

Monitoring faults and scheduling maintenance on modern wind turbines is a complex task. The turbines are often situated in remote locations, making maintenance expensive. Norwegian weather conditions, especially in windy locations where turbines are placed, are quite diverse, making the need for maintenance difficult to predict universally. General practice for monitoring turbine conditions is today based on sensor data whenever possible. However, only a few turbines have sensors mounted on the blades themselves. To estimate the remaining life time of the wind turbine blades, one can either use this sensor data, or one can model the blade degradation based on sensor data from other parts of the turbine. To do the latter, one can use numerical simulations.

These simulations, while precise, carry with them a high cost in computational power. They model quite complex physical processes and calculate numerical values for a broad range of properties of the entire turbine. To initialize them, a specific input is needed to match the weather conditions at the location of the turbine, which is in itself challenging. Additionally, the physical measurements that are simulated are so interdependently that one cannot easily omit parts of the simulations, if one wanted to reduce computation cost by only focusing on the parts that were relevant for the purpose of modeling blade degradation. This project aims to estimate the degree of material degradation on the wind turbine blades caused by their vibrations. Since most turbines are equipped with sensors inside the nacelle, we investigate whether the degradation on the blades can be modelled using the data given by these sensors.

Herein lies the project's hypothesis: that a well trained machine learning model can be applied to predict material degradation on a range of different makes of wind turbines, and under a range of weather conditions. If this is the case, this model can be used as a replacement for the numerical simulations, which would both reduce computation cost, and simplify the process of making predictions for varying circumstances.

1.2 Problem definition

This master's thesis serves as a continuation of an ongoing research project at SINTEF where a deep learning model that trains on simulated data is already under development. A large data set consisting of several runs of the turbine simulations, as well as a pretrained version of the deep learning model in question, form the basis for this thesis' experimental work.

The concrete task of this project is to investigate the model's behavior on differing subsets of the dataset, and quantify aspects of the data set that cause the model to drop in performance when it is applied to a data subset it is not trained for. The model is retrained on a particular subset of the data, such that its predictive performance on data that is similar to the training data can serve as a baseline. From that baseline, its performance on less similar data is analyzed. To quantify the similarity between data subsets, some metrics for measuring the difference between datasets' distribution are introduced. These are compared to the drop in performance to see whether they can serve as an indication of how much worse the model will perform given the measured distance between the two subsets.

The justification for this is the commonly arising problem in machine learning with having insufficient data to properly train a model for a specific task. The end goal is to train a model that can accurately predict material degradation on wind turbine blades under a wide range of weather conditions, including harsher ones than where turbines are normally located today. To do this we need appropriate amounts of data that can train this model. Using transfer learning we can reduce the amount of data that is necessary, by transferring a model that has been trained on a somewhat similar data set. However, to understand how much data is needed to adequately fine-tune this model, we seek a way to understand what aspects of the data causes the model to drop in performance, and to what degree. That way, one can measure how much data is needed based on how much the new data set is different from the training data of the model, and compare that to how fast the model adapts when given a certain amount of new data to train on. Alternatively we can judge whether the real data the model is to be applied on is different enough to warrant the need for transfer learning in the first place. Additionally, this distance measure can be used as part of a transfer learning implementation in the loss function to further accelerate the model's rate of adaptation. We use a simulated data set and a model trained on it to develop a method that can inform how much data gathering is required in order to transfer the model to real sensor data.

1.3 Research questions

The thesis answers the following research questions:

- **RQ1:** How can transfer learning be applied to time series data for wind turbines?

- **RQ2:** How is it feasible to measure the distance between sets of multivariate time series?
- **RQ3:** How does the error in wind turbine models change after introducing transfer learning methods?
- **RQ4:** How does the change in error after introducing transfer learning relate to the data sets?
- **RQ5:** How can our knowledge about transfer learning in our domain be applied in other domains?

1.4 Thesis outline

Chapter 2 gives some theoretical grounding needed for understanding the project's topic. Chapter 3 gives a breakdown of the related work that was found in the field, which will give a context for the research and clarify its contributions. Chapter 4 elaborates on the dataset to be used for this task, as well as an introduction to the physics needed to understand them. The data preprocessing steps are also explained in this chapter. Chapter 5 details the experimental work undertaken as part of this thesis, from the retraining of the deep learning model, to the production of results. Chapter 6 presents the results, and provides some immediate commentary as needed. Finally, chapter 7 brings everything together into context with the literature to discuss the ramifications of the research and answer the research questions.

Chapter 2

Theoretical background

This chapter is designed to supply the reader with a reasonable understanding of the technical and mathematical aspects of the thesis. It is assumed that the reader is familiar with the inner workings of a regular feed-forward neural network. If the reader is not familiar with the subject, we refer to Goodfellow et al. [2] for a more thorough introduction. Regarding the mathematics, we assume that the reader has a solid foundation in calculus, and is familiar with the concept of probability distributions and probability density functions.

2.1 Machine learning process

The process of designing, developing and deploying a machine learning solution is often described as a multi-step, cyclical procedure. Different sources will disagree on the exact number of steps, but certain key distinctions between these steps remain constant throughout them, namely the distinction between data gathering, data refining, model selection, and model engineering [3]. The work presented in this thesis has followed this step-wise process, and the work relating to the different steps will be presented in different parts of the thesis. For data gathering and data refining see chapter 4, for model selection and training see chapter 5, and for discussions about model engineering see chapters 6 and 7.

2.2 Deep learning

Deep neural networks have gradually become more and more sophisticated through the recent years, and have recently started being used on time series data. This section will explain two of the most commonly used deep learning methods for time series regression.

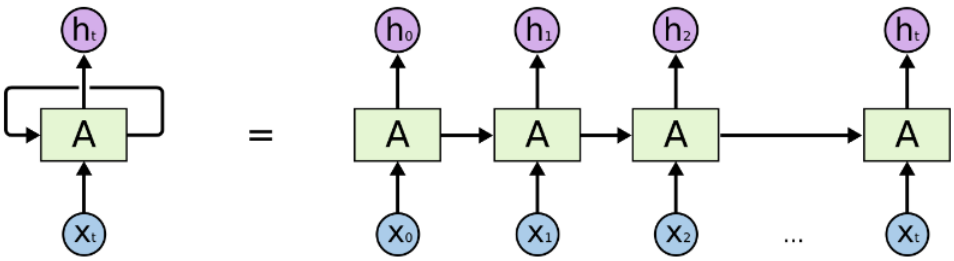


Figure 2.1: Diagram of an unrolled recurrent neural network [5].

2.2.1 Recurrent neural networks

Recurrent neural networks (RNNs) are a variation of neural networks that enable the learning of temporal dependencies in sequential data. This is achieved by creating multiple copies of a network, where each one inherits a value from a precursor (see figure 2.1). In each network, the input vector and the inherited vector from the previous network are assigned trainable weights. The output of the network is then passed to the next one. During back-propagation, the weights are updated backwards through time steps. If each input is a single measurement from a time series, then the RNN will learn temporal dependencies within this time series.

RNNs struggle with a phenomenon called the vanishing gradient problem. This problem is also present in deep neural networks with too many layers [4]. When propagating the gradient loss over many layers, the gradient becomes vanishingly small as the number of layers increases, due to how the weights in the network are applied multiplicatively to subsequent values. In RNNs, this applies to the weights applied to values from previous time steps. This makes it difficult to model long-term dependencies in the data, since the information contained in the gradients gets lost along the way.

2.2.2 Long short-term memory

In recent times, a handful of variations to RNNs have been made in an effort to deal with the vanishing gradient problem. So far, the most popular one has been long short-term memory (LSTM) [4]. Where the inherited values between modules can be thought of as the network's working memory, LSTM incorporates something called a cell state, which emulates a sort of long-term memory between modules.

There are several different versions of the LSTM architecture, but this explanation will follow the basic structure presented in Christopher Olah's blogpost on the topic [5]. For a visualization of the structure, refer to figure 2.2.

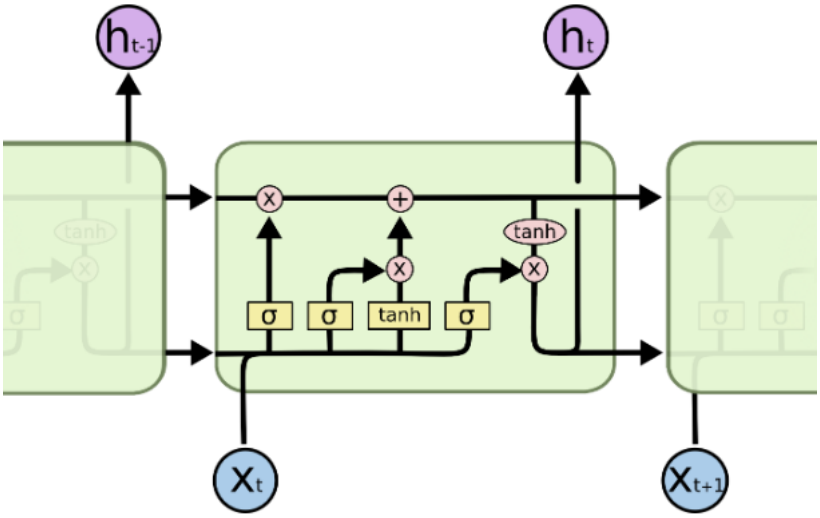


Figure 2.2: A diagram of the internal components of a repeating LSTM module. The module contains four neural network layers, here denoted with yellow boxes. The red ellipses are pointwise operations [5].

The core of the LSTM architecture is the cell state, in the diagram represented by the horizontal line running across the top of each module. This is only ever updated using two different linear operations, meaning that the values stored there can easily pass from one module to the next, until it is time to update them. The rest of the architecture is divided into three gates, which use the inherited working memory concatenated with the input value, and operate as follows:

- **The forget gate** consists of a single weighted layer with a sigmoid activation function, whose output is multiplied pointwise with the previous cell state. This determines which parts of the previous state to keep, and which to reduce or erase entirely.
- **The input gate** consists of two layers, one with a sigmoid and one with a hyperbolic tangent activation function. The outputs of these are multiplied together and added to the cell state. This determines which parts of the input, and how much, is to be stored in the new cell state.
- **The output gate** consists of a single sigmoid-activated layer. The result value from this is multiplied with a normalized (tanh) version of the cell state. This determines what the module outputs as working memory to the next module, and as output value for this time step.

As for RNNs, the weights inside the different layers are shared between modules, and are updated backwards through time steps using a gradient loss function [4].

2.3 Transfer learning

Machine learning generally operates on a principle that the data a model is being trained on more or less accurately represents the data it is to make predictions for. We say that the training data and the test or prediction data should follow the same distribution. By distributions we refer to the theoretical underlying statistical distributions that the data is drawn from, or the empirical distributions shown in the data. We do not know what these underlying distributions are most of the time, but we can estimate them using empirical data. In reality the prediction data may deviate somewhat from the training data, which will often cause a trained model to become less accurate. Sometimes this loss in accuracy is within a certain acceptable margin, and other times some measures are required to raise the model back to acceptable performance levels. This is where transfer learning comes in.

Transfer learning, as described by Weber et al. [6], is a technique often applied to deep learning methods to adapt models to different tasks and datasets. The idea is to keep knowledge gained during training, and reuse it for something different. This is a more intuitive process in deep neural networks, due their layered structure. For example, in a computer vision model trained to classify images of cats and dogs, one can reuse the layers that learned to represent animals in the image in a regression model that tries to estimate the size of the observed animal.

A more formal definition can be given as follows:

Definition: Given a source domain D_s with related task T_s, and a target domain D_t with related task T_t, transfer learning seeks to improve the learning of the target predictive function $f_t(\cdot)$ using knowledge gained from D_s and T_s when the domains or tasks are dissimilar between source and target [6].

From this definition one can separate transfer learning approaches into two categories: domain adaptation and task adaptation, based on whether the domains or the tasks are different. The previous example with the cat and dog classifier would be a case of task adaptation, since the task changed from animal classification to size regression. In this project, the application of transfer learning is a case of domain adaptation. The task stays the same: we want to predict material degradation on the turbine blades over time. The domain changes when the model is applied to a subset of data with a different distribution.

Notice also that this definition allows for several interpretations of how transfer learning can be applied. In the simplest case, the model is reused in its entirety on the target domain or task. A more common approach in deep learning is something called fine-tuning. This is done by freezing the weights of some of the layers, and letting only the remaining weights be updated during retraining. More advanced domain adaptation methods will also employ a distance measure between the source and target domain's distributions as part of the optimization objective, so that the discrepancies between the two data sets are accounted for during fine-tuning [6]. This technique is most often referred to as distribution alignment in the literature. The most commonly applied distance measures in this way are maximum mean discrepancy (MMD) or Kullback-Liebler divergence.

2.4 Distance measures for data distributions

To evaluate how different a pair of distributions are to each other, we need to define one or more distance measures that can be used on data distributions. For this project, the n-dimensional Kolmogorov-Smirnov test statistic and the Wasserstein distance were chosen, due to their reported performance both in efficiency and accuracy [7] [8]. This section will explain these two measures, the difficulty in calculating them in higher dimensions, and how this can be solved with modern approximation methods.

2.4.1 n-Dimensional Kolmogorov-Smirnov statistic

The one-dimensional KS statistic

The Kolmogorov-Smirnov test is a commonly used test for determining the similarity between two distributions. It is used due to the fact that it is non-parametric, meaning that is assumes nothing about the nature of the sample distribution, and that it is sensitive to all ways in which distributions can be dissimilar [8]. We separate between the one-sample and two-sample versions of the test, where the one-sample version compares the empirical distribution function (EDF) of a sample to a hypothesized underlying cumulative distribution function (CDF), and the two-sample version compares the EDFs of two different samples. We will mainly be concerned with the two-sample version. This test statistic is calculated as follows:

$$D_{n,m} = \sup_x |F_{1,n}(x) - F_{2,m}(x)| \tag{2.1}$$

Here $F_{1,n}$ and $F_{2,m}$ denote the EDFs of two samples with n and m elements respectively, and sup is the supremum function. In the test, this statistic is used to discern whether the samples are produced from the same underlying distribution. If the KS-statistic reaches or exceeds some critical value, we conclude that the underlying distributions are dissimilar from each other. However, due to its general applicability, the test does not give any insight into how these distributions are dissimilar from each other.

Extending to multiple dimensions

Several attempts have been made in an effort towards extending this test to multivariate distributions. The main problem with this is that there is no single way to order the data points of a multivariate sample in such a way that an EDF is uniquely defined [9]. However, an implementation by Hagen et al. [8] of an early proposal for an alternative calculation method for these EDFs shows promising results.

2.4.2 Wasserstein Distance

The Wasserstein distance between distributions, sometimes referred to as the earth mover's distance, is highly related to the optimal transport problem. Intuitively, if we were to imagine the probability distributions as a physical mass, the Wasserstein distance measures the amount of work needed to move mass in the first distribution in order to transform it into the second.

In order to express this concept mathematically, we introduce the coupling of two probability distributions $\mu(x)$ and $\nu(y)$, denoted $\gamma(x, y)$, where x and y are elements of the event spaces of μ and ν respectively. The coupling is a joint probability distribution whose marginal distributions correspond to those of its constituent distributions, in other words, it must fulfill the following properties:

$$\int \gamma(x, y)\, dy = \mu(x), \int \gamma(x, y)\, dx = \nu(y) \tag{2.2}$$

With respect to the previous analogy of moving mass, the coupling represents a transport plan from one distribution to the other. Given x and y, $\gamma(x, y)\, dx\, dy$ states how much mass is to be moved from point x to point y. The conditions in 2.2 then translate to the fact that the amounts of mass to be moved to and from the distributions need to be equal to the total amount of mass in the distributions. All of the total mass needs to be accounted for in the transport plan. It is important to note that this plan is not uniquely defined for each pair of distributions. The goal of the Wasserstein metric is then to find the optimal transport plan that minimizes the amount of work needed to perform the transportation. To define the amount of work, we define a cost function associated with moving mass from x to y, denoted $c(x, y)$. Denoting Γ as the space of all valid transport plans, the total cost of performing the optimal plan is then given as

$$C = \inf_{\gamma \in \Gamma(\mu, \nu)} \int c(x, y)\, d\gamma(x, y) \tag{2.3}$$

where inf is the infimum function. Given that x and y both are elements of event spaces where it makes sense to define a distance metric d between the points, it is common to simply use this metric as the cost function. It is also common to introduce an order parameter p to the Wasserstein metric. All of this put together gives rise to the following formula:

$$W_p(\mu, \nu) = (\inf_{\gamma \in \Gamma(\mu, \nu)} \int d(x, y)^p\, d\gamma(x, y))^{1/p} \tag{2.4}$$

Complexity in higher dimensions

The Wasserstein distance formula can be reduced to closed form expression in two cases: on gaussian distributions and univariate distributions [7]. The distributions we are dealing with are non-gaussian and multivariate, so the distance will need to be estimated. Otherwise, we would need to search through the whole space of

transport plans in order to find the optimal one. An approximation method that takes advantage of these two closed forms are presented by Nadjahi et al. [7], and will be applied in this project.

2.5 Data representation methods

We use the term representation methods to refer to ways to transform data sets into feature representations or visualizations of a lower dimension than that of the data set. Here we briefly describe t-distributed stochastic neighbor embedding (t-SNE), which was introduced to the project towards the end, as an attempt to identify a lower-dimensional representation of the data based on inherent distributions.

2.5.1 t-SNE

t-SNE uses Kullback-Liebler divergence as an optimization objective. This is a commonly used distance measure between probability distributions, the calculation of which we treat as a black box for the purpose of this thesis.

t-distributed stochastic neighbor embedding (t-SNE) is foremost a visualization technique that is commonly applied to high-dimensional data. A probability distribution of the data set is constructed in such a way that similar entries are given a higher joint probability than dissimilar points. The points are then mapped to a lower-dimensional space, usually two or three dimensions, and a similar distribution is constructed for this mapping. t-SNE then minimizes the KL divergence between these two distributions given the locations in the lower-dimensional space. As such, similar entries in the data set are clustered together in a visually interpretable space [10].

2.6 Statistical measures

For the purpose of distilling the data set into smaller representations, a set of statistical measures are taken. A full detailing of this process is found in section 5.2.2. The four chosen statistics are mean, standard deviation, kurtosis and skewness. We assume the former two to be well enough known that an explanation here is unnecessary.

Kurtosis is a measure of the shape of the distribution. Essentially it describes how often its outliers occur, or how "flat" or "sharp" the curve of the distribution is. Skewness is a measure of asymmetry about the mean in the distribution. Put simply, a positive skew says that the right tail is longer, and the central mass of the distribution is "skewed" to the left, and vice versa for negative skew.

Chapter 3

Related work

This chapter will discuss the findings of a literature review with a narrow focus. The goal of is both to present relevant literature that can be used as inspiration for this project, and to provide the context that this project is conducted within. Through these literature summaries, I will assert the knowledge gap that this project aims to fill.

3.1 Review method

In order to identify which papers have contributed to the same niche as this project, an online literature search was conducted. The search engines used were IEEE Xplore and Web of Science. The search was kept narrow, so as to limit the amount of articles that needed to be filtered. The queries were designed in such a way as to capture a very specific domain. Since the task to be solved in this project is so specific, a broader search would only help to include irrelevant papers. For the wind power domain specific applications, the term "wind turbines" was combined with "transfer learning" and "LSTM". On IEEE Xplore this was changed to the combination of "transfer learning", "wind turbines" and "deep learning", due to the low amount of returned results regarding LSTM models. For other studies in time series transfer learning, the terms "transfer learning", "LSTM", and "forecast" were used. The results from these searches are presented in the following sections.

For each of these searches, a selection procedure was followed to reduce the amount of papers to present here, and to reduce the result to only the papers that were truly relevant to this project. Papers were excluded according to the following criteria:

- **EC1:** The abstract does not detail the way in which transfer learning was used.
- **EC2:** The study does not include the use of domain adaptation.
- **EC3:** The study does not explain its relevance to all parts of the search term in the abstract, implying only a slight relevance.

- **EC4:** The study has been superseded by a new study by the same authors.

After this procedure, an initial collection of 25 papers in the wind power domain and 40 on forecasting were reduced to ten and nine respectively.

3.2 Transfer learning with wind power data

Over the last handful of years, several advances have been made in the pursuit of applying transfer learning based methods for wind turbine data, although almost exclusively on the tasks of fault detection and fault diagnosis. The majority of the papers that were included after the selection process present some sort of transfer method between distributions of data, and applies it to wind turbine fault detection data from several different wind farms [11] [12] [13] [14] [15]. Zhu et al. [16], Zhang et al. [17], and Yue et al [18] also use LSTM as a part of their model, which is of particular interest to this project. However, all of these papers design models for fault diagnosis, and defines this as a classification problem, either binary (fault or no fault), or multi-class in the cases where they want to determine which fault has occurred in the turbine. This is opposite to the aim of this project, which is to perform regression and quantify performance after model transfer. Shao et al. [19] present a very recent study that designs a graph-based recurrence modelling scheme to represent temporal dependence in the data, and feeds this into a CNN based classifier. Finally, Sun et al. [20] present a review of methods used in wind turbine fault diagnosis. From this we see a notable lack of research done in forecasting and regression problems related to wind turbine data.

3.3 Transfer learning for time series forecasting

Before conducting the literature search, a mapping study from early 2021 [6] was found that details comprehensively the studies done on applying transfer learning with time series data. The studies found in this mapping study are grouped according to their application domains and what particular method of transfer learning was applied. The main application domains up until that point were in fault diagnosis and human activity recognition. The particular findings in the field of fault diagnosis is that this was either addressed in the form of an anomaly detection problem (binary classification) or a fault classification problem.

Due to this mapping study's comprehensiveness, the search for literature on this topic was restricted to results from 2021 and later. Among the studies kept after the selection process there is an evident focus on applying various transfer learning strategies on a broad range of application domains where data is in the form of time series. From IEEE Xplore, all the kept studies were examples of this kind of study. The applications of transfer learning were done with three main purposes in mind: either to overcome the issue of lacking training data for a highly

specific task or domain, to reduce the overall training time, or to increase the general accuracy of a model from that of an approach without transfer learning involved. Concretely, the studies applied transfer learning to the following domains and tasks: building load forecasting [21] [22] [23], crop plant yield forecasting [24] [25], battery remaining useful life prediction [26], lightpath signal-to-noise ratio prediction [27], crime rate forecasting [28], and volcanic activity forecasting [29].

This project stands out from most of these studies in that it does not seek to optimize the machine learning model directly. We neither look to reduce training time or increase the model's predictive accuracy. Instead, the purpose of the project is to understand how the predictive accuracy of the model is affected by the properties of the data it is being asked to make predictions for. To do this we simply accept its current performance as a baseline, then we measure its performance on a variety of data. Additionally we want to measure how much data is needed to fine-tune a model towards a certain level of accepted accuracy. For this we will use simulated data, which is readily available from SINTEF's earlier runs of the simulation. Nevertheless, the methods used in the papers found here will likely be of great use moving further in the project.

From this search, we conclude that there is a research gap both with regards to time series forecasting and with applications with wind turbine data.

Chapter 4

Data and task domain

This chapter details the data set and gives a brief introduction to the domain of wind turbine physics necessary to interpret the data. The simulations from which the data files are extracted are introduced, but are treated as a black box for the sake of this project. Due to the size of the data set, only the variables that are used for the machine learning task are detailed here. Finally, a description is given of how this data is preprocessed before being used in the neural network.

4.1 Data set

4.1.1 Wind turbine inner components

To contextualize the variables in the data set, a brief explanation of the inner workings of the turbine is given here. A visualization of the interior of the nacelle is given in figure 4.1. The rotor consists of three blades and a hub. The blades' pitch angle can be adjusted to accommodate for changing wind speeds, and will control how much of the wind energy is converted to rotational motion in the rotor. This is done by a controller at the back of the nacelle which measures the wind speed and direction at any given time. In addition, the controller controls the yaw drive, which can swivel the turbine to face the direction of the wind. The hub of the rotor is connected to the low speed shaft which transfers the rotational motion from the rotor to the gearbox. The gearbox increases the rotational velocity and sends it through the high speed shaft, which in turn is connected to the power generator.

4.1.2 Simulations

The data set is produced from a range of simulations of a wind turbine under different weather conditions. The simulations are kindly provided by DNV by running the Bladed simulator [31]. The simulations themselves are property of DNV and

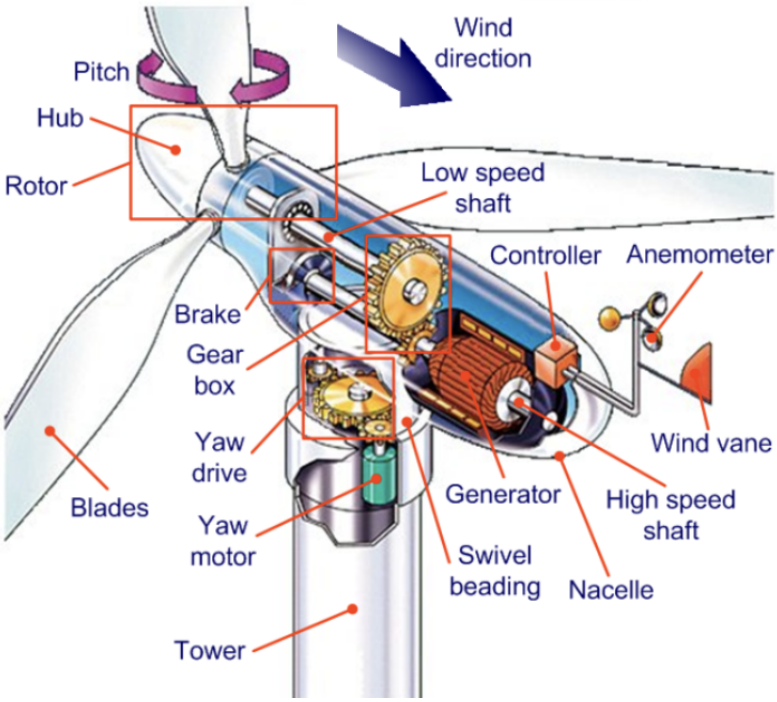


Figure 4.1: View of the internal components of a common, modern wind turbine [30].

cannot be shared, but analysis results will be shown. The simulations are initiated with a set of parameters. Each simulation generates ten minutes of data stored in a new file, which allows for easy separation of the data produced by a certain initialization. The entire turbine is simulated throughout, modeling intricate physical relationships between the different components, resulting in a table with 218 variables with values for 30,000 time steps (each time step is 0.02 seconds). For this research, SINTEF has provided 897 of these data files from within their system, with file names based on the values for the initialization parameters.

The initialization parameters that the data files are named after are aspects of the weather conditions around the simulated wind turbine. The parameters are as follows:

- **U - Wind speed** (m/s) - The average wind speed around the turbine.
- **TI - Turbulence intensity** (%) - The ratio between the standard deviation of the wind speed and its mean. Gives an indication of how much the wind speed, and wind direction, changes over the simulation period.
- **D - Air density** (kg/m^3) - The mean air density around the turbine.
- **SH - Shear** $(\frac{m}{s}) \cdot m = s^{-1}$ - How much the wind speed changes by altitude. Measured as a difference in wind speed to the wind speed at a certain reference point at ground level, divided by the altitude above that reference point.
- **DIR - Wind direction** $(radians)$ - The mean angle of the wind direction relative to north.

Additionally, the simulations are initialized with a randomized set of initial values. Among the files used in this research, three different seeds were used as part of this randomization. In this way, the data set consists of triplets of data files that have identical weather parameters. An example of a data file name would be "U7.4_TI8.71_D1.09_SH0.52_DIR-0.04_1", which would contain the results of the first of three simulations that were done with those values for the weather parameters.

4.1.3 Features

Target variable

The motion of the blades is represented in the simulations by a series of numerical values. Of interest to us is the torque exerted onto the blades at a certain distance away from the rotor, mainly in the direction perpendicular to the plane of rotor rotation. In figure 4.2 this is represented by the vector **n**.

To understand this choice of target variables, consider the following analogy with a paper clip: If you unwind a paper clip and bend it back and forth in the same direction, you expect it to eventually snap. This is due to subsurface cracks that are formed in the metal when it is forced to change shape. These cracks grow larger as more stress is put on the material, until they eventually reach the surface. This will also happen on the wind turbine blades if they are forced to bend by the

wind blowing on them. The blades are designed in such a way that they convert the movement of the wind into rotational motion very efficiently, but not all of it. The remainder of that energy causes the blades to be rocked back and forth relative to the rotor, and hence creates stress in the material.

Subsurface cracks are very difficult to detect. Therefore, by regressing how much and how intensely the blades have been put under stress over a certain time period, we can estimate the total amount of stress the blades have been exerted to. In turn this lets us estimate the remaining life time of the blades, and judge whether maintenance or replacement is necessary before a given time.

Feature selection

Recall that sensor data is typically only available for measurements of the internal components of a wind turbine mounted inside the nacelle. For this reason, a deep learning model that will eventually be deployed to deliver predictions based on live sensor data must only be dependent on data from these internal components. From the 218 variables in the simulation data files, only the ones relating to the nacelle components were kept, and further filtered away until only the variables with the strongest relation to the target variable remained. This resulted in a total of nine variables to be used as features.

Features and their physical properties

The model uses the following variables from the simulations as features. For a visualization of the relevant turbine components, see figure 4.1.

- **Rotor speed** ($radians/s$) - Rotational speed of the rotor.
- **Generator speed** ($radians/s$) - Rotational speed of the power generator.
- **Blade 1 pitch angle** ($radians$) - The pitch angle of one of the blades. This is assumed to be identical for all three blades.
- **Blade 1 controller demanded pitch rate** ($radians/s$) - The rate at which the controller is demanding the blade to change its pitch angle.
- **HSS torque** ($Newton \cdot meters$) - The torque exerted onto the high speed shaft by the gearbox.
- **Electrical power** ($Watts$) - Generated electrical power output of the turbine.
- **Nacelle fore-aft acceleration** (m/s^2) - The acceleration of the entire nacelle, either towards or away from the direction the rotor is facing.
- **Nacelle side-side acceleration** (m/s^2) - Horizontal acceleration of the nacelle, perpendicular to the direction the rotor is facing.
- **Rotor azimuth angle** ($radians$) - The angle of rotation at the rotor, essentially how far along the blades are in their current rotation, denoted ψ in figure 4.2.

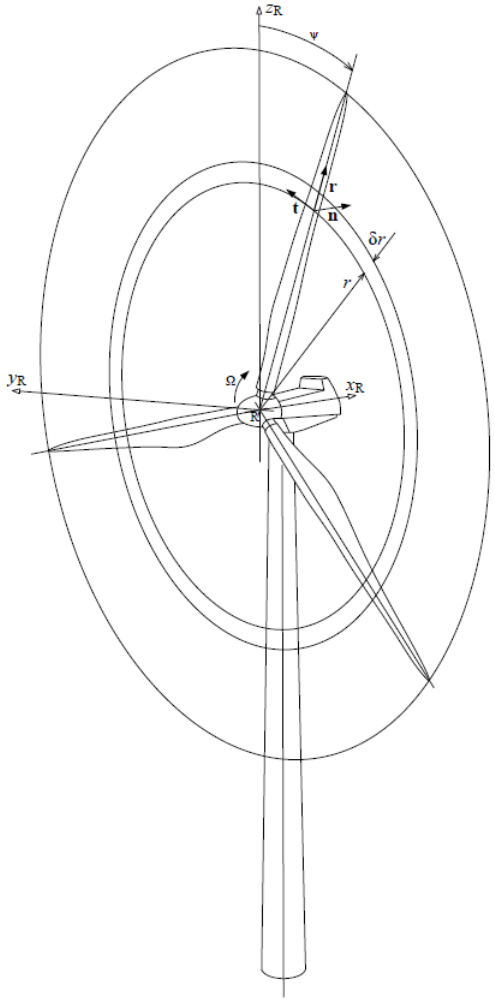


Figure 4.2: Geometry of wind turbine rotor including a rotor and blade coordinate system [32]

Data example and visualization

Figure 4.3 and figure 4.4 show visualizations of the variables from the data set that are used in the deep learning model. These are snippets from the data files named "U10.5_TI5.86_D1.09_SH0.42_DIR0.01_1" and "U7.4_TI19.67_D1.07_SH0.09_DIR-0.12_1" respectively. Within the data set, these files are representations of low and high turbulence intensities respectively. Additional visualizations can be found in appendix A. Due to the proprietary nature of the data set, only snippets can be shown. From these graphs we see two major takeaways. The first, which the graphs in appendix A confirm, is that most of the variables, especially the target variable, become much more erratic in their oscillations when turbulence intensity increases. The second observation is that the blades' pitch angle only reaches non-zero values in a certain portion of the data files. This is due to the fact that the controller only pitches the blades when there are sufficient wind speeds. When the blades are pitched, the rest of the system changes in response, as can be seen in figure 4.3, where the blades start pitching after the 6000th time step.

4.2 Data preprocessing

Apart from the feature selection described earlier in this chapter, two additional data preprocessing steps were implemented alongside the deep learning model. A detailed description of the model itself is given in section 5.2.1.

The first preprocessing step can be described as a form of data augmentation, similar to how image cropping is often implemented in computer vision tasks. The model takes 500 lines of data as input at a time, corresponding to ten seconds of simulation runtime. Instead of segmenting the 30,000 lines in the files into 60 chunks of 500 lines each, a random offset of 0 to 500 lines is added to the start of each file, such that the first chunk begins at this offset. Each following chunk then begins where the previous one ended. The resulting unused lines at the beginning and end of each file are not used in training. This data augmentation is applied due to the fact that all of the data files are generated with one of three potential random seeds, and therefore show similar patterns in their initial phases due to having similar initial conditions.

After the data augmentation a standard scaler is applied. This means that the mean and the standard deviation are calculated for each variable, across the entire set that remains after the augmentation. The corresponding mean is then subtracted from each variable's value, before dividing by the standard deviation. If the data is normally distributed, this brings the distribution closer to that of the standard normal distribution. This helps ensure that the variables are similarly scaled. This scaling is applied separately to the training data before training and to any input data before the model makes predictions.

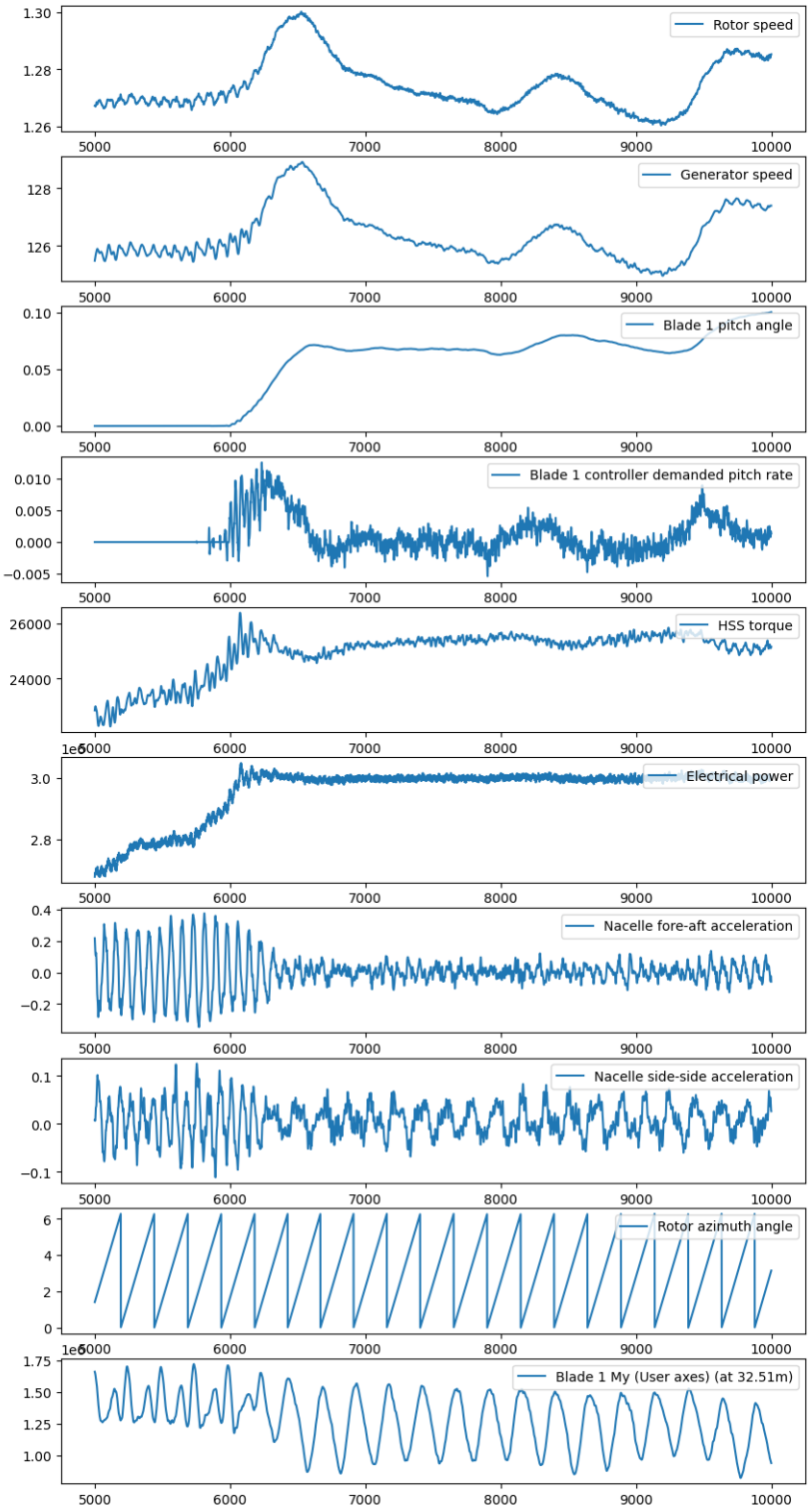


Figure 4.3: Example of input data for the deep learning model. Here we show time step 5000-10000 (100-200 s), for a simulation with initial conditions U:10.5, TI:5.86, D:1.09, SH:0.42, DIR:0.01. In the data set, this corresponds to average wind speed and low turbulence.

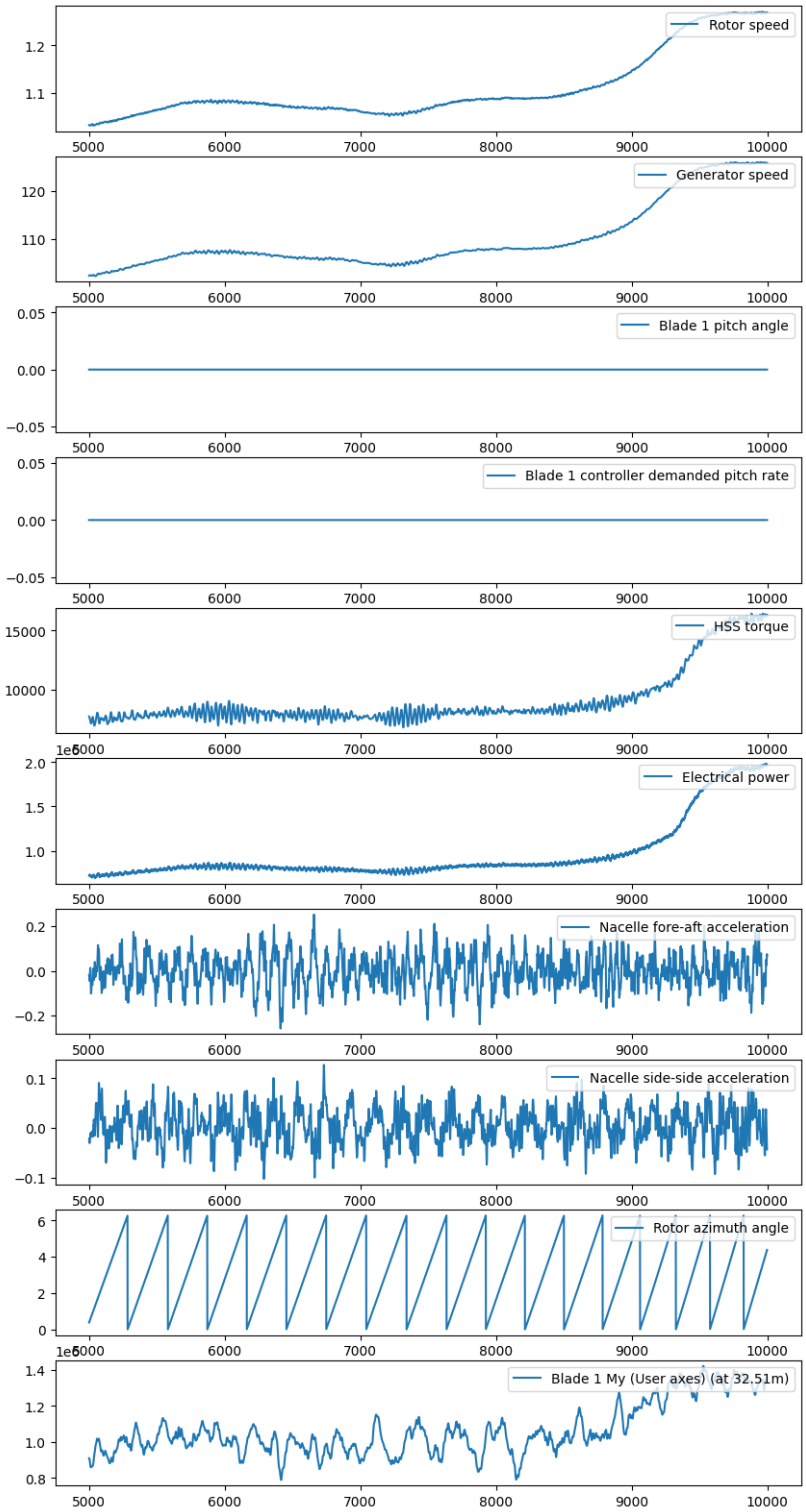


Figure 4.4: Example of input data for the deep learning model. Here we show time step 5000-10000 (100-200 s), for a simulation with initial conditions U:7.4, TI:19.67, D:1.07, SH:0.09, DIR:-0.12. In the data set, this corresponds to low wind speed and high turbulence.

Chapter 5

Experiments

Given a model trained on simulations, the aim is to quantify the uncertainty on new input data which may deviate from the training data. To do this, we first re-train the model on a homogenous part of the data, and then quantify performance on other parts of the data. We calculate values for various distance measures between the different parts of the data, with the intention of using these as candidates for measuring the uncertainty in the model when using the new parts of the data as input.

This chapter describes the process of setting up the experimental work on a remote cluster and splitting the data set. Later it describes retraining the model on one of the resulting parts of the data, along with a description of the model's architecture. Finally it describes the method of measuring distance between the data subsets, and how this is executed alongside applying the model on the different data subsets.

5.1 Code base and set up

The code base for the model architecture and training was provided by Katarzyna Michalowska through the Center for Research-based Innovation NorwAI. This code base is almost completely Python-based, using Pandas for data extraction and manipulation, and Tensorflow for deep learning implementations. The experiments described in this chapter were conducted with that code base as a foundation, with permission from the developers. For the distributional distance calculations, open source code developed by Hagen et al. [8] and Nadjahi et al. [7] was used from their respective GitHub repositories [1] [2]. From these, the approximate_sw and ddks methods were used with default parameters.

A small amount of preparatory work was done as part of the fall project associated with this thesis, which mainly revolved around testing whether the gathered code was suited for the task of the thesis. The conclusion was that the distance

[1] https://github.com/pnnl/DDKS

[2] https://github.com/kimiandj/fast_sw

measures indeed managed to capture differences in the distributions in the data files, and that this distance correlated somewhat with some of the parameters used to initiate the simulations, namely the turbulence intensity. The remaining code work in the fall was done to understand the functionality of the model, and to explore the distributions of the variables used in the model.

The code work in this thesis was done on an internal development cluster within SINTEF. This cluster has the hardware necessary to execute the performance-demanding operations associated with training and using a deep neural network. Specifically it has 24 CPU cores, 512 GB of RAM, and two NVIDIA A30 GPUs. Almost all parts of the following experiments were done remotely through slurm jobs on this cluster.

To start developing on the remote cluster, some of the code that was used in the fall project had to be moved over to the cluster and integrated in such a way that is was as reusable as possible.

5.2 Experiments

5.2.1 Deep learning model

Data split

Our leading hypothesis regarding the data distributions and the deep learning model's performance is that we can quantify the distance between the training data and prediction data, and use this to say something about the expected performance of the model. To test this, we need to engineer a situation in which the model has been trained on a subset of the data, and then use it to make prediction on a different subset of the data with different characteristics. We know that the turbulence intensity used to initialize the simulations has a significant impact on the characteristics of the data. Therefore we choose to split the data set on turbulence intensity values to test our hypothesis.

A train-test-predict split of the dataset was done with a hard cut-off at turbulence intensity value of ten, meaning that all files with higher than ten turbulence intensity values were only used to make predictions, and never in training. For the files with lower than ten turbulence intensity, one of each three files with the same parameters, chosen randomly, were kept as testing data, and not used in training. The data files that use the same random seed show some level of similarity to one another due to the seed, and therefore it is important to include files that use all three in the training set. This resulted in a training set consisting of 306 files, a test set with 153 files, and a prediction set with 438 files.

Network layers, input and output

The model architecture was slightly modified and improved by the original author between the fall and spring work, and for the following, we use the improved model. A diagram of the layers in the model can be seen in figure 5.1. The layers

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 500, 30)           300

 time_distributed (TimeDist  (None, 500, 30)           930
 ributed)

 lstm (LSTM)                 (None, 500, 100)          52400

 dense_2 (Dense)             (None, 500, 70)           7070

 dense_3 (Dense)             (None, 500, 50)           3550

 dense_4 (Dense)             (None, 500, 3)            153

 dense_5 (Dense)             (None, 500, 1)            4

=================================================================
Total params: 64407 (251.59 KB)
Trainable params: 64407 (251.59 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 5.1: A layer-by-layer description of the model used in the experiments.

are as follows: one dense layer with 30 units, one time-distributed dense layer with 30 units, one LSTM layer with 100 units, then finally four dense layers with 70, 50, 3 and 1 units respectively. The time-distributed layer and the LSTM layer use hyperbolic tangents as activation functions, and the dense layers use rectified linear units as activation functions, except for the final one which uses a linear activation. In total the network contains 64,407 trainable parameters.

As input, the model takes a batch of 500x9 tables, which are 500 time steps of the nine input variables. For the rest of this thesis, these 500x9 tables will be referred to as data points. For each data point it outputs 500 time steps of the single target variable. The model trains to predict values for the target variable that correspond with the values of input variables during the same 500 time steps. For a full breakdown of the data set's variables that are used in the model, see section 4.1.3.

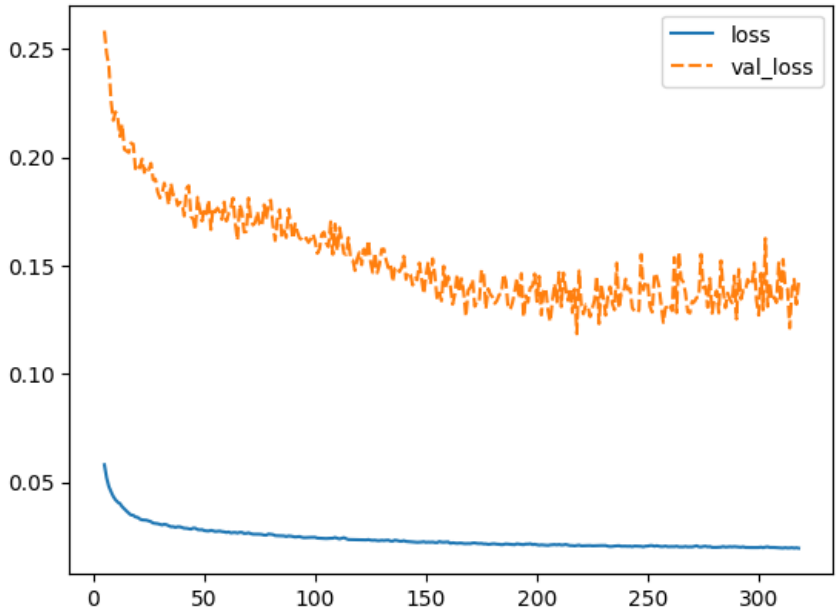


Figure 5.2: Training loss (blue) and validation loss (orange) per epoch during training. The first five epochs are not plotted, due to how the high values in those epochs would scale the graph in such a way that the two lines would be hard to distinguish between.

Training procedure and hyperparameters

The training procedure itself had to be partially rewritten for the purpose of this thesis. The choices of loss function and optimizer algorithm remained unchanged throughout the duration of this thesis and the preceding fall project. The provided training procedure was developed through experimentation on different architectures, and contains was rewritten as part of this project to fit the purpose of the thesis. The procedure uses the Adam optimizer with a learning rate of 0.0005, and mean square error (MSE) as the loss function.

The model is trained with the whole training set, but keeping the last 10% for validation. The data is split into batches of 1024 data points, and the training is set to run for at most 800 epochs. Due to the relatively low batch count per epoch (the whole training set after the 10% validation holdout is divided into 16 batches), a very patient early stopping procedure is implemented that begins at epoch 20. This will stop training and keep the model weights that resulted in the lowest validation loss after observing 100 epochs in a row without any improvement on the lowest observed validation loss so far.

Performance during training

Figure 5.2 shows the training loss and validation loss during training. As we can see, the model seems to train very effectively on the training data, but there is reason to believe that it overfits on this, since the validation loss is quite a bit higher. Still, we consider the validation loss to be low enough for our purposes. We can accept a certain amount of error as a baseline, and instead direct our focus towards how much this error increases when the model is used on data in the prediction split. The early stopping procedure was implemented to remedy the worst of this overfitting.

Figure 5.3 displays the predicted values of the target variable plotted against the ground truth from one of the data files in the testing split of the data set. A slight overfitting is evident, however, the model performs accurately enough for our purposes.

5.2.2 Statistics and distance calculation

The data files are very large, which is a problem for the methods we have chosen to measure the distances between distributions. In an effort to reduce their size but conserve their statistical distributions, four statistics are calculated for each of the nine input variables. This is done once for each data point. This results in a transformation from a 9-column table with 30,000 entries, to a 36-column table with 60 entries.

The chosen statistics are mean, standard deviation, kurtosis and skewness. These were chosen with the intent of finding ways to represent our data set which would also be applicable for other data sets from other domains. In this way, what we find in this project could also be used in other applications. A secondary motivation for this choice is that, while we know turbulence intensity to be a good indicator for the distributional distance of the data sets, this is often difficult to use in practice, since the way turbulence is measured today is quite inaccurate.

From we use the statistics to calculate the Wasserstein distance and the d-dimensional KS distance between the files' distributions. For this, the approximate_sw and ddks methods are fetched from their respective GitHub repositories (as mentioned in section 5.1), and applied with default parameters, except for one key change: the centering parameter in approximate_sw was set to false. This is due to the risk of centering low turbulence data and high turbulence data down to a similar scale. If that happens the distance calculator might not be able to catch all of the ways in which the data sets differ from one another.

These distance calculations are quite operationally heavy. In order to remedy this, a random sampling was done from the statistics representing the training data. The size of this sample was determined incrementally, i.e. it was gradually increased until there was no noticeable change in distributional distance between the sample and statistics from other data files. For the Wasserstein distance, this resulted in a training set sample of 5000 data points where the distance estimate had converged. For ddks, a large enough sampling resulted in much longer calculation time than anticipated. For this reason, we decided to not use ddks as a

metric moving forward.

5.2.3 Predictions

The predictions and distance calculations are done in parallel. This process begins with loading in all of the data files from the testing and prediction splits, keeping track of which files are from each of them. The model weights are loaded in from the earlier training procedure. Then a prediction is made for each data point in each file. The MSE of the predictions is measured over each files, which is stored alongside the distances of the data in that file to the training data's distribution. The results from the two splits are stored separately, so as to better differentiate them when presenting the results.

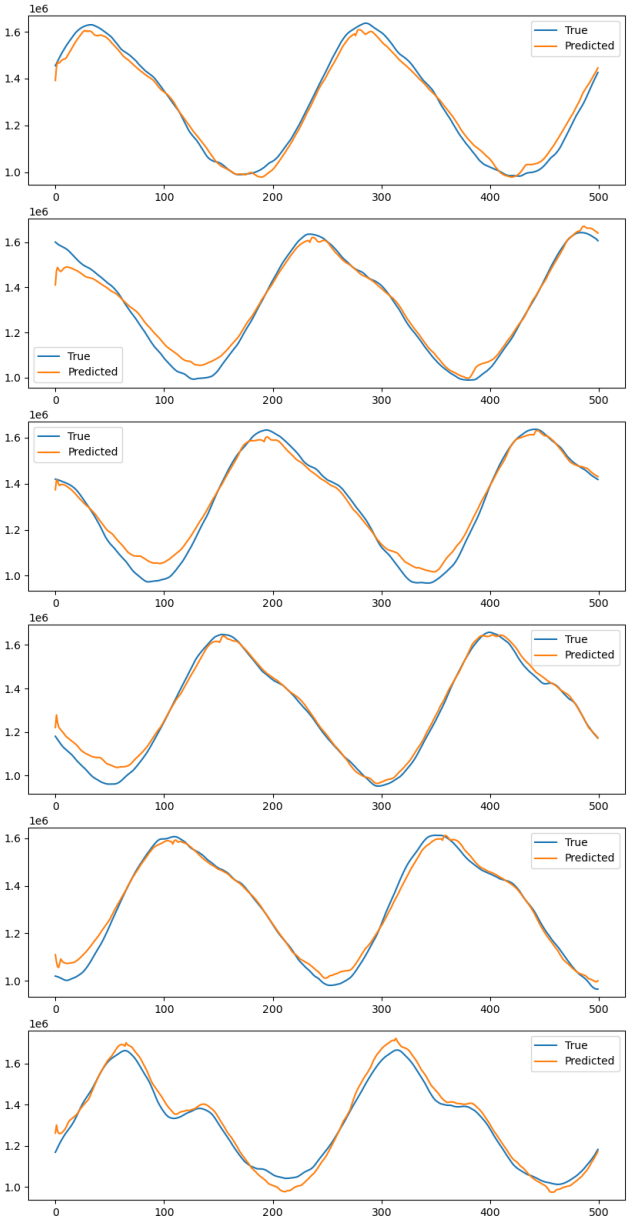


Figure 5.3: Predicted values plotted against ground truth for six data points from the file named "U7.9_TI4.68_D1.11_SH0.79_DIR-0.03_2", which is part of the testing data split. The data points are the first points drawn from each sixth of the file's contents. The MSE over the file is measured at 1,329,135,928. This is taken over non-scaled values of the target variable. Recall that this variable contains values in range of 1 to 2 millions.

Chapter 6

Results

This chapter gives an overview of the results from the various stages of the experiments. The changes to the experiment procedure between stages are presented when appropriate. Primary insights are presented based on the individual results, and are used to argue for the directions taken during the project.

6.1 Prediction results

This section will present the results from the prediction runs, and then compare them to the various distance measures we have chosen for measuring distance between data distributions.

6.1.1 Prediction performance

Refer to figure 6.1 for a plot of predicted values against ground truth for a file in the prediction data split. See appendix B for additional prediction plots in the same format. The model shows an obvious decline in performance when applied on the prediction data from that when applied on the test data. Recall that the only condition that the splits were made on was whether the turbulence intensity was above or below ten. In other words, the differences in distributions caused by this increase in turbulence intensity are large enough that the model is unable to regress properly when it is only trained on low turbulence data.

6.1.2 Comparison between predictive performance and data metrics

The results from the prediction procedure described in section 5.2.3 were downloaded from the cluster. The values for the five simulation parameters were extracted from the file names and stored in the same lookup table. Then, these seven values were plotted pairwise against each other. The full plot grid can be seen in figure 6.2. The two data subsets are kept separate and colored distinctly so as to visualize the differences between them.

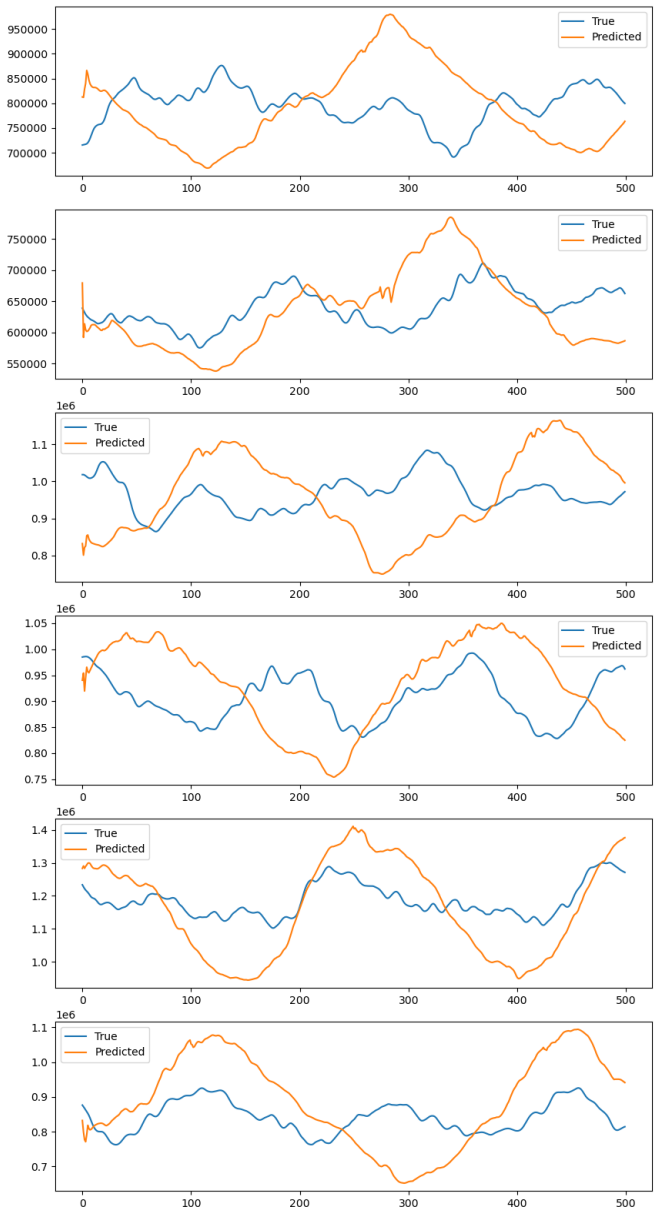


Figure 6.1: Predicted values plotted against ground truth for six data points from the file named "U5.6_TI19.84_D1.07_SH0.10_DIR0.10_1", which is part of the prediction data split. The data points are the first points drawn from each sixth of the file's contents. The MSE over the file is measured at 9,650,075,971.

The primary insight to be read from this is that the predictive performance of the model on a set of data shows a very weak, if any, correlation to the distributional distance of that data to the model's training data. This can be seen more clearly in figure 6.4. However, there is a much stronger correlation between model performance and turbulence intensity values. Shown in figure 6.5, this is coupled with a seemingly inverse proportional relationship between prediction performance and wind shear. This can be explained by the notion that turbulence often makes the blade vibrations we are predicting much more unpredictable, whereas high shear leads to larger, but also more stable, oscillations in the blades.

Finally, it is worth mentioning the relations between prediction performance and the remaining simulation parameters. These are visualized in figure 6.3. The correlations here are minimal, which is to be expected considering how each of the data subsets contain files that are similarly distributed over the same values for these two parameters.

Too see whether the Wasserstein distance values were being unduly influenced by the similarities between the data sets, the statistics tables were reduced to only include the statistics that showed the most different values in the different data splits. These statistics were the following:

- Rotor speed mean and standard deviation
- Generator speed, all four statistics
- Blade 1 pitch angle, all four statistics
- HSS torque, all but kurtosis
- Nacelle fore-aft acceleration, all but mean
- Nacelle side-side acceleration standard deviation

With that reduction, the Wasserstein distances were recalculated with the same procedure as for the previous results. A new grid of density plots was generated with the recalculated distances. However, this resulted in a almost identical table to that shown in figure 6.2.

After this initial run of experiments we have failed to find a way to reasonably quantify the inherent distances between the data sets that lead to the model's reduced performance. However, we know that there is such a potentially quantifiable difference there, due to how the model drops in performance after transfer. In a continued effort to find a way to quantify this difference, the research went in two directions, which will be detailed further in the rest of this chapter.

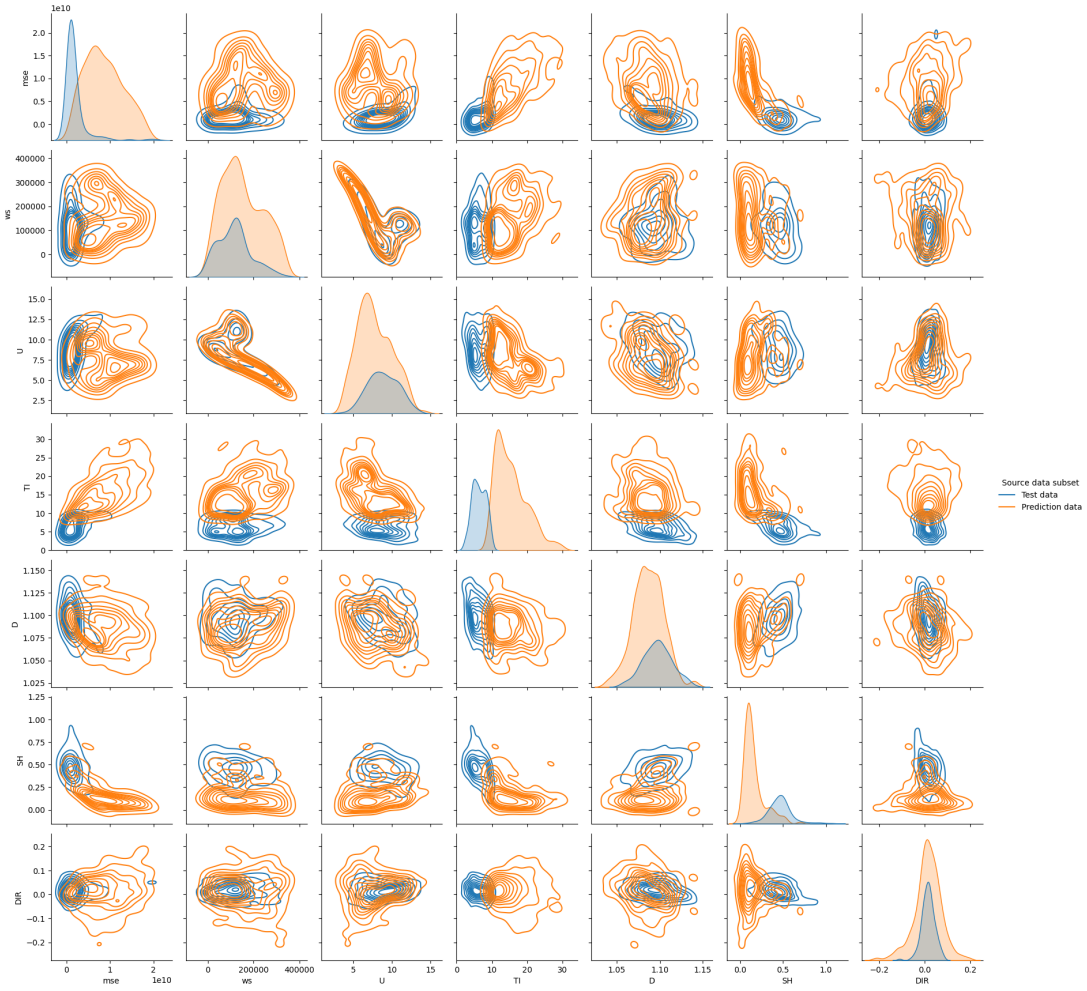


Figure 6.2: The MSE from the model's prediction over a data file, that file's Wasserstein distance to the training data, and the file's initialization parameters plotted pairwise against each other to show their distributions. Along the diagonal are the kernel densities of each of these metrics, the other plots are two-dimensional kernel density plots.

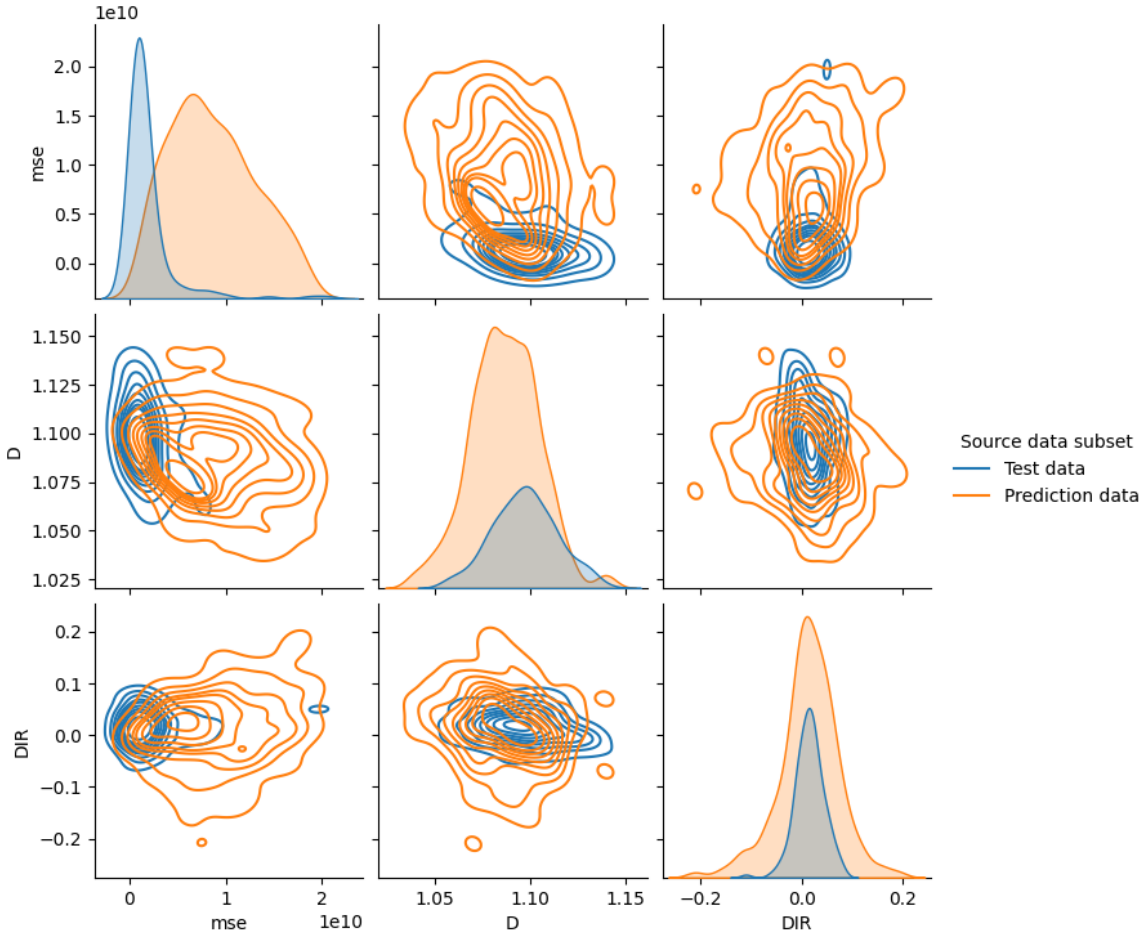


Figure 6.3: Pairwise kernel density plots between prediction MSE, and air density (D) and wind direction (DIR) parameters used to create the data files.

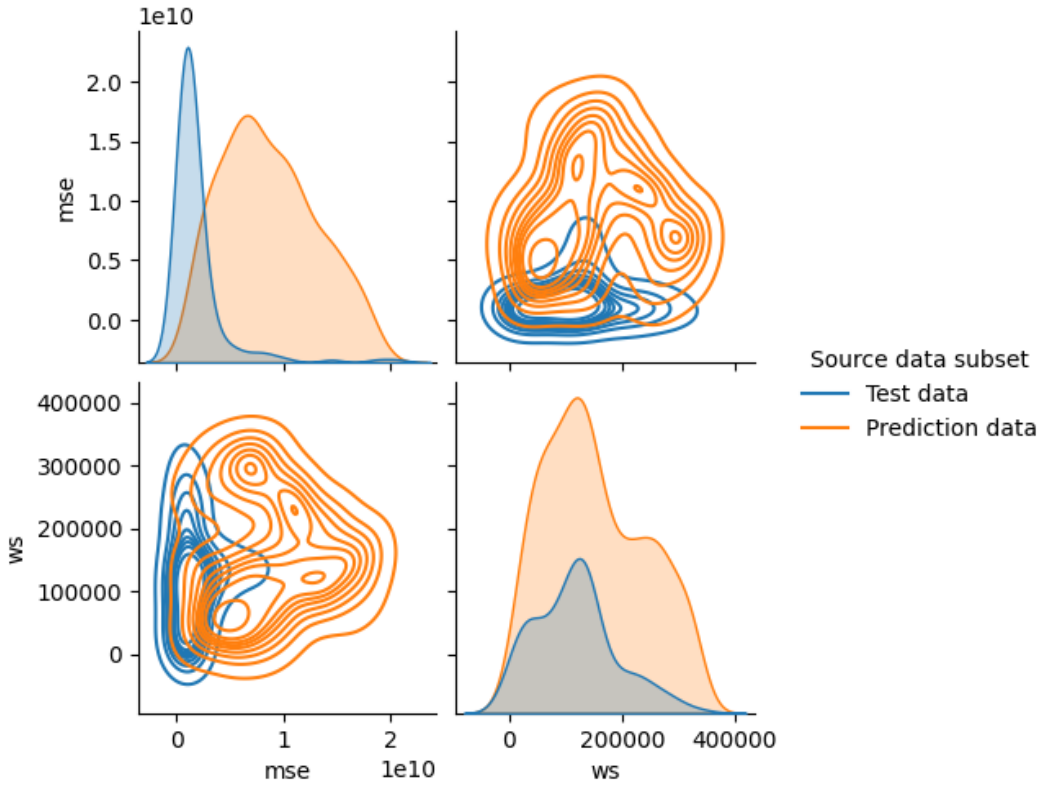


Figure 6.4: Pairwise kernel density plots between prediction MSE and Wasserstein distance to the model's training data.

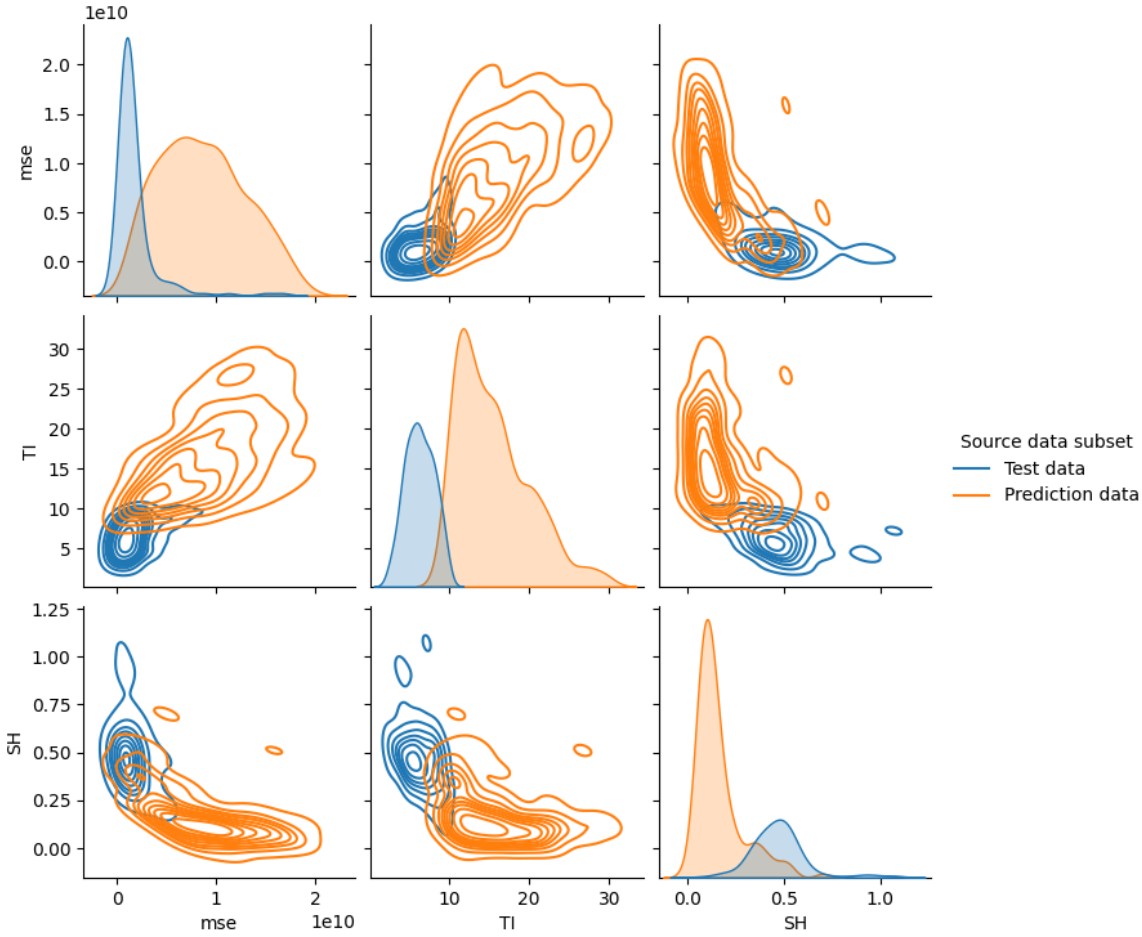


Figure 6.5: Pairwise kernel density plots between prediction MSE, and turbulence intensity (TI) and shear (SH) parameters used to create the data files.

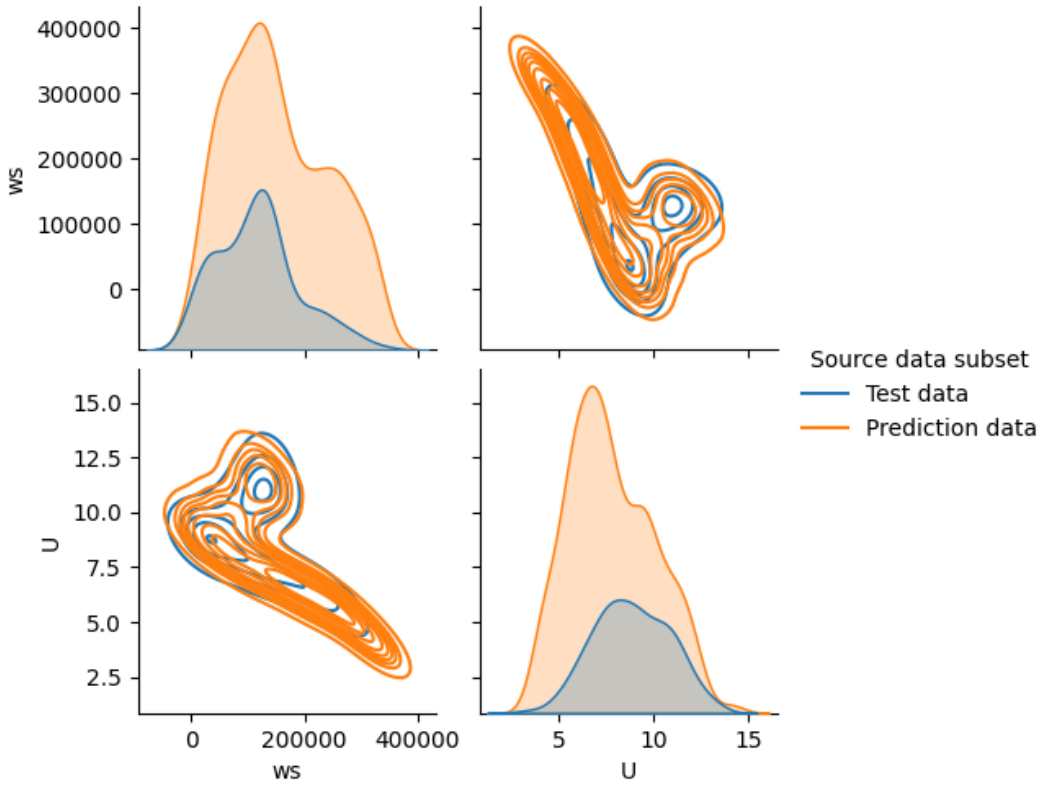


Figure 6.6: Pairwise kernel density plots between Wasserstein distance to the training data, and the wind speed (U) parameters used to create the data files.

6.2 Alternative methods for distance calculation

As stated, the research continued in two directions. The first was to review the findings in the data exploration phase of the project and try to find some way to represent the data set in a more domain specific way. The other was to use more advanced data representation methods that could better represent the distributional distances. The project did not go very far in either direction, mainly due to time constraints.

6.2.1 Review of findings from data exploration

Recall that during data exploration there were two main observations that stood out: how the data becomes more erratic and unpredictable in its oscillations over time, and how that a lot of the variables show a significant change in their ranges after the blades enter into a pitch angle rather than facing the wind direction head on. Of these two, the latter was deemed to be the simplest to create potentially meaningful statistics out of.

6.2.2 Blade pitch angle statistics

The other variables of the data set seemed to change the most when the blades were changing their pitch angle. To measure this change, the difference between time steps was measured for each time step for the variables "Blade 1 pitch angle" and "Blade 1 controller demanded pitch rate". The median of these differences was stored for each data point. This resulted in a new 2x60 table for each data file. Then the Wasserstein distances between files were calculated following the same procedure as earlier, and used to produce a new grid of density plots. The grid can be seen in appendix C. The relationships between the Wasserstein distance and the other metrics were now even less correlating than before, suggesting that this is either a dead end, or that there are much better ways of creating a statistic based on the blades' pitch angle.

6.2.3 t-SNE

In the effort of using a more advanced data representation method a simple, out-of-the-box implementation of t-SNE from the scikit-learn package was applied to the data set. This implementation was fetched from the scikit-learn framework. The intention was to create a visualizable 2-dimensional embedding of the data that could provide greater insight into the relationships between the variables. However, from this initial attempt at executing it, and due to the lack of remaining time to complete the thesis, we conclude that this method is too computationally heavy to be used with this implementation. However, we believe that there is reason to explore further with a more efficient implementation.

Chapter 7

Discussion

This chapter provides a discussion of the findings in the earlier chapters, put into context with the findings from the literature search. The relevant points of discussion from the literature are presented here, before a discussion on the implications of our findings is given. We then answer research questions before talking about the strengths and limitations of the thesis. Suggestions for further research are given throughout the discussion whenever relevant.

7.1 Interpretation of results

7.1.1 Contextualizing with literature

Several papers that were found in the literature review on transfer learning on wind turbine data mention the use of variations of maximum mean discrepancy as an additional optimization objective during domain adaptation, like Shen et al. [12], Lv et al, [14], Zhu et al. [16] and Zhang et al. [17]. This is done to make the model account for the difference in distributions between the source and target domains, and map them very similarly in the latent space. Deep joint distribution alignment is presented as a mechanism that fulfills the same purpose by Qin et al. [15], using Wasserstein distance and information entropy between the target distribution and the source conditional distribution given the target domain. Many of these papers also highlight the importance of performing transfer learning to reach adequate performance levels, especially when there is a lack of representative data for the target domain. If the model we are using were to be transferred in a similar way, there is reason to believe that the distance measure we are searching for to represent the model's uncertainty also could be used as part of a loss function during the transfer, similarly to these papers. It is likely that all of these distance measures that were presented among the papers our literature review to be viable options. We chose Wasserstein distance as a promising candidate, but believe that there is reason to try the others as well in future research.

7.1.2 Implication of findings

To reiterate the motivation behind finding a distance measure that is representative of the model's uncertainty on new data sets, consider the following: if we were to deploy a deep learning model that uses sensor data within the turbine nacelle to deliver prediction on the forces acting on the blades, we would want to deploy it in a state where we are certain that the model will deliver sufficiently accurate prediction. Without the use of transfer learning, we simply do not have access to enough sensor data that can train this model, unless the sensor data is similar enough to something we can simulate. To get a concept of how much new data we need to acquire, if any, for a given location, we sought a distance measure that can quantify the performative drop the model is expected to have on this sensor data in this location, and therefore how much data is required, to make the model adapt. As an additional motivation for finding this distance measure, we see in the literature a common trend of applying domain adaptation methods that incorporate these distance measures as part of the loss function, often to greater success than without [12] [14] [15] [16].

We know that the search for a viable distance measure was not exhaustive. There are several candidate measures presented in the literature that were not attempted. Maximum mean discrepancy, and variations on this, have thus far been the most common distance measures used as additional loss functions during domain adaptation [6]. Kullback-Liebler divergence is similarly a commonly adopted distance measure between distributions [6].

We believe that there is good reason to continue the search, both because we believe there to be something to search for, and due to the potential advantages this model could entail for the wind power industry. We see from our experiment results that turbulence intensity is a good indicator for a drop in model performance in our data set. However, as previously stated, this is difficult to measure in practice, and is only really useful when one has values for it declared beforehand like we have with the simulations. If one can find one or more aspects of the data that change in tune with wind turbulence, these will likely be strong candidates for useful representations of the data. In our data exploration we found that the jaggedness, or the chaotic oscillations, of the variables increased when turbulence did. This is also a likely reason why our search failed, that our choices of measures and representations were unable to account for this jaggedness.

7.1.3 Answering research questions

RQ1: How can transfer learning be applied to time series data for wind turbines? For the answer to this question we refer to the findings in our literature search. In the literature we see several successful implementations of transfer learning methods on time series data, both of wind turbines and other applications. In particular we see plenty of success stories applying LSTM-based or transformer-based architectures, or other deep learning methods for modelling time series data, that incorporate transfer learning. In recent years, the develop-

ment in our field has focused mostly on LSTM- and transformer-based architectures, and how transfer learning can be applied to them in order to solve the problem of missing data for a specific task. To solve this, domain adaptation techniques like fine-turning and distribution alignment are employed, allowing a network to account for the similarities and differences between a general data set and more specific ones.

RQ2: What methods are feasible for measuring the distance between sets of multivariate time series? Our task was to find a generally applicable methodology for measuring the differences between training data and input data which related to the overall performance of the deep learning model. We recognize that such a methodology will always rely somewhat on the properties of the data set. However, what we tried to implement out of the box was not able to capture the properties in the data set that we know are caused by a domain specific quality, namely wind turbulence. Inspecting the target time series by eye, we see that the they become more jagged with increasing turbulence, but we struggled to find informative statistics in the input data related to this.

RQ3: How does the error in wind turbine models change after introducing transfer learning methods? And RQ4: How does the change in error after introducing transfer learning relate to the data sets? In this thesis we retrained a model and directly applied it to a range of inputs that were increasingly distant from its training data. The purpose of this was to whether we needed to employ transfer learning in order to bring it within an acceptable margin of error in its predictions. This was done to get an understanding of when transfer learning would be required. We know that the model is invalid for data that is far away from its training data. But what happens when we apply it something that is only somewhat distant? Will the model's performance stay within acceptable margins? This would be answerable without labelled data if we had the distance measure we are looking for. This again relates to the point that the simulated data is not precisely representative of the real world. An appropriate distance measure would help us determine whether or not a model trained on simulated data needs to be fine-tuned on real data before being deployed. The answer to these two questions therefore lies in this distance measure. Our search for it has turned out inconclusive, but we have ruled out some options and given suggestions for further work.

RQ5: How can our knowledge about transfer learning in our domain be applied in other domains? To whatever degree a functional distance measure is specific to characteristics of our domain, some precautions will need to be taken when applying it to other domains. Data set characteristics and interrelations of dataset features should be accounted for, and domain experts should be involved in the process. But, the idea of measuring distance to training data will in principle be valid for all supervised learning models given that a proper distance measure can be defined.

7.2 Strengths and limitations

7.2.1 Strengths

A strength of this research is its intention of finding more universally applicable solutions. The focus in the experimental work could have been directed at finding more domain specific solutions, and we recognize now that this might be necessary for future research. A less domain specific approach would require less extra work to properly apply to other domains.

Another strength is the previously stated high potential upside should the search turn out successful. A deep learning model for predicting forces exerted on wind turbine blades, that can be applied to a wide range of weather conditions, and be deployed with a certainty of its expected performance, can have several great benefits. Maintenance of the blades can be scheduled at more appropriate times, and further ahead of time, than today, potentially minimizing the associated cost. Additionally one can predict whether or not impending weather changes are likely to cause accidents, letting appropriate precautions to be put in place in response to this. Only this year have we seen to such accidents in Norway, where a wind turbine blade or the rotor itself fell off during operation [33] [34].

7.2.2 Limitations

Perhaps the greatest limitation with this research is the lack of access to real data. If real data had been available in large enough quantities, this could have supplied a more accurate understanding of the turbine's systems over time, under a broader range of naturally varying weather conditions. Due to how the simulations are initiated with random seeds, and how the data extracted from them show similarities based on these seeds, the data set is likely not representative enough of the diverse weather patterns one might expect in the real world.

There are several data representation methods and distance measures between distributions that weren't tried in this thesis. Due to the limited time frame we decided to focus on a narrow selection, with a prioritization towards those that could be broadly applicable across domains. This focus was evidently too narrow to produce any significantly positive results.

7.3 Relevance to sustainability goals

The United Nation's Sustainability Goals are 17 goals for global sustainability [35]. This thesis has a considerable relevance to several of them.

In terms of being a part of research on a renewable and fossil-free energy source, the thesis shows strong relevance to goals number seven and thirteen. These goals are to supply sustainable and reasonably priced energy to all, and to stop climate change respectively [35]. However, as deep learning algorithms usually are quite energy demanding, they will be relevant to both of these goals

regardless of their application domains. We believe that finding ways to reduce their energy demand, like decreasing training time or need for data gathering like in this thesis, to be a step in the right direction.

Wind turbines are known to have a negative impact on wildlife wherever they are deployed. With modern technology they can be deployed both on land and at sea. Thus this research is related to sustainability goals number fourteen and fifteen, which are to conserve wildlife at sea, and on land, respectively [35]. We believe that the research this thesis is a part of can help increase the yield from deployed turbines, such that fewer of them need to be deployed to meet energy needs.

Chapter 8

Conclusion

To conclude we have deemed that our attempts at quantifying the variations in performance in our model based on data distributions to be unsuccessful. However, we believe there are several insights to be drawn from the process of finding a distance measure that is representative of this uncertainty, despite not finding the measure itself.

We have found that this uncertainty measure is more likely to be domain dependent that initially hoped. In our data set we have found that wind turbulence has a significant effect the characteristics of the variables. As such, we conclude that a proper representation of these changes in characteristics is a good candidate for further exploration towards measuring the uncertainty of the model on this data set.

We encourage the continued exploration of possible solutions to this problem mainly for the reason that it will make it possible to anticipate a minimal requirement of fine-tuning to bring the model's performance within acceptable margins. Secondarily, this distance measure can be used to employ more advanced transfer learning techniques on this particular task and domain.

Finally, we conclude that a greater domain knowledge will be of great help toward finding and taking advantage of the proposed methodology. The field of deep learning, and the use of transfer learning within it, are rapidly advancing. Finding appropriate methodologies needed to deploy a well-functioning model will be of great benefit to the wind power industry as a whole.

Bibliography

[1] E. G. Hansen, 'Using deep learning as a surrogate for wind turbine simulations,' *Project report for the course IT3915 - Master in Informatics, Preparatory Project*, 13th Dec. 2023.

[2] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[3] M. Banoula, 'Machine learning steps: A complete guide,' *Simplilearn*, 21st Aug. 2023. [Online]. Available: https://www.simplilearn.com/tutorials/machine-learning-tutorial/machine-learning-steps#:~:text=Machine%20learning%20is%20the%20process,that%20data%20to%20learn%20more. (visited on 04/06/2024).

[4] M. West. 'Explaining recurrent neural networks.' (2023), [Online]. Available: https://www.bouvet.no/bouvet-deler/explaining-recurrent-neural-networks (visited on 06/12/2023).

[5] C. Olah. 'Understanding lstms.' (2015), [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs (visited on 06/12/2023).

[6] M. Weber, M. Auch, C. Doblander, P. Mandl and H.-A. Jacobsen, 'Transfer learning with time series data: A systematic mapping study,' *IEEE Access*, vol. 9, pp. 165 409–165 432, 2021. DOI: 10.1109/ACCESS.2021.3134628.

[7] K. Nadjahi, A. Durmus, P. E. Jacob, R. Badeau and U. Şimşekli, *Fast approximation of the sliced-wasserstein distance using concentration of random projections*, 2022. arXiv: 2106.15427 [stat.ML].

[8] A. Hagen, S. Jackson, J. Kahn, J. Strube, I. Haide, K. Pazdernik and C. Hainje, *Accelerated computation of a high dimensional kolmogorov-smirnov distance*, 2021. arXiv: 2106.13706 [stat.CO].

[9] E. Feigelson and G. J. Babu. 'Beware the kolmogorov-smirnov test!' (2019), [Online]. Available: https://asaip.psu.edu/Articles/beware-the-kolmogorov-smirnov-test/ (visited on 07/12/2023).

[10] L. van der Maaten and G. Hinton, 'Viualizing data using t-sne,' *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, Nov. 2008.

[11] R. Jigyasu, V. Shrivastava and S. Singh, 'Efficient condition monitoring of off shore wind turbines using deep networks,' in *2023 International Conference on Computer, Electronics Electrical Engineering their Applications (IC2E3)*, 2023, pp. 1–6. DOI: 10.1109/IC2E357697.2023.10262466.

[12] Y. Shen, B. Chen, F. Guo, W. Meng and L. Yu, 'A modified deep convolutional subdomain adaptive network method for fault diagnosis of wind turbine systems,' *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–10, 2022. DOI: 10.1109/TIM.2021.3128708.

[13] M. Ulmer, J. Zgraggen, G. Pizza and L. G. Huber, 'Scaling up deep learning based predictive maintenance for commercial machine fleets: A case study,' in *2022 9th Swiss Conference on Data Science (SDS)*, 2022, pp. 40–46. DOI: 10.1109/SDS54800.2022.00014.

[14] M. Lv, S. Liu, X. Su and C. Chen, 'Deep transfer network with multi-kernel dynamic distribution adaptation for cross-machine fault diagnosis,' *IEEE Access*, vol. 9, pp. 16 392–16 409, 2021. DOI: 10.1109/ACCESS.2021.3053075.

[15] Y. Qin, Q. Qian, J. Luo and H. Pu, 'Deep joint distribution alignment: A novel enhanced-domain adaptation mechanism for fault transfer diagnosis,' *IEEE Transactions on Cybernetics*, vol. 53, no. 5, pp. 3128–3138, 2023. DOI: 10.1109/TCYB.2022.3162957.

[16] Y. Zhu, C. Zhu, J. Tan, Y. Tan and L. Rao, 'Anomaly detection and condition monitoring of wind turbine gearbox based on lstm-fs and transfer learning,' *RENEWABLE ENERGY*, vol. 189, pp. 90–103, Apr. 2022, ISSN: 0960-1481. DOI: 10.1016/j.renene.2022.02.061.

[17] G. Zhang, Y. Li, W. Jiang and L. Shu, 'A fault diagnosis method for wind turbines with limited labeled data based on balanced joint adaptive network,' *NEUROCOMPUTING*, vol. 481, pp. 133–153, Apr. 2022, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2022.01.067.

[18] G. Yue, G. Ping and L. Lanxin, 'An end-to-end model based on cnn-lstm for industrial fault diagnosis and prognosis,' in *2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, 2018, pp. 274–278. DOI: 10.1109/ICNIDC.2018.8525759.

[19] K. Shao and Y. He, 'Fine-gained recurrence graph: Graphical modeling of vibration signal for fault diagnosis of wind turbine,' *IEEE Transactions on Industrial Informatics*, vol. 19, no. 8, pp. 8878–8888, 2023. DOI: 10.1109/TII.2022.3222396.

[20] T. Sun, G. Yu, M. Gao, L. Zhao, C. Bai and W. Yang, 'Fault diagnosis methods based on machine learning and its applications for wind turbines: A review,' *IEEE Access*, vol. 9, pp. 147 481–147 511, 2021. DOI: 10.1109/ACCESS.2021.3124025.

[21] M. Nawar, M. Shomer, S. Faddel and H. Gong, 'Transfer learning in deep learning models for building load forecasting: Case of limited data,' in *SoutheastCon 2023*, 2023, pp. 532–538. DOI: 10.1109/SoutheastCon51012.2023.10115128.

[22] S. Yadav, A. Jain, K. C. Sharma and R. Bhakar, 'Load forecasting for rare events using lstm,' in *2021 9th IEEE International Conference on Power Systems (ICPS)*, 2021, pp. 1–6. DOI: 10.1109/ICPS52420.2021.9670200.

[23] H. Dong, J. Zhu, S. Li, Z. Chen, W. Wu and X. Li, 'Dynamic load forecasting with adversarial domain adaptation based lstm neural networks,' in *2023 IEEE/IAS Industrial and Commercial Power System Asia (ICPS Asia)*, 2023, pp. 1130–1135. DOI: 10.1109/ICPSAsia58343.2023.10294791.

[24] M. Chaudhary, M. S. Gastli, L. Nassar and F. Karray, 'Transfer learning application for berries yield forecasting using deep learning,' in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9533530.

[25] K. C. Gadepally, S. B. Dhal, M. Bhandari, J. Landivar, S. Kalafatis and K. Nowka, 'A deep transfer learning based approach for forecasting spatio-temporal features to maximize yield in cotton crops,' in *2023 57th Annual Conference on Information Sciences and Systems (CISS)*, 2023, pp. 1–4. DOI: 10.1109/CISS56502.2023.10089748.

[26] A. Gupta, M. Sheikh, Y. Tripathy and W. D. Widanage, 'Transfer learning lstm model for battery useful capacity fade prediction,' in *2021 24th International Conference on Mechatronics Technology (ICMT)*, 2021, pp. 1–6. DOI: 10.1109/ICMT53429.2021.9687230.

[27] S. Allogba, S. Aladin and C. Tremblay, 'Multivariate machine learning models for short-term forecast of lightpath performance,' *Journal of Lightwave Technology*, vol. 39, no. 22, pp. 7146–7158, 2021. DOI: 10.1109/JLT.2021.3110513.

[28] N. Tasnim, I. T. Imam and M. M. A. Hashem, 'A novel multi-module approach to predict crime based on multivariate spatio-temporal data using attention and sequential fusion model,' *IEEE Access*, vol. 10, pp. 48 009–48 030, 2022. DOI: 10.1109/ACCESS.2022.3171843.

[29] Á. Bueno Rodríguez, R. Balestriero, S. De Angelis, M. C. Benítez, L. Zuccarello, R. Baraniuk, J. M. Ibáñez and M. V. de Hoop, 'Recurrent scattering network detects metastable behavior in polyphonic seismo-volcanic signals for volcano eruption forecasting,' *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–23, 2022. DOI: 10.1109/TGRS.2021.3134198.

[30] M. G. Molina and J. G. Alvarez, 'Technical and regulatory exigencies for grid connection of wind generation,' in *Wind Farm*, G. O. Suvire, Ed., Rijeka: IntechOpen, 2011, ch. 1. DOI: 10.5772/16474. [Online]. Available: https://doi.org/10.5772/16474.

[31] DNV. 'Wind turbine design software - bladed.' (2023), [Online]. Available: https://www.dnv.com/services/wind-turbine-design-software-bladed-3775 (visited on 20/11/2023).

[32] DNV. 'Bladed 4.15 documentation.' (2023), [Online]. Available: https://dnvgldocs.azureedge.net/Bladed%204.15%20Theory%20Manual/index.html (visited on 24/11/2023).

[33] A. N. Brevig, H. N. Ridola, O. B. Strande and A. J. Lexander, 'Vindturbinblad knakk og falt av,' *NRK*, 11th Apr. 2024. [Online]. Available: https://www.nrk.no/innlandet/turbinblad-knakk-og-falt-av-vindturbin-i-odal-1.16839676 (visited on 03/06/2024).

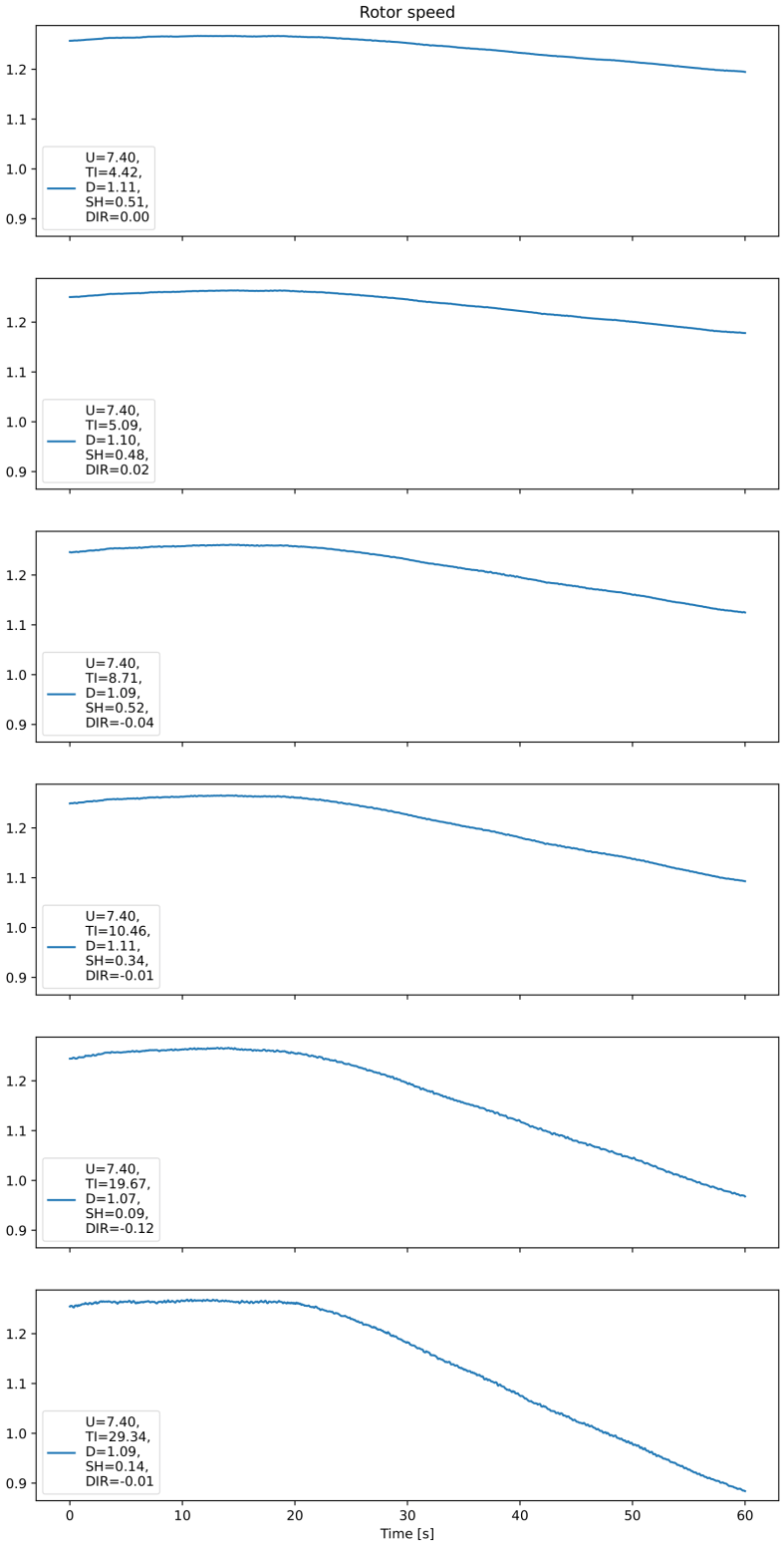
[34] M. Guttormsen, 'Lover opprydning: Gigantisk rotorblad løsnet 100 meter over bakken,' *NRK*, 31st Mar. 2024. [Online]. Available: https://www.nrk.no/nordland/oyfjellet_-150-meter-bred-rotor-falt-ned-fra-odelagt-vindturbin-1.16816996 (visited on 03/06/2024).

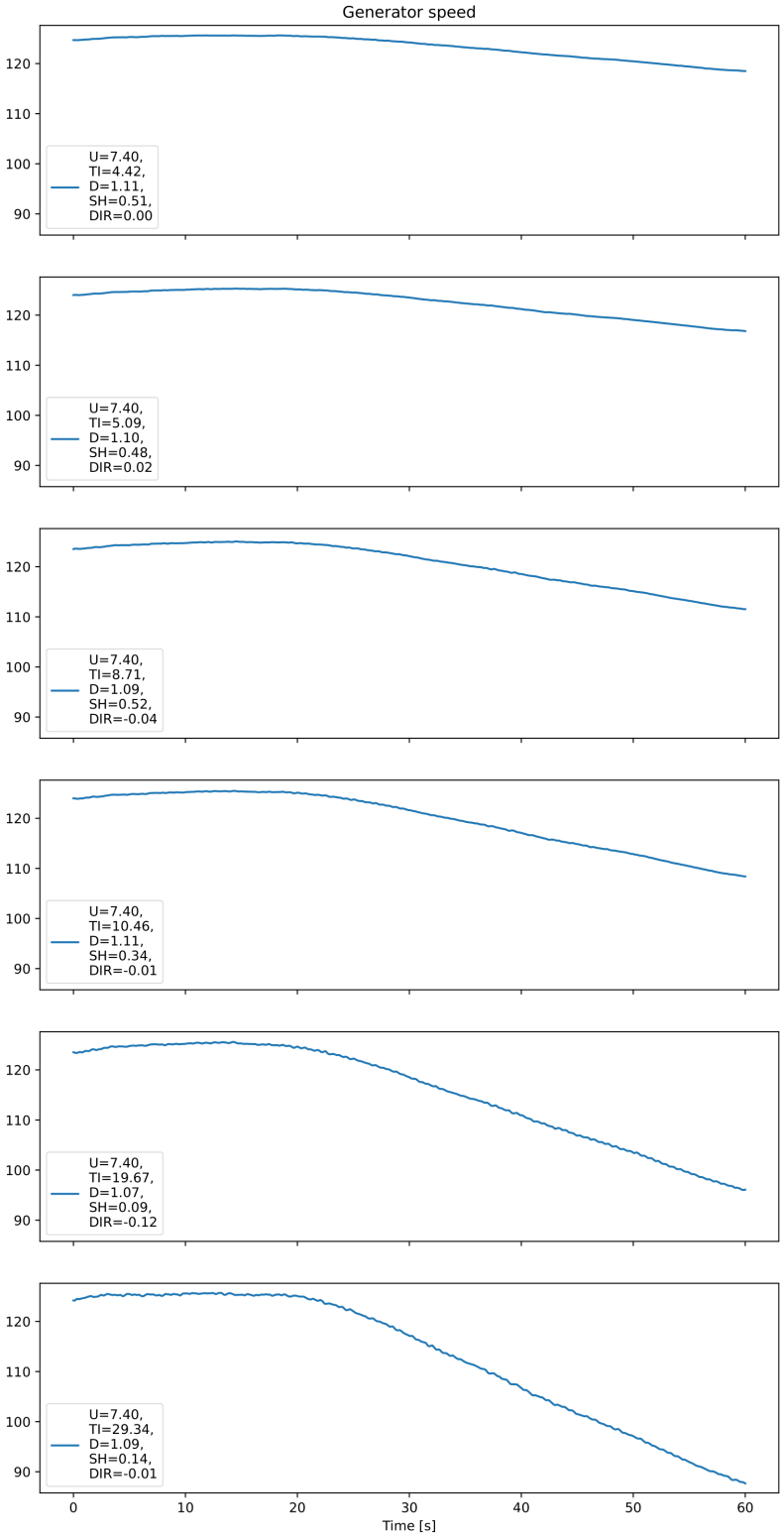
[35] FN-Sambandet, 'Fns bærekraftsmål,' *FN*, 1st Feb. 2024. [Online]. Available: https://fn.no/om-fn/fns-baerekraftsmaal (visited on 03/06/2024).

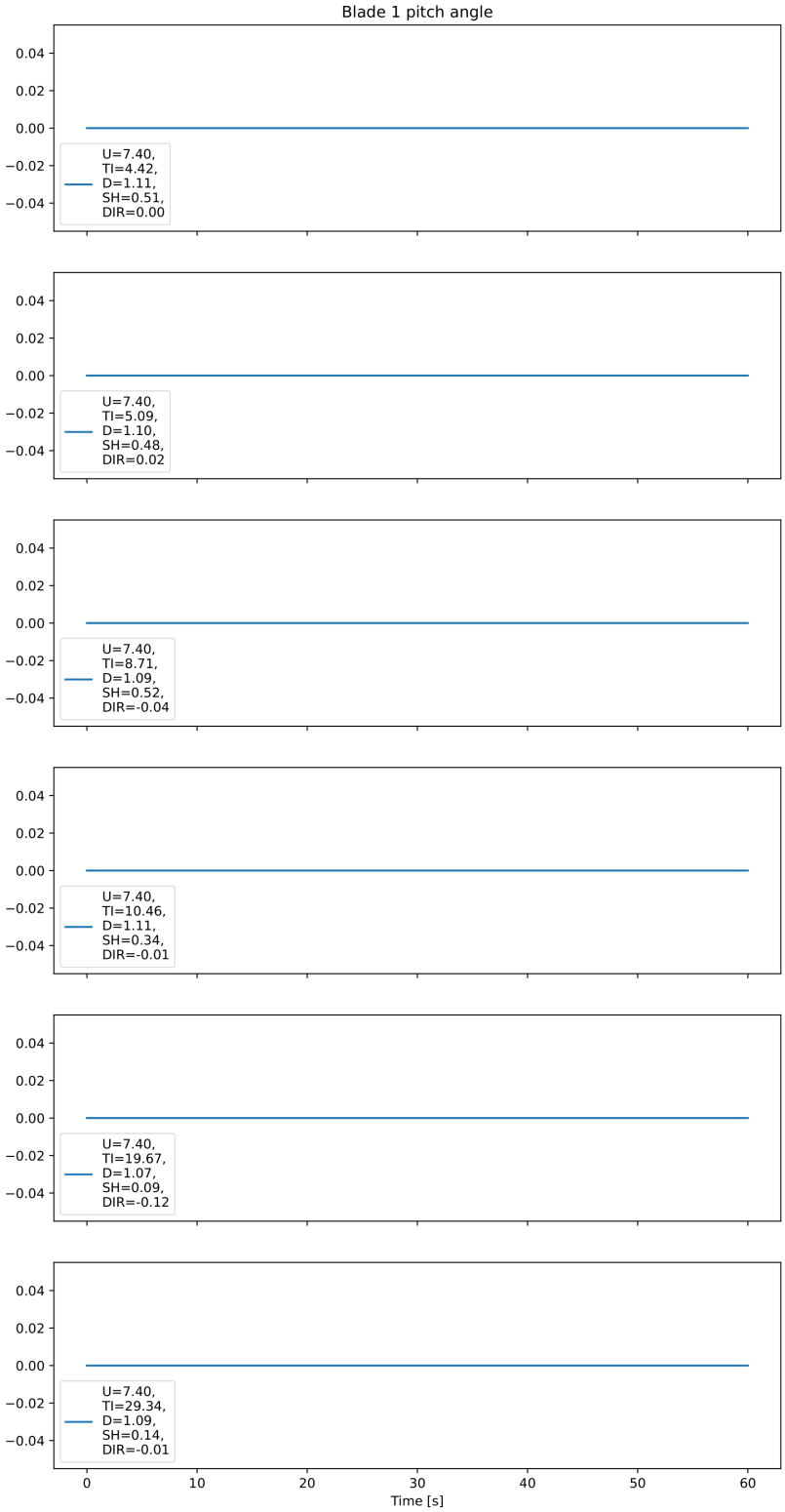
Appendix A

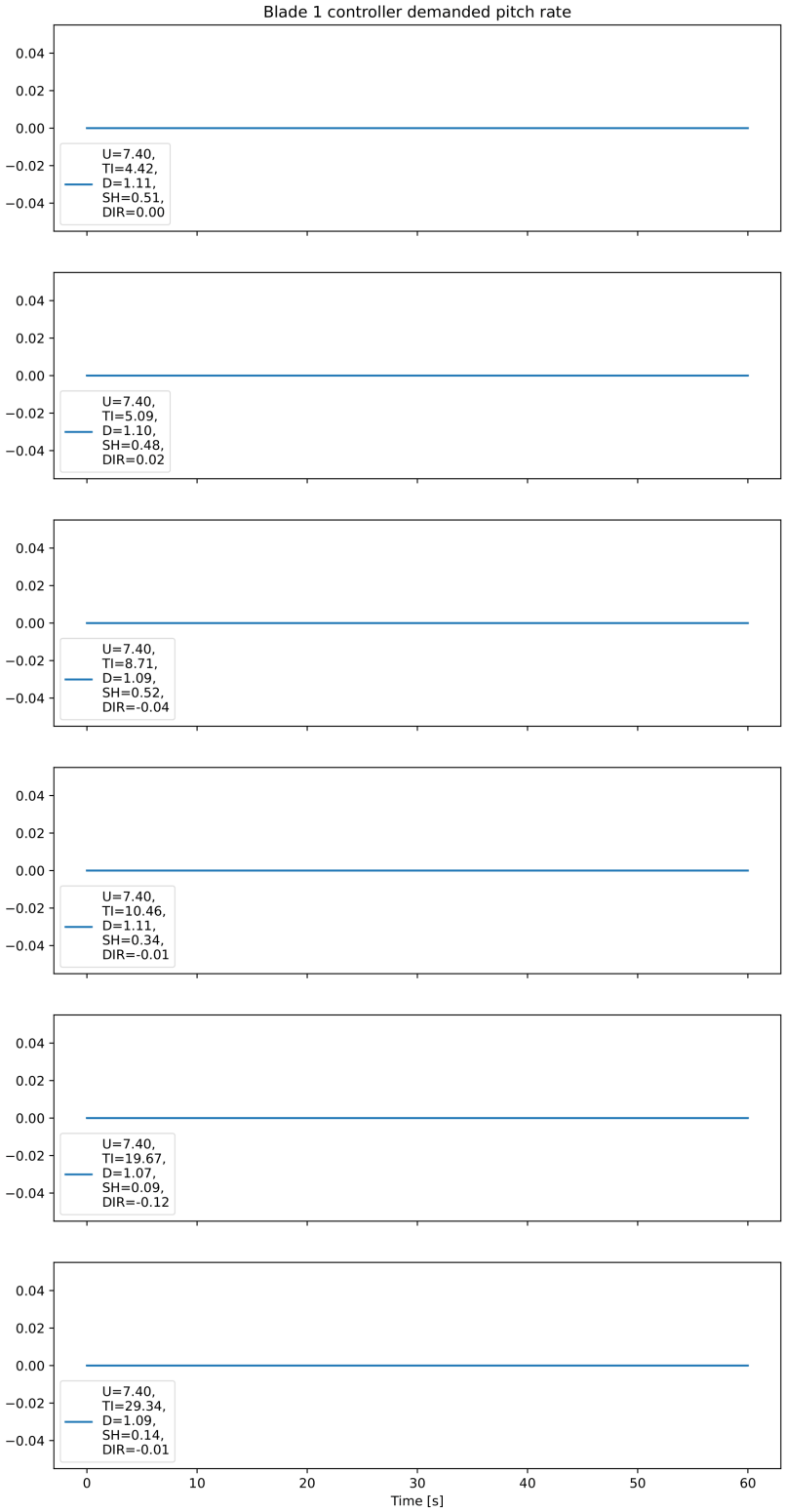
Additional data examples

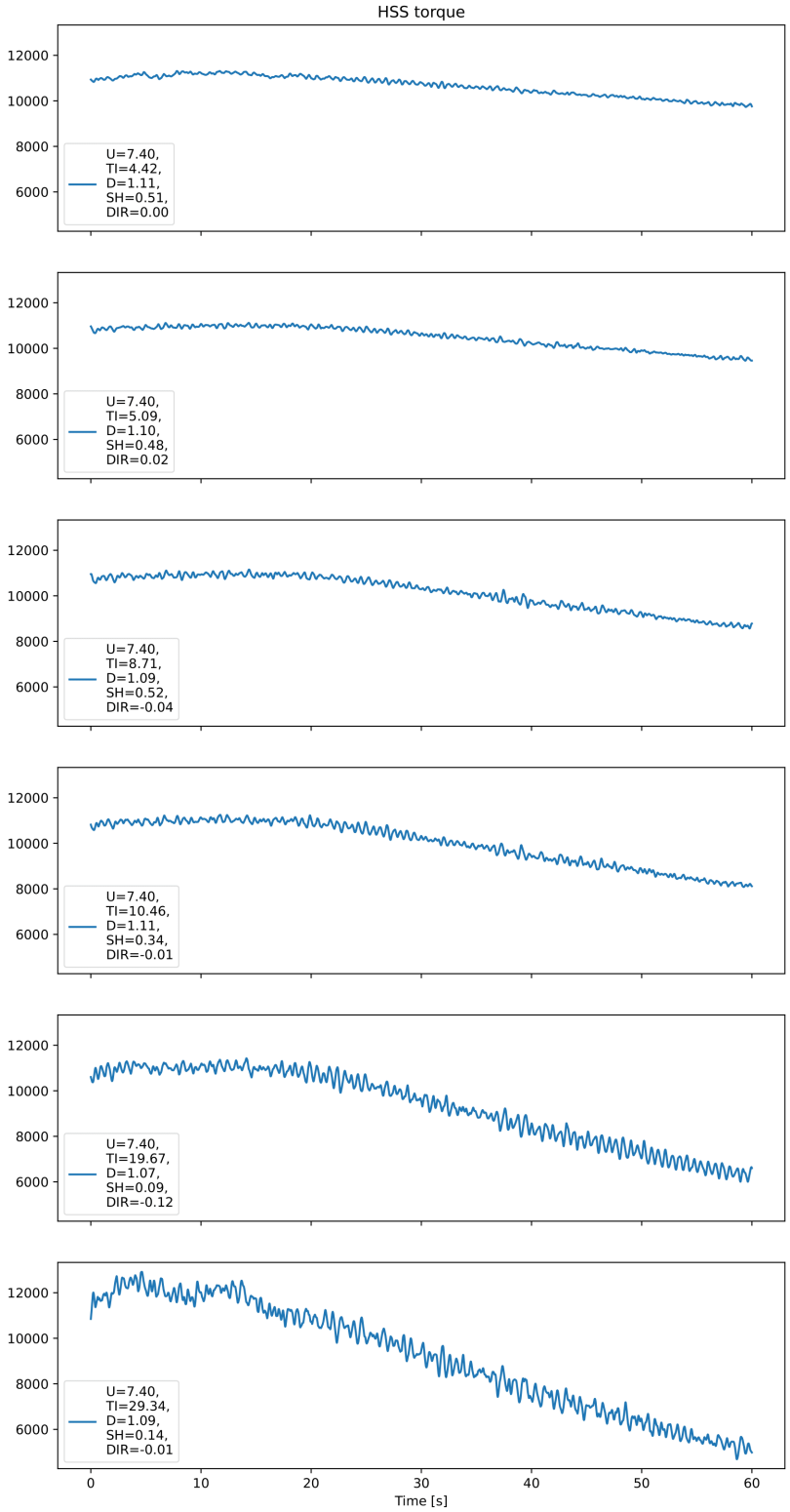
This appendix shows additional data plots that were done in relation to the data exploration portion of this project. The variable name is listed at the top of the figures, and the parameters that make up the file name are shown in the bottom left corner of each plot. All of these files were extracted from simulations that used the first of three randomization seeds. The data shown in the plots are the first ten seconds worth of data from each respective data file. For more information about the simulations, see chapter 4.

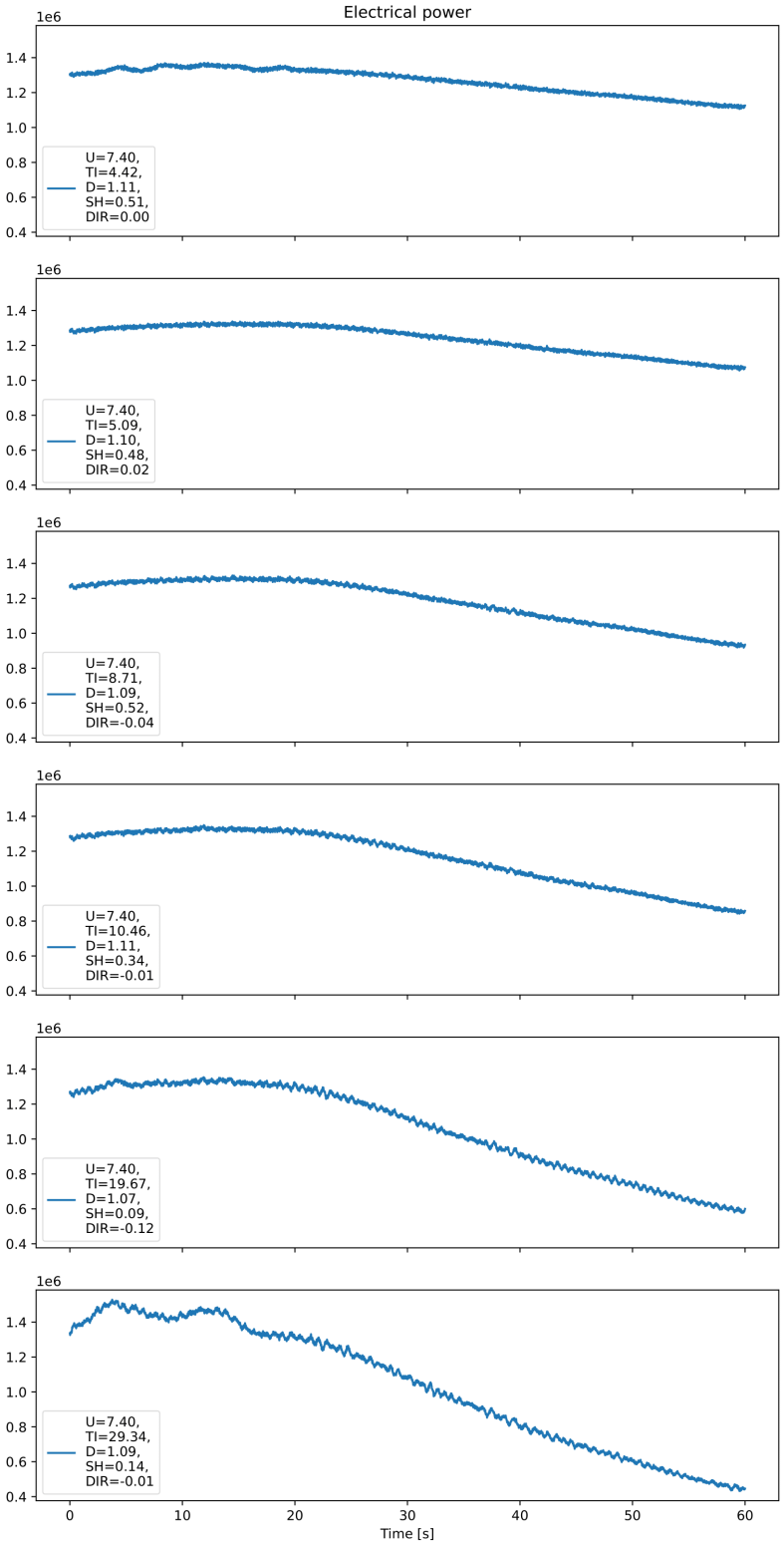

Rotor speed
1.2
1.1
1.0
0.9
U=7.40,
TI=4.42,
D=1.11,
SH=0.51,
DIR=0.00
U=7.40,
TI=5.09,
D=1.10,
SH=0.48,
DIR=0.02
U=7.40,
TI=8.71,
D=1.09,
SH=0.52,
DIR=-0.04
U=7.40,
TI=10.46,
D=1.11,
SH=0.34,
DIR=-0.01
U=7.40,
TI=19.67,
D=1.07,
SH=0.09,
DIR=-0.12
U=7.40,
TI=29.34,
D=1.09,
SH=0.14,
DIR=-0.01
0
10
20
30
40
50
60
Time [s]

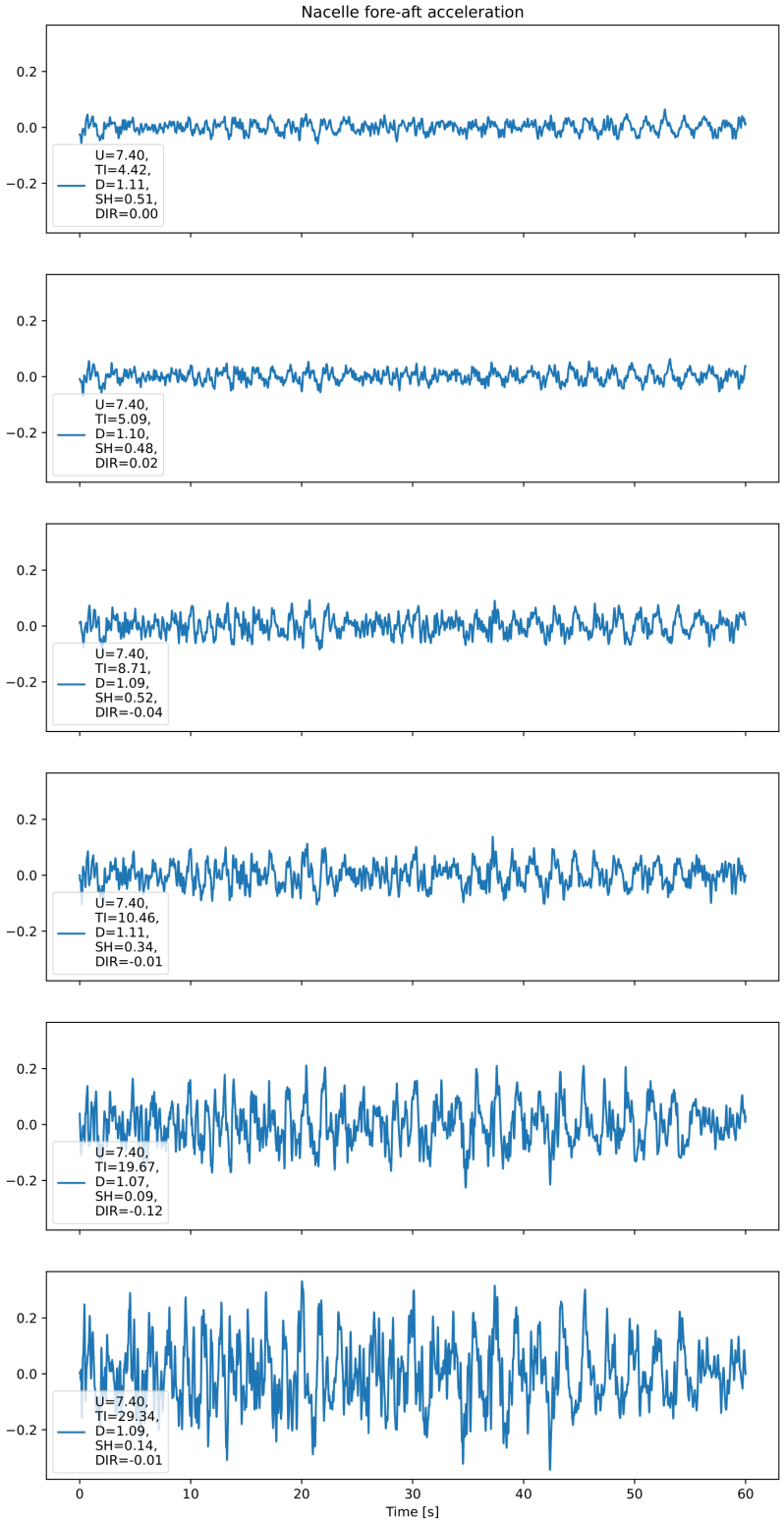

Generator speed
120
110
100
90
U=7.40,
TI=4.42,
D=1.11,
SH=0.51,
DIR=0.00
U=7.40,
TI=5.09,
D=1.10,
SH=0.48,
DIR=0.02
U=7.40,
TI=8.71,
D=1.09,
SH=0.52,
DIR=-0.04
U=7.40,
TI=10.46,
D=1.11,
SH=0.34,
DIR=-0.01
U=7.40,
TI=19.67,
D=1.07,
SH=0.09,
DIR=-0.12
U=7.40,
TI=29.34,
D=1.09,
SH=0.14,
DIR=-0.01
0
10
20
30
40
50
60
Time [s]

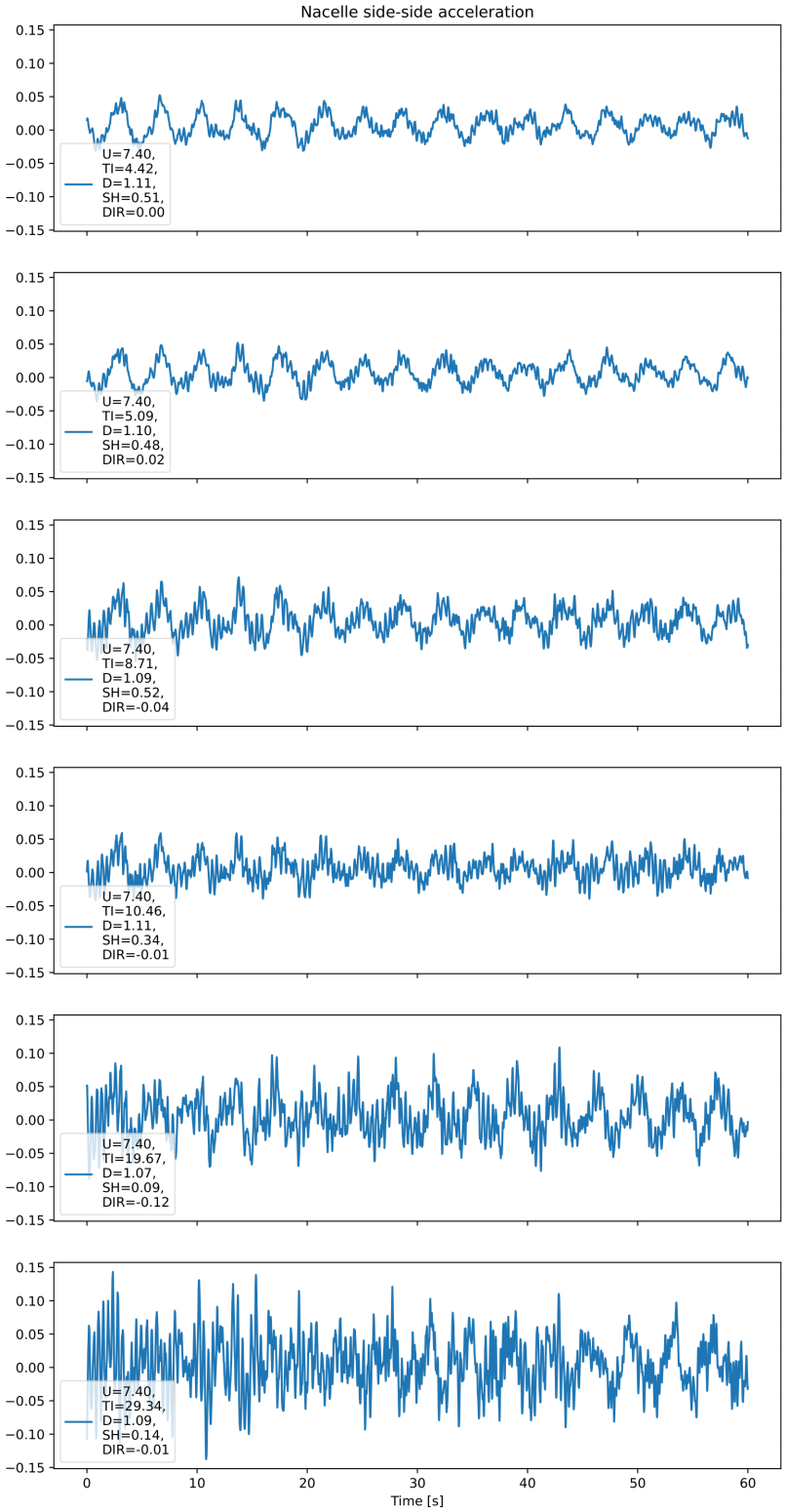

Blade 1 pitch angle
0.04
0.02
0.00
−0.02
−0.04
U=7.40,
TI=4.42,
D=1.11,
SH=0.51,
DIR=0.00
U=7.40,
TI=5.09,
D=1.10,
SH=0.48,
DIR=0.02
U=7.40,
TI=8.71,
D=1.09,
SH=0.52,
DIR=-0.04
U=7.40,
TI=10.46,
D=1.11,
SH=0.34,
DIR=-0.01
U=7.40,
TI=19.67,
D=1.07,
SH=0.09,
DIR=-0.12
U=7.40,
TI=29.34,
D=1.09,
SH=0.14,
DIR=-0.01
0
10
20
30
40
50
60
Time [s]

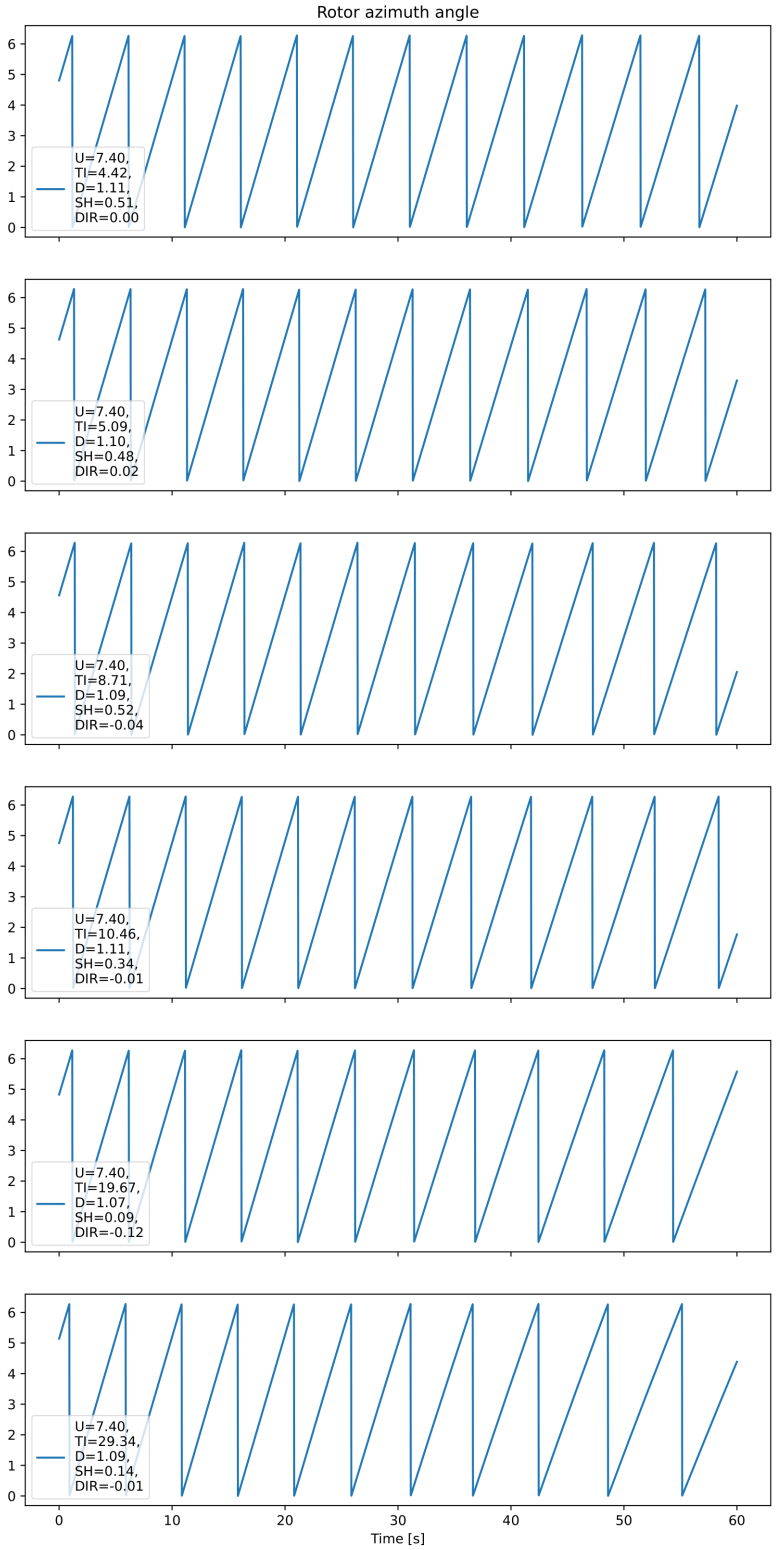

Blade 1 controller demanded pitch rate
0.04
0.02
0.00
−0.02
−0.04
U=7.40,
TI=4.42,
D=1.11,
SH=0.51,
DIR=0.00
U=7.40,
TI=5.09,
D=1.10,
SH=0.48,
DIR=0.02
U=7.40,
TI=8.71,
D=1.09,
SH=0.52,
DIR=-0.04
U=7.40,
TI=10.46,
D=1.11,
SH=0.34,
DIR=-0.01
U=7.40,
TI=19.67,
D=1.07,
SH=0.09,
DIR=-0.12
U=7.40,
TI=29.34,
D=1.09,
SH=0.14,
DIR=-0.01
0
10
20
30
40
50
60
Time [s]

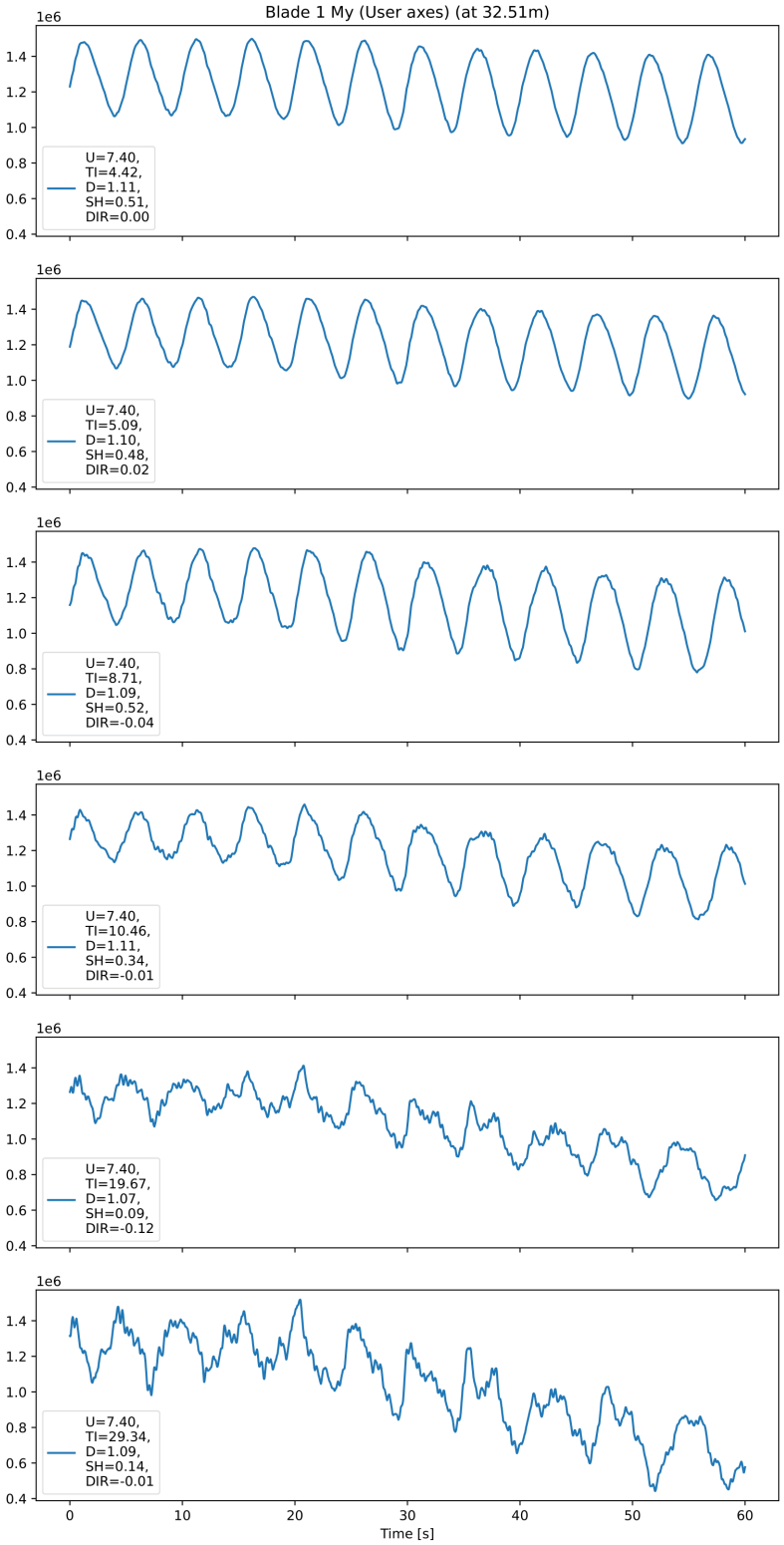

HSS torque
12000
10000
8000
6000
U=7.40,
TI=4.42,
D=1.11,
SH=0.51,
DIR=0.00
U=7.40,
TI=5.09,
D=1.10,
SH=0.48,
DIR=0.02
U=7.40,
TI=8.71,
D=1.09,
SH=0.52,
DIR=-0.04
U=7.40,
TI=10.46,
D=1.11,
SH=0.34,
DIR=-0.01
U=7.40,
TI=19.67,
D=1.07,
SH=0.09,
DIR=-0.12
U=7.40,
TI=29.34,
D=1.09,
SH=0.14,
DIR=-0.01
0
10
20
30
40
50
60
Time [s]


Electrical power
1e6
1.4
1.2
1.0
0.8
0.6
0.4
U=7.40,
TI=4.42,
D=1.11,
SH=0.51,
DIR=0.00
U=7.40,
TI=5.09,
D=1.10,
SH=0.48,
DIR=0.02
U=7.40,
TI=8.71,
D=1.09,
SH=0.52,
DIR=-0.04
U=7.40,
TI=10.46,
D=1.11,
SH=0.34,
DIR=-0.01
U=7.40,
TI=19.67,
D=1.07,
SH=0.09,
DIR=-0.12
U=7.40,
TI=29.34,
D=1.09,
SH=0.14,
DIR=-0.01
0
10
20
30
40
50
60
Time [s]


Nacelle fore-aft acceleration
0.2
0.0
−0.2
U=7.40,
TI=4.42,
D=1.11,
SH=0.51,
DIR=0.00
U=7.40,
TI=5.09,
D=1.10,
SH=0.48,
DIR=0.02
U=7.40,
TI=8.71,
D=1.09,
SH=0.52,
DIR=-0.04
U=7.40,
TI=10.46,
D=1.11,
SH=0.34,
DIR=-0.01
U=7.40,
TI=19.67,
D=1.07,
SH=0.09,
DIR=-0.12
U=7.40,
TI=29.34,
D=1.09,
SH=0.14,
DIR=-0.01
0
10
20
30
40
50
60
Time [s]


Nacelle side-side acceleration
U=7.40, TI=4.42, D=1.11, SH=0.51, DIR=0.00
U=7.40, TI=5.09, D=1.10, SH=0.48, DIR=0.02
U=7.40, TI=8.71, D=1.09, SH=0.52, DIR=-0.04
U=7.40, TI=10.46, D=1.11, SH=0.34, DIR=-0.01
U=7.40, TI=19.67, D=1.07, SH=0.09, DIR=-0.12
U=7.40, TI=29.34, D=1.09, SH=0.14, DIR=-0.01
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
0
10
20
30
40
50
60
Time [s]


Rotor azimuth angle
U=7.40, TI=4.42, D=1.11, SH=0.51, DIR=0.00
U=7.40, TI=5.09, D=1.10, SH=0.48, DIR=0.02
U=7.40, TI=8.71, D=1.09, SH=0.52, DIR=-0.04
U=7.40, TI=10.46, D=1.11, SH=0.34, DIR=-0.01
U=7.40, TI=19.67, D=1.07, SH=0.09, DIR=-0.12
U=7.40, TI=29.34, D=1.09, SH=0.14, DIR=-0.01
0 1 2 3 4 5 6
0 10 20 30 40 50 60
Time [s]


Blade 1 My (User axes) (at 32.51m)
1e6
1.4
1.2
1.0
0.8
0.6
0.4
U=7.40,
TI=4.42,
D=1.11,
SH=0.51,
DIR=0.00
U=7.40,
TI=5.09,
D=1.10,
SH=0.48,
DIR=0.02
U=7.40,
TI=8.71,
D=1.09,
SH=0.52,
DIR=-0.04
U=7.40,
TI=10.46,
D=1.11,
SH=0.34,
DIR=-0.01
U=7.40,
TI=19.67,
D=1.07,
SH=0.09,
DIR=-0.12
U=7.40,
TI=29.34,
D=1.09,
SH=0.14,
DIR=-0.01
0
10
20
30
40
50
60
Time [s]

Appendix B

All prediction plots

This appendix shows additional plots of the deep learning model's predictions compared to the ground truth of the target variable. These predictions are from the same prediction run as the values shown in chapters 5 and 6. The plots shown in those chapters were generated in the same way as the plots shown here. The files used here were sampled from the testing and prediction splits at regular intervals. The data sets were sorted on increasing wind speed.

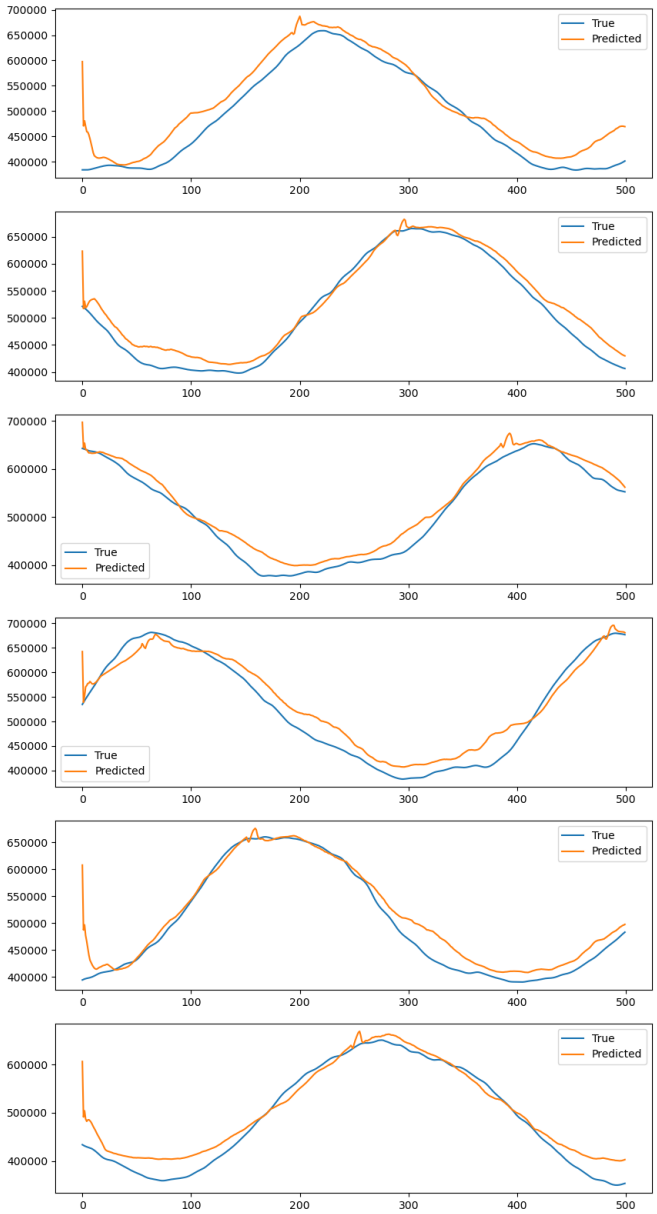


Figure B.1: File name: U3.4_TI7.23_D1.11_SH1.07_DIR-0.11_3

Figure B.2: File name: U3.9_TI11.51_D1.14_SH0.02_DIR0.07_1

Figure B.3: File name: U4.4_TI16.39_D1.12_SH0.26_DIR0.08_2

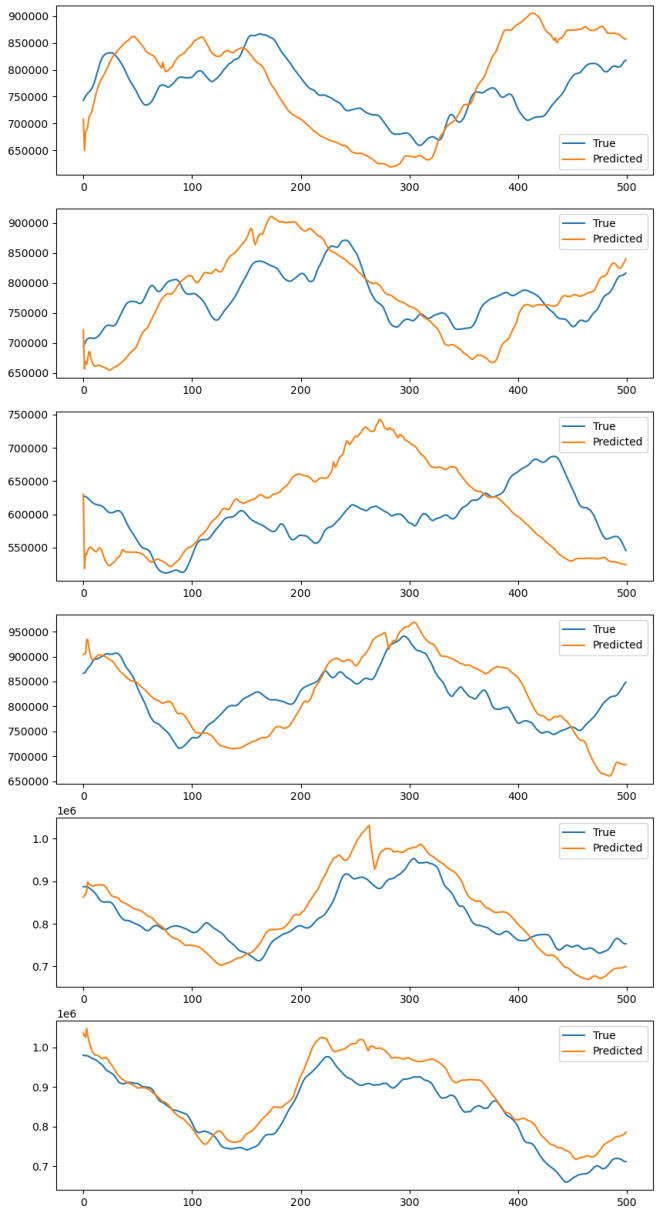


Figure B.4: File name: U5.3_TI16.15_D1.09_SH0.15_DIR-0.10_3

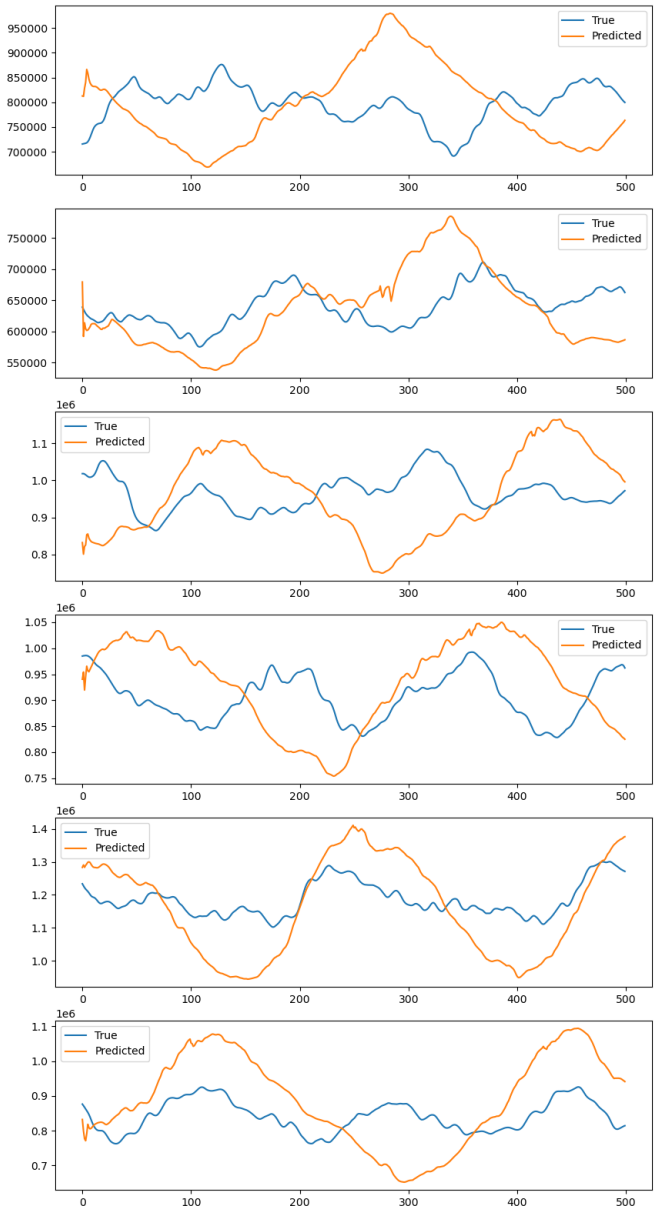


Figure B.5: File name: U5.6_TI19.84_D1.07_SH0.10_DIR0.10_1

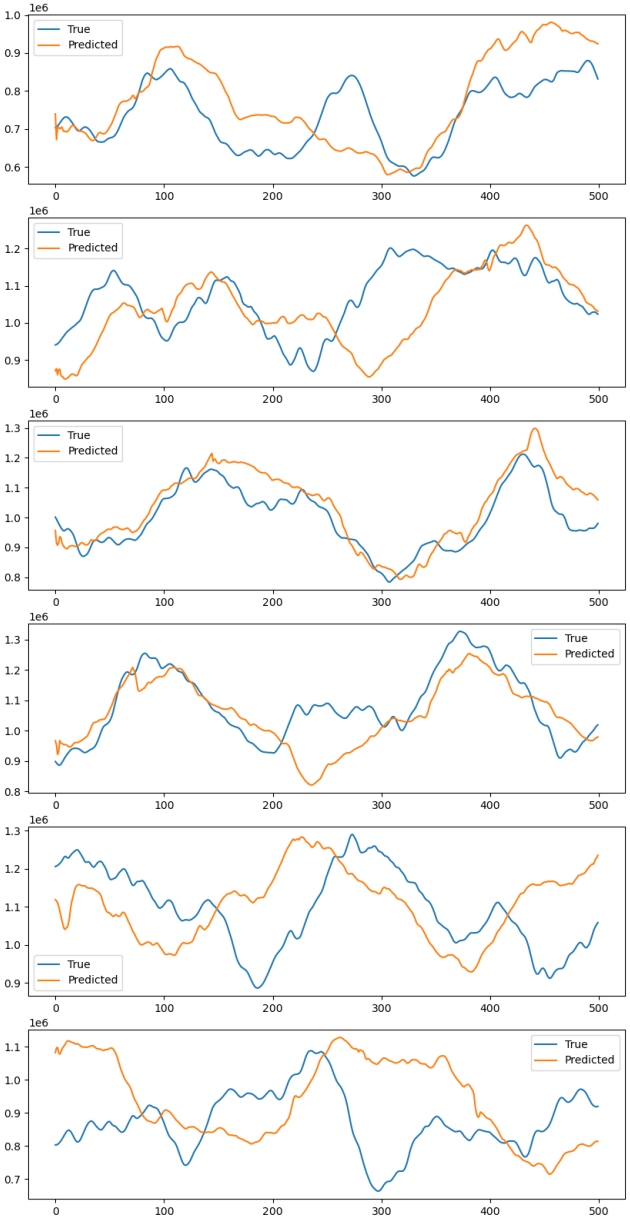


Figure B.6: File name: U5.9_TI29.35_D1.08_SH0.11_DIR-0.06_2

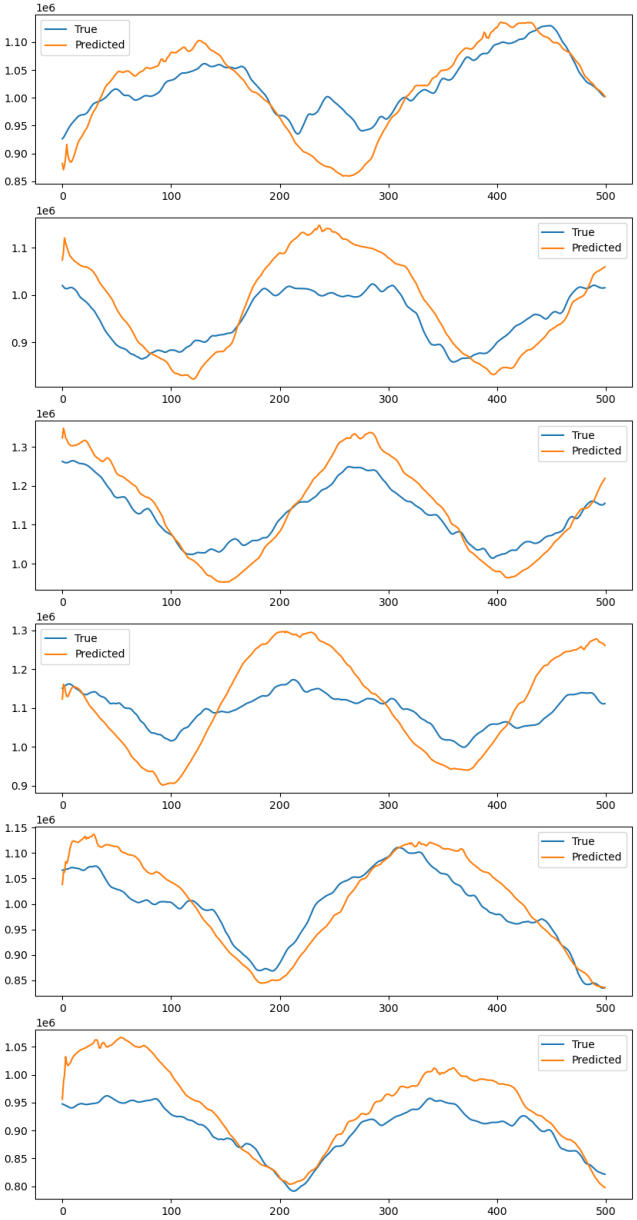


Figure B.7: File name: U6.4_TI10.70_D1.07_SH0.17_DIR0.02_3

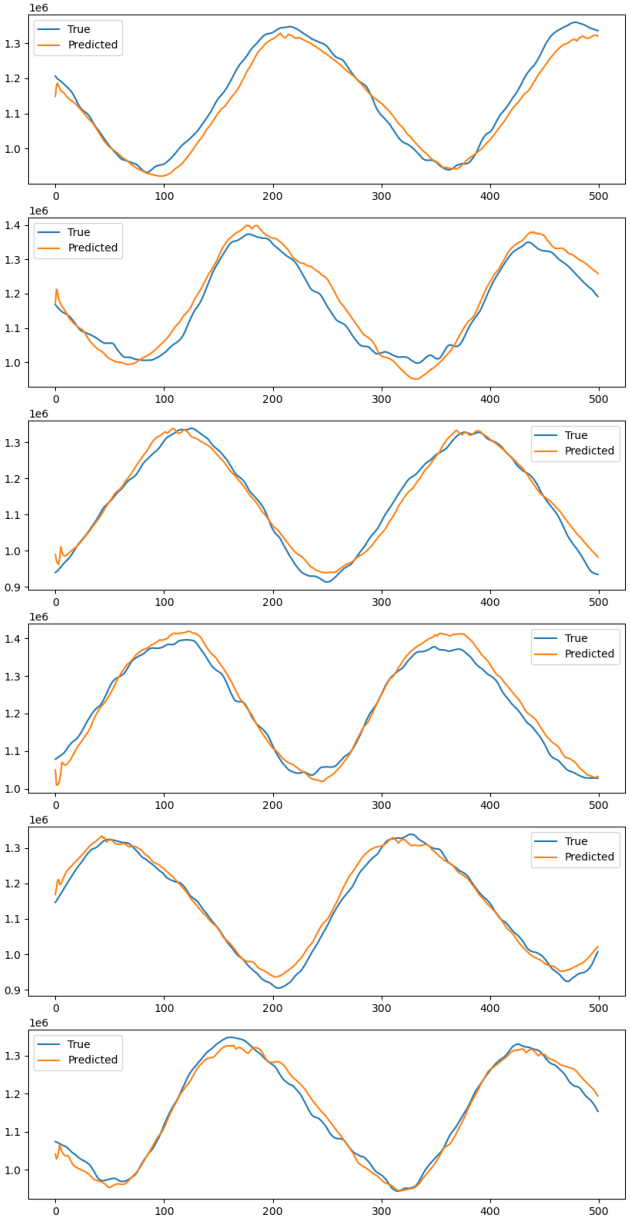


Figure B.8: File name: U6.9_TI6.03_D1.10_SH0.49_DIR0.01_2

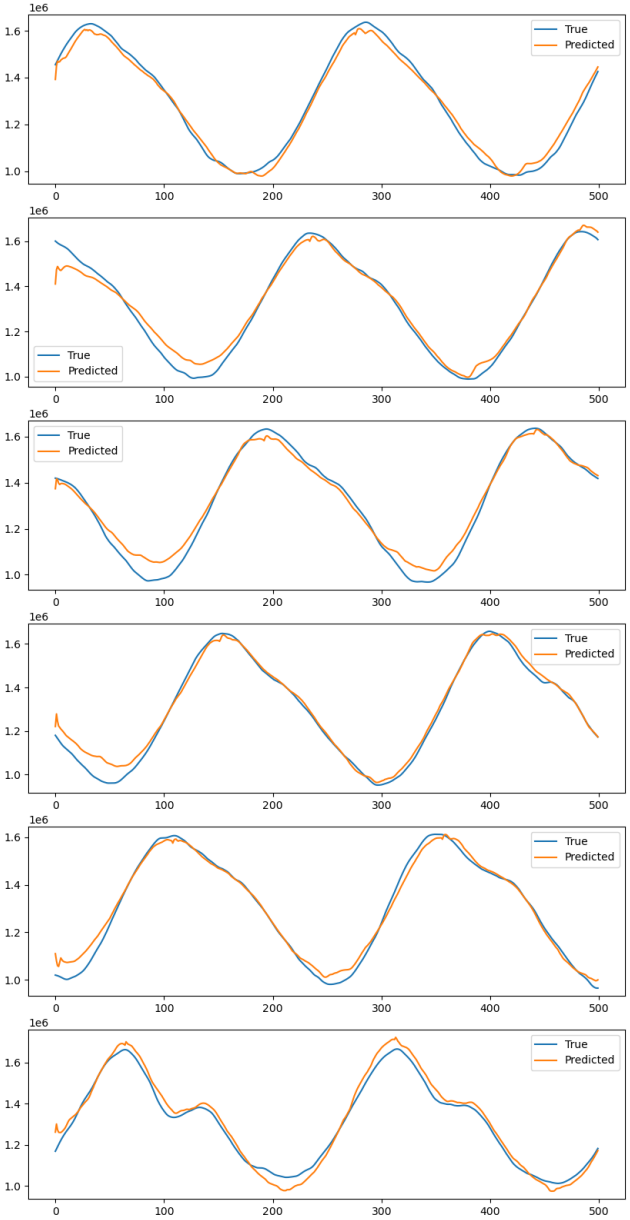


Figure B.9: File name: U7.9_TI4.68_D1.11_SH0.79_DIR-0.03_2

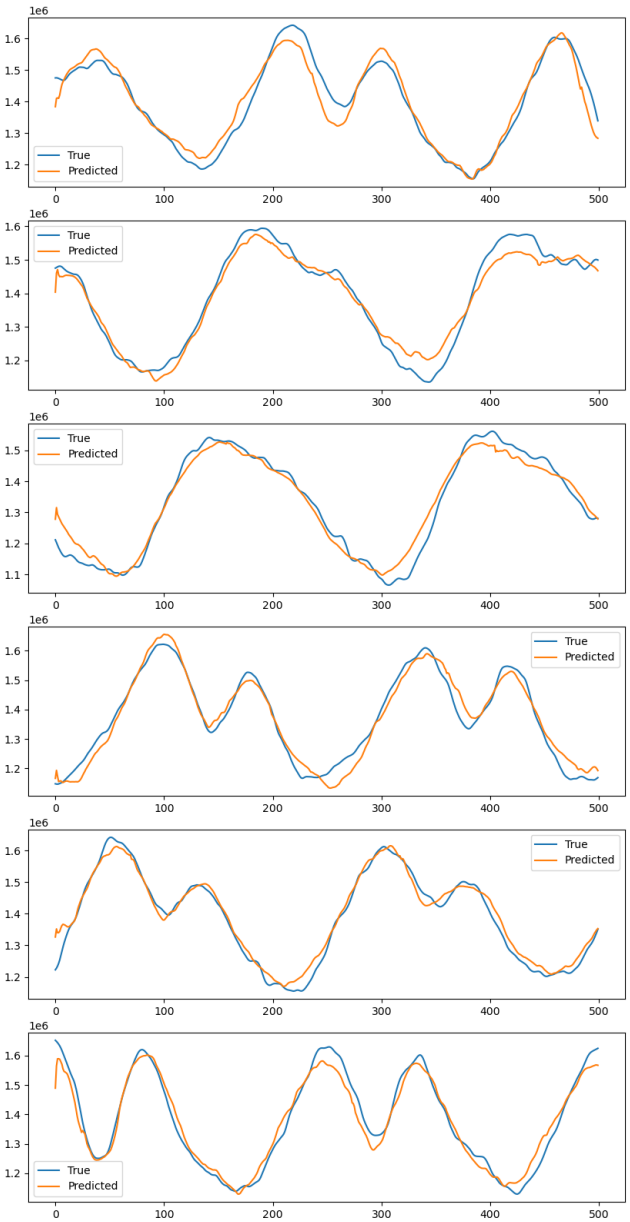


Figure B.10: File name: U8.9_TI7.94_D1.11_SH0.49_DIR0.01_3

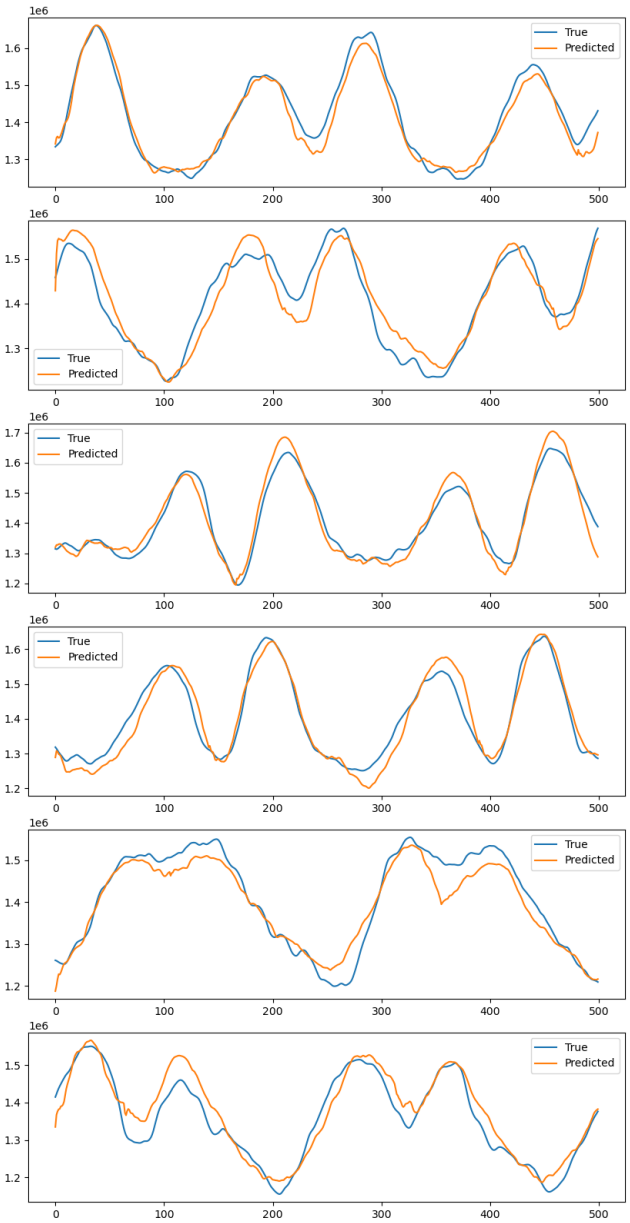


Figure B.11: File name: U9.8_TI7.61_D1.10_SH0.41_DIR0.01_2

Appendix C

Additional metric comparisons

This appendix shows a grid of pairwise kernel density plots, generated with the same procedure as the ones presented in chapter 6. The Wasserstein distance values are here calculated based on statistics engineered from the blades' pitch angle during the simulations. The procedure for generating these values is described in section 6.2.2.

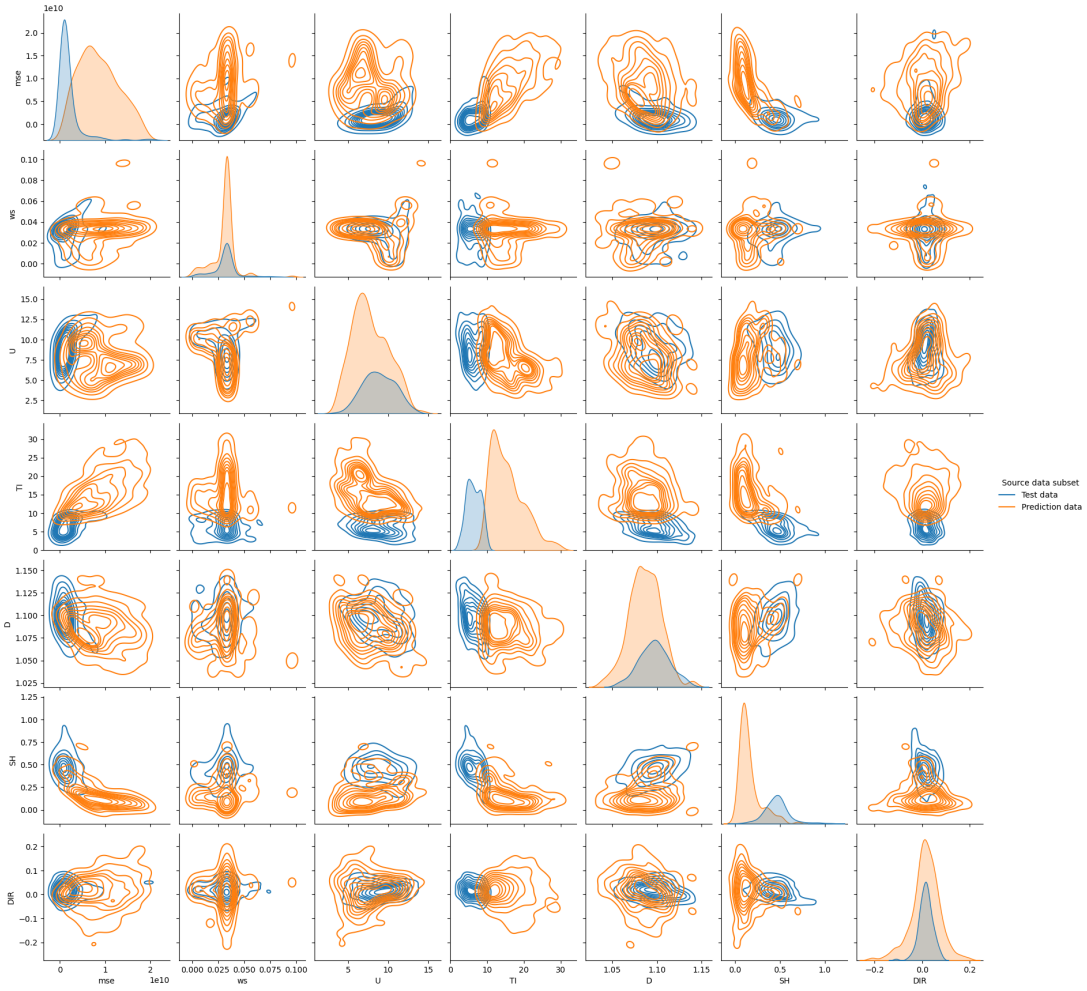
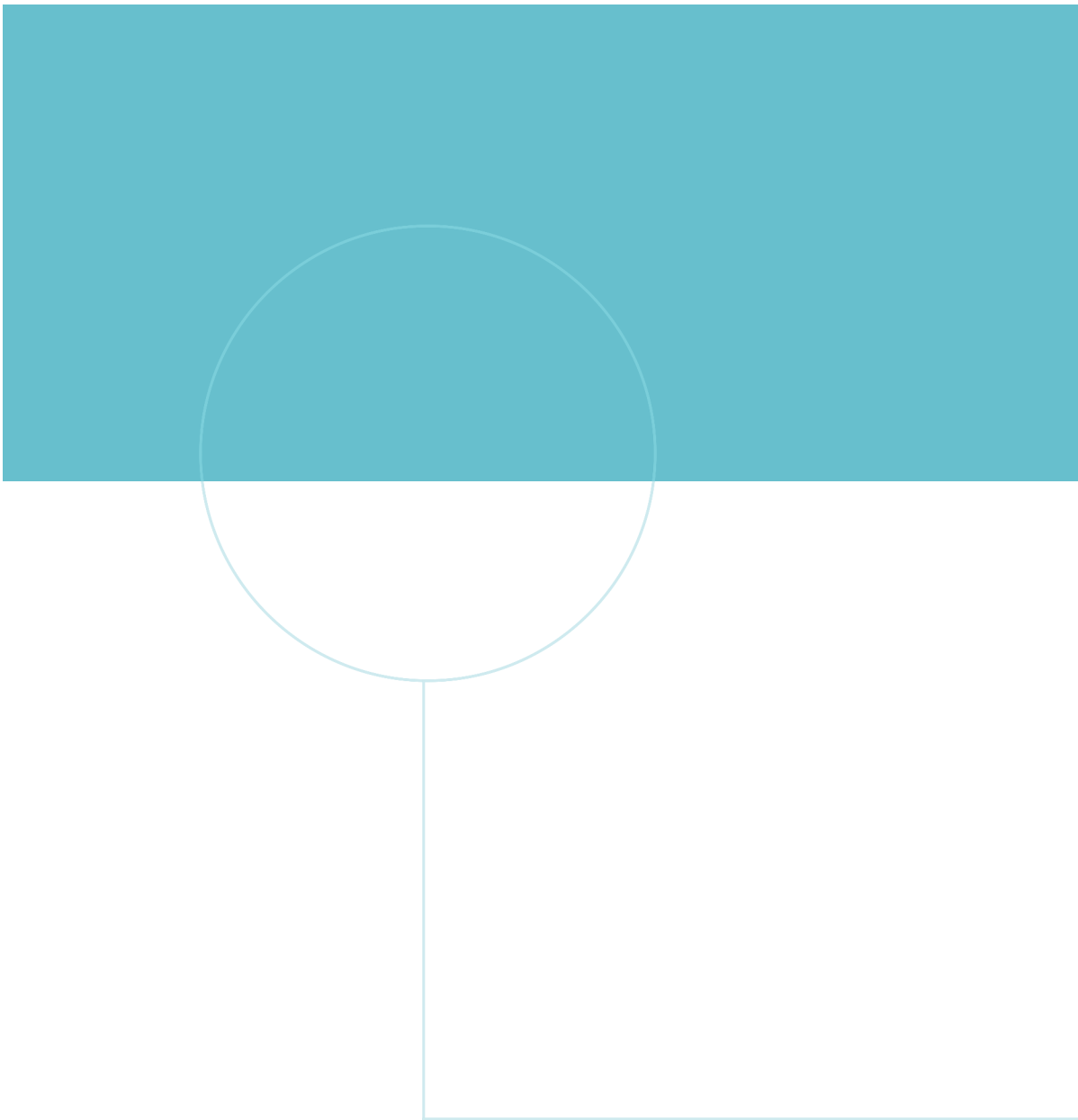


Figure C.1: The MSE from the model's prediction over a data file, that file's Wasserstein distance to the training data, and the file's initialization parameters plotted pairwise against each other to show their distributions. Along the diagonal are the kernel densities of each of these metrics, the other plots are two-dimensional kernel density plots.



NTNU
Norwegian University of
Science and Technology

SINTEF