Modeling the Stochastic Evolution of Wind Profiles Using Neural Networks

Fernando Flores Vogt-Kielland

Thesis submitted for the degree of
Master in Applied Computer and Information Technology - ACIT
(Data Science)
60 credits

Department of Computer Science
Faculty of Technology, Art and Design

Oslo Metropolitan University — OsloMet

Spring 2023

Modeling the Stochastic Evolution of Wind Profiles Using Neural Networks

Fernando Flores Vogt-Kielland

Modeling the Stochastic Evolution of Wind Profiles Using Neural Networks

http://www.oslomet.no/

Printed: Oslo Metropolitan University — OsloMet

Preface

Looking back to fall 2021 when the available topics were published, I remembered looking through the list and finding many interesting subjects. After discussing some of them with their associated advisor, I asked to be a part of this project because of two major reasons. The first being my interest in the topic of renewable energy and the potential to have even a small contribution in a field I strongly believe in. The second being the prospect of working alongside with my favorite teacher and TA from the first semester of the program, a pair I grew to like and admire during those first months.

When this project started at the beginning of 2022, I didn't grasp the magnitude of the task ahead. It's been a roller-coaster since then, with ups and downs but generally an enriching and great experience, that could have turned out completely different. In that regard, I would like to thank OsloMet for allowing me to take this Masters program and presenting me the possibility of taking this project. Additionally, I would like to thank FINO1 for providing the information and clearing doubts along the way. Moreover, I would like to thank my advisors for sharing their knowledge, guidance, experience, and interest through this last year and a half (two years for both Pedros and Anis). I would like to thank my family and friends as well for their unconditionally support and encouragement. Finally, I would like to thank my wife for being my rock and inspiring me every single day for the last years. I believe I wouldn't be able to be were I am today without these great people.

Fernando Flores Vogt-Kielland
Oslo 2023

Abstract

The aim of this project is to profile a multivariate set of wind speeds corresponding to different heights and composing a vertical speed profile in time. Most importantly, these analysis will be done over a data-set with a $1Hz$ sampling frequency, which is uncommon for the industry's standards, using typically 10 or 15 minute averages. The purpose of for such small sampling frequency is to augment the knowledge over current models under different circumstances. To that end, we will use measurements collected at off-shore platforms at the North Sea FINO1 platform, made available to OsloMet for research purposes.

The project is divided into two parts. First, we conduct an analysis of the time and spatial patterns of the set of wind speeds using basic statistical and numerical tools. Second, the implementationtation of Deep Learning architectures using the findings from the analysis conducted on part one. These architectures will then be trained and validated with the data at hand.

The final goal is to provide a comparative analysis of how these AI-based models capture the fundamental statistical and dynamical features of wind speed profiles and retrieves predictions of future profile states. Such insights could be of interest for better predicting several aspects related to the design, operation, and maintenance of the next generation of wind turbines. Particularly in the estimation of fatigue loads, which is a major aspect in wind farm monitoring and maintenance protocols.

Contents

List of Figures

List of Tables

List of acronyms

Through this thesis we will be utilizing some acronyms. The use of the acronym will be exchanged with its definition sporadically in an attempt to facilitate reading and avoid monotony. The following list contains the acronyms we used and what they stand for.

Acronym	**Meaning**
ANN	Artificial Neural Network
AI	Artificial Intelligence
DL	Deep Learning
GAN	Generative Adversarial Networks
RNN	Recurrent Neural Network
NN	Neural Network
LSTM	Long Short-Term Memory
PCA	Principal Component Analysis
PC	Principal Component
PDF	Probability Distribution Function

Chapter 1

Motivation and Problem Statement:

Modeling the Stochastic Evolution of Wind Profiles

From cavemen adopting clothes and developing shelter to protect themselves from the elements; passing through the age of exploration, when different cultures travelled long distances by sea with ships powered by wind (and human sweat) in search of new territories; to today's efforts to efficiently extract electrical power from wind turbines, it is safe to say that wind has played a considerable role in human history. With the rise of importance of renewable energy sources, wind is probably under a bigger spotlight than ever. However, pivotal as it is, do we understand well enough this elemental power?

With the increased public awareness regarding climate preservation and the decrease in cost of constructing/operating it, the usage of Wind Energy has increased heavily in the last years ('Wind Power', 2022). Along with the increased usage, comes the task of maximizing the benefits we can reap from it and here is where the challenge lies. Wind being uncontrollable and to some extent, unpredictable, creates a situation where it is extremely difficult to maximize its benefits. Hence, our desire to find a work frame that can be of help in the understanding of wind by modelling the randomness of its behavior.

Our aim goes beyond increasing our understanding of wind profiles. We are motivated by the possibility of making an impact on a subject we believe is vital to preserving/improving this planet's future, hence we aim to produce a model that can be easily interpreted and applied into practical scenarios.

As renewable energies become more important, wind energy has gone through an interesting developing process. Starting with On-Shore wind turbines and progressively moving far away from land into the sea to floating Off-Shore wind turbines; wind power seems to be moving in a clear direction. As described by Equinor's Head of Wind Turbine Technology, Kristian Holm, on a webinar: "Evolution is turning upside down by us moving back into the sea" ('Data Science Workshop - Wind Energy Analytics, a practical approach', 2021). With this in

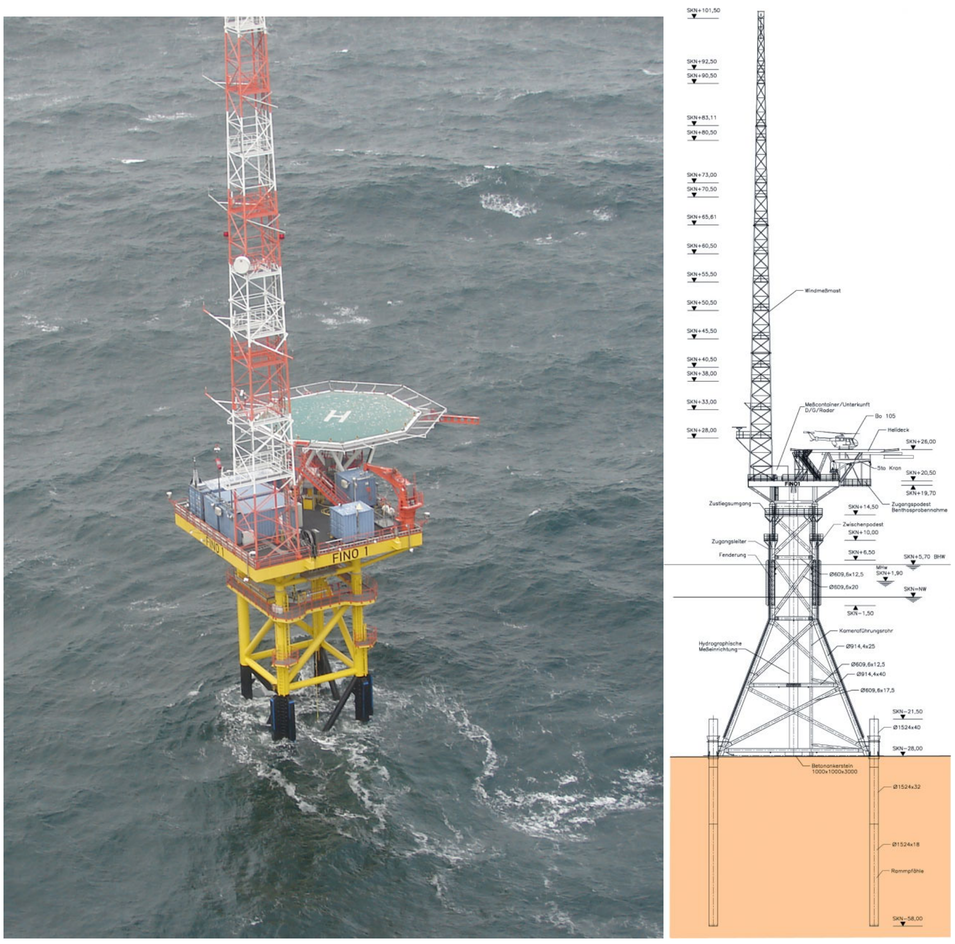


Figure 1.1: (Left) Aerial photograph of FINO1. (Right) Schematics of FINO1's equipment [1].

mind, we were fortunate to secure information from FINO1, a research platform (pictured in Fig. 1.1) located in the North Sea approximately 45 kilometers north of the German territory of Borkum. The platform is equipped with comprehensive measuring equipment from research and university institutes, operated with no permanently stationed personnel, and monitored centrally from onshore ('The FINO1 project', 2022).

For Off-Shore wind, like the one we are going to be analyzing, there is not a lot that can affect its behavior. There are no buildings in its path or any topographic anomalies that might change its course. Nonetheless, we will encounter some phenomena that affect how wind behaves, like drag and turbulence. Drag is a type of friction present in the interaction between two fluids or a fluid and a solid object, in this case between the wind and the ocean surface ('What is Drag?', 2021). Turbulence is a blanket term used to describe many different physical phenomena, which exhibit the common characteristics of disorder and complexity (Rieutord et al., 2006).

Contrary to some of the current and previous work in the area, done

[1]Left image taken from https://www.fino1.de/de/medien/fotos.html.
Right image taken from https://www.fino1.de/en/about/design.html

using wind speed averages at 10-minute intervals, we aim to run our analysis and create our model at a 1 second intervals resolution. As a good friend used to say, "The greatest Giant slayer in all mythology is Sir Average", so having the data at this resolution provides us with the opportunity of analyzing wind profiles in a level of detail that might provide valuable insights that get lost in the commonly used industry averages. However, with the added information, detail and opportunities, come extra challenges. These challenges mostly lie in not having a "buffer" that can smooth out the phenomena (such as turbulence) affecting wind and its impact on the recorded wind speed.

Additionally, as per why we are focusing on predicting a complete wind profile instead of just an average time speed, is the trend in size for wind turbines. As time passes, exploiting the direct link between the size of the rotors and the amount of energy produced has been the trend. With technology enabling the construction of bigger and bigger blades, wind turbines are bigger than ever and most likely they will continue to grow with the coming years ('Wind Turbines: the Bigger, the Better', 2021). Associated with this increase in size, wind turbines are getting exposed to larger wind profiles than before, which in turn can be related to loads and torques not experienced by wind turbines of smaller dimensions.

So, with the aim of modeling the stochastic evolution of wind profiles we are looking to address two main research questions in this thesis:

1. What are the main statistical and dynamical features of wind profiles?

2. How effective is an AI-based approach to model and predict such evolution of wind profiles?

In sight of the previously stated, we aim to create a model capable of capturing this stochastic behavior at this reduced time scale. With this model we aim to analyze wind profiles at a multivariate level and not just the average speed of over the different measured heights, thus increasing the current understanding and enabling predictions of future wind states. We believe that if successful, these predictions can be of great relevancy in many tasks such as: estimating the potential energy output of a wind farm, maintenance planning, and even the design of future equipment to name a few.

Chapter 2

Background and Related Work

2.1 Basic statistics

We will begin by establishing some fundamental Statistical concepts, to be able to build upon these definitions to describe more complex parts of our work.

Throughout this project we will be making inferences upon Wind Speed data. For the most part, when dealing with Wind, most (if not all of the) variables associated are continuous, which presents a particular challenge. Opposed to discrete variables, continuous variables can take an infinite number of values, thus making the probability of the variable taking a particular value equals zero. Being able to determine probabilities from our data is crucial to our success, and to this end, we will be using some Probability Density Functions (PDF). The PDF of a continuous variable describes the likelihood (or probability) that a random value of the variable is found within a specified range of values ('PennState: STAT 414 | Introduction to Probability Theory', 2022), thus enabling us to achieve our purpose.

Most data can be fitted into a known Distribution, which will be discussed more in Section 2.3.1. As observed from Fig. 2.1, a Histogram is a great visualization tool to begin this endeavor with. Histograms are graphical representations of the distribution of the data they represent. They are constructed by dividing the entire range of available data into "bins" of equal size and plotting as height the frequency on which the observations appear on each bin. From this visualization we can begin observing some more elements from Basic Statistics we are building upon, like: Mean, Standard Deviation, Skewness and Kurtosis.

2.1.1 The first four moments of Statistical Distributions and their meaning

In mathematical terms, the **Mean** is the result of the sum of all the values of a finite set divided by the number of values. This concept applies as well when referring to samples, as in the sum of the value of the samples divided by the number of samples. On an evenly distributed (normal)

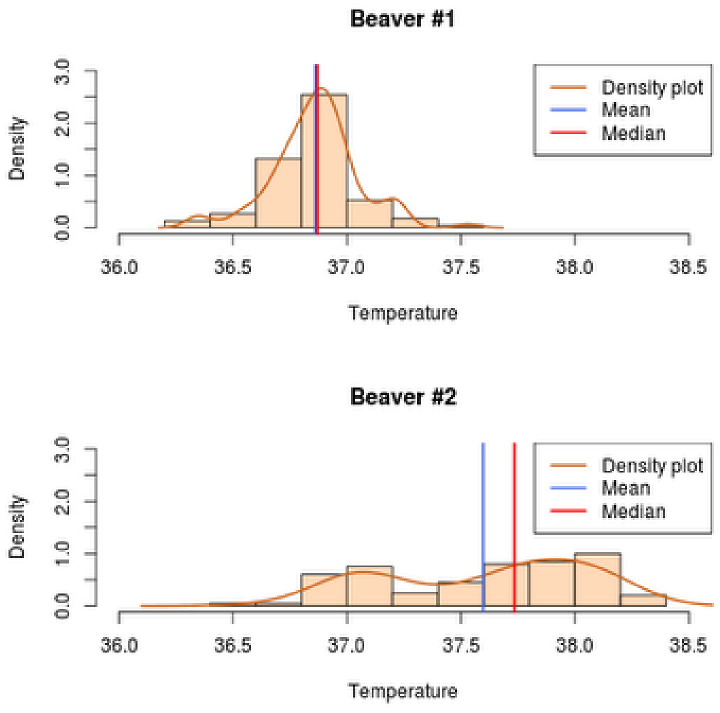


Figure 2.1: Depictions of Histograms and how they aid in distribution fitting [1].

set of data, the mean can be thought of as a single value that is most representative of the whole set. This is especially true of a set with a low Standard Deviation. The mean is represented in Fig. 2.1 with the red line. Often, associated with the concept of Mean we find Mode and Median as well. Briefly, Mode is the value that occurs the most in the data-set and Median is the value that separates the lower half from the lower half of the sample.

Standard Deviation (SD or σ) is a measure of how close the values in a set are to the Mean. Calculated by the square root of the sum of the squared difference between the observed values and the mean, divided by the number of observations. The equation for SD can be observed in Equation 2.1 where: X is the sample, μ the mean, and N is the number of samples.

$$\sigma = \sqrt{\frac{\sum (X-\mu)^2}{N}} \tag{2.1}$$

The lower the SD of a set, the closer the values of the set are to its Mean. In Fig. 2.1 we observe that figure a)'s SD is lower than figure b)'s as the values are closer to the mean, resulting in a narrower looking PDF.

Since the SD is calculated by the squared value of the difference between the values and the Mean, it means that it doesn't take into

[1]Image taken from: https://www.r-bloggers.com/2012/10/adding-measures-of-central-tendency-to-histograms-in-r

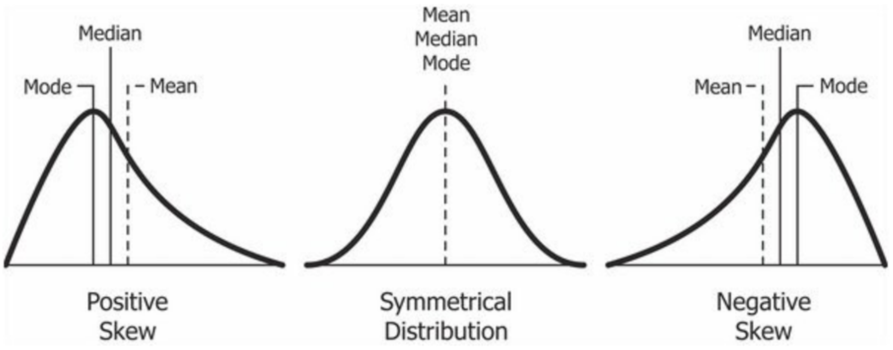


Figure 2.2: Depiction of how different Skewness values reflect on the shape of the PDF [2].

consideration if the values are higher or lower than the Mean, just how far from it they are. To evaluate the symmetry of the data around the mean, we need to calculate its **Skewness**. This measurement can result in either positive or negative values, or zero, indicating to which end of the data's range lies the concentration of representative data and the outliers as well. Visually, Skewness is represented by how centered the data is in the histogram/PDF as pictured in Fig. 2.2.

Finally, **Kurtosis** is the fourth moment from a statistical point of view (the first three being the previous elements discussed) ('Wolfram MathWorld: Kurtosis', 2022). It is a measurement of how likely extreme values can occur in the data-set. The higher the value, the most likely the occurrence of extreme values. As observed in Fig. 2.3, the higher the Kurtosis (red line), the higher the tails of the PDF, hence the more likelihood of extreme values.

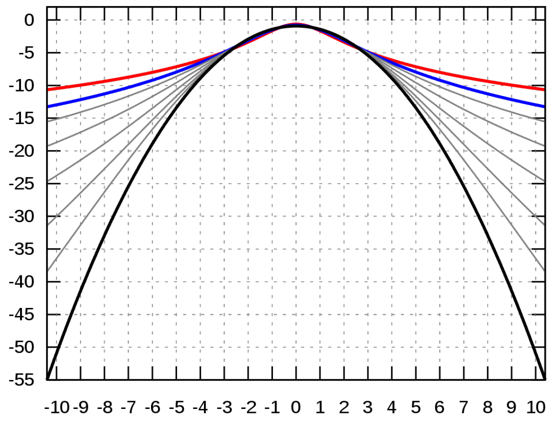


Figure 2.3: Depiction of Distributions with different values for Kurtosis and their impact on the shape of the PDF [3].

[2]Image taken from: https://en.wikipedia.org/wiki/Skewness
[3]Image taken from: https://en.wikipedia.org/wiki/Kurtosis

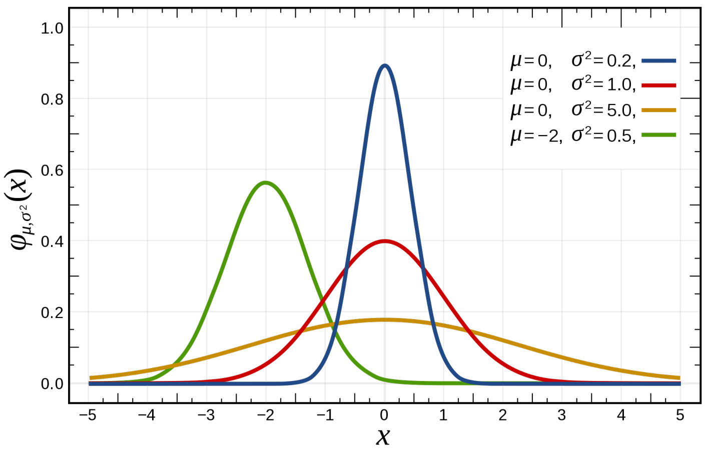


Figure 2.4: Depiction of Normal Distributions with different Mean and Variance [4].

2.1.2 Important Distributions for Wind Analysis

The **Normal Distribution** (also known as Gaussian Distribution) is one of the most important distributions in mathematics, probability, and statistics, both due to its occurrence in many natural and social phenomena and its versatility due to the Central Limit Theorem. This theorem states that even when variables are not normally distributed, under certain conditions, their sums and averages might approximate a Normal Distribution (Devore et al., 2012). The equation that describes a Normal Distribution can be observed in Equation 2.2 and a visual representation can be observed in Fig. 2.4 for distributions with different Mean and Variance ($StandardDeviation^2$).

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2 / 2\sigma^2} \tag{2.2}$$

The **Weibull Distribution** was popularized as such in 1951, and used in many engineering, communication and weather applications. It is described as Equation 2.3 for X equal or larger than zero, where λ is referred as the scale parameter and k as the shape parameter of the distribution. A visualization can be seen in Fig. 2.5. As we will observe in Section 2.5 this is probably the most used PDF used while fitting Wind Speed, as the way most Wind Speed is distributed tends to resemble this distribution.

$$f_X(x;\lambda,k) = \frac{k}{\lambda}\left(\frac{x}{k}\right)^{k-1} e^{\left(-(x/\lambda)^k\right)} \tag{2.3}$$

[4]Image taken from: https://en.wikipedia.org/wiki/Normal_distribution

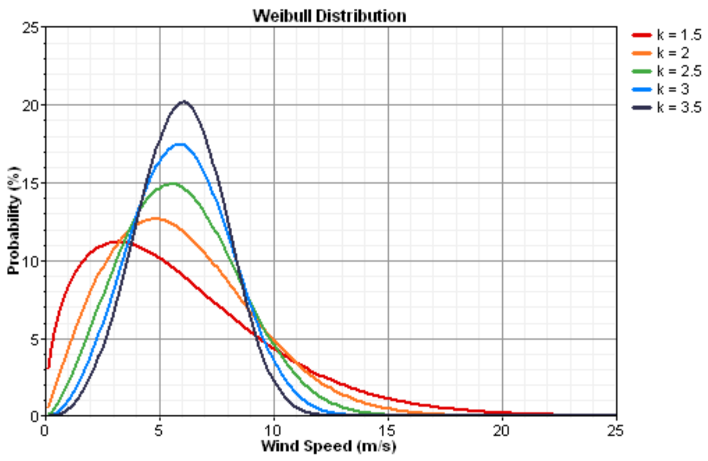


Figure 2.5: Depiction of Weibull Distributions with different parameters [5]. Regardless of the parameters, the curves are characterized for their asymmetric and positive skewed behavior.

2.2 Wind speed data

Even though they are often studied together, for this project we are concentrating solely on wind speed, leaving analysing its direction for a potential future project. On a general approach, Wind Speed data can be captured with relative ease. As stated by Harris "Apart from sonic and 3-axis propeller anemometers, which have mainly been used to capture atmospheric turbulence for specialized research, virtually all the past and present wind data have been acquired with a combination of an instrument which measures the wind speed (i.e. the scalar modulus of the two-dimensional wind vector), and a separate wind vane to measure the direction." (R. I. Harris & Cook, 2014).

As per standardization standards regarding wind data, we can refer to the World Meteorological Organization for our intents and purposes. Since 1950, WMO has acted as the United Nations' specialized agency for Meteorology and other related geophysical sciences. Its main purpose is to support with several Meteorological and Hydrological services in their commitment to reduce the risk of disasters, mitigate climate change, and sustainable development ('World Meteorological Organization', 2022). To achieve this task WMO promotes "the creation of standards for observation and monitoring in order to ensure adequate uniformity in the practices and procedures employed worldwide and, thereby, ascertain the homogeneity of data and statistics" and "the establishment and maintenance of data management centres and telecommunication systems for the provision and rapid exchange of weather, climate and water-related data" ('World Meteorological Organization - Library', 2022), hence our interest in it.

[5]Image taken from: https://www.homerenergy.com/products/pro/docs/latest/weibull-distribution.html

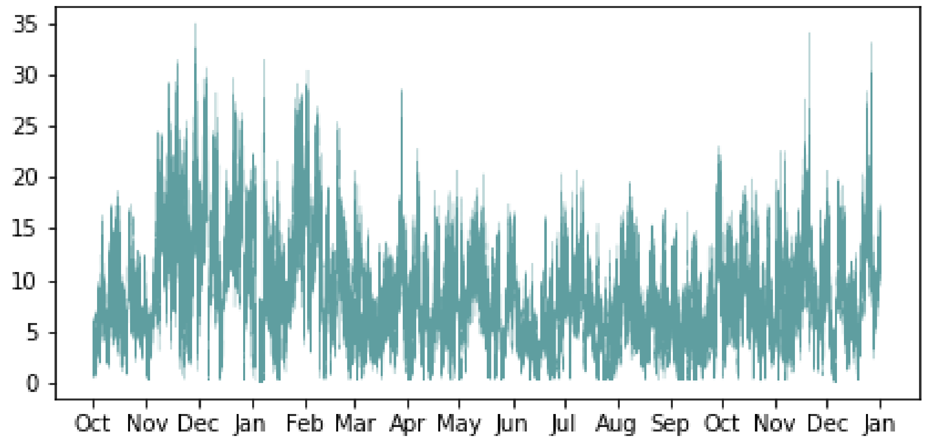


Figure 2.6: Depiction of average wind speed over a year at FINO1. A stochastic behavior can be observed along the months with no clear pattern or seasonality.

Some of the definitions by WMO which we will be referencing through this project can be cited straight from their library: "More and more applications also require information on the variability or gustiness of the wind. For this purpose, three quantities are used, namely the peak gust and the standard deviations of wind speed and direction. Averaged quantities are quantities (for example, horizontal wind speed) that are averaged over a period of 10 to 60 min. This chapter deals mainly with averages over 10 min intervals, as used for forecasting purposes. (...) Averaging periods shorter than a few minutes do not sufficiently smooth the usually occurring natural turbulent fluctuations of wind; therefore, 1 min “averages” should be described as long gusts. Peak gust is the maximum observed wind speed over a specified time interval. With hourly weather reports, the peak gust refers to the wind extreme in the last full hour. Gust duration is a measure of the duration of the observed peak gust. The duration is determined by the response of the measuring system. Slowly responding systems smear out the extremes and measure long smooth gusts; fast response systems may indicate sharp wave-front gusts with a short duration. (...)Wind speed should be reported to a resolution of 0.5 m s–1 or in knots (0.515 m s–1) to the nearest unit, and should represent, for synoptic reports, an average over 10 min." (‘World Meteorological Organization - Library’, 2018).

As a preview of the challenge in front of us, we have generated a couple of figures that help visually illustrate some of the properties of our data. In Fig. 2.6 we can observe our data plotted as a time series. There we can observe our data’s stochastic behavior, without any discerning tendencies or patterns, which as we will evidence in Section 2.5 is the source of many studies and a lot of work. Complementing this preview we have Fig. 2.7 which depicts the distribution of our data. As discussed in Section 2.1.2, the distribution closely resembles a Weibull PDF as it was expected.

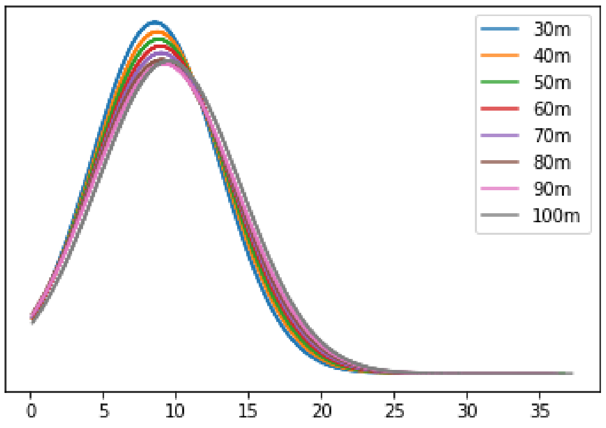


Figure 2.7: FINO1 wind speed distribution per height, where it can be observed that all heights PDF similar to the Weibull distribution. A slight variation of the statistical moments between each height can also be observed.

2.3 Statistical Learning Background

Statistical Learning can be described as a series of methods and tools used to solve problems by understanding the data behind them. Generally, these tools and methods can be classified as Supervised and Unsupervised (James et al., 2019). The biggest differences between these two are: the availability of labeled data and the specific desired outcome.

Supervised Learning deals with the challenge of making a prediction from the available data. This prediction is estimated by analyzing how all the available variables in the data influence (or not) the response variable. In this kind of problem, there is usually one response variable for each observation which is referred to as labeled data. Thus, the result can be either an inference (predicting a result from variables within the observed range but not measured) or extrapolation (predicting the result from variables outside the observed range).

On the other hand, Unsupervised Learning problems are characterized for the lack of a response variable. Hence, the challenge with this kind of learning is finding and understanding the relationships between the available variables.

2.3.1 Fitting Distributions

As discussed before, most data can be fitted into a Function. The challenge now lies in how to do it reliably. There is two main approaches to it: the Maximum Likelihood Estimate (MLE) method and the Least Squares Estimate (LSE) method. Both methods aim for the same goal: to determine which function our data-set resembles the most. However, they use opposite methods to achieve their purpose. Where MLE is a (spoilers on the name) maximization of parameters problem, LSE aims to minimize error within the resulting parameters.

The process of MLE entails figuring out a likelihood function that resembles the assumed/desired distribution and optimizing its parameters to maximize the likelihood that the data resembles the distribution. The parameter optimization is achieved through calculating the point where the derivative in function of the parameters equals zero (Greene, n.d.). In comparison, LSE is a process where an initial estimation of the parameters of a function improves through iterations. These iterations aim to minimize the sum of the squared residuals between the outcome variable and the projection with the current parameters (James et al., 2019).

Both methods have proven to be effective under the right circumstances. LSE is better suited for linear distributions, while MLE has proven more effective when we are dealing with a small sample and the data has fewer failures. The reason behind this is the fact that MLE uses more of the information in the data in its calculations (Genschel & Meeker, 2010).

2.3.2 Linear Time Series analysis

As stated by Shumway and Stoffer (2000) analysing data that has been observed at different points in time introduces problems related to correlations, thus restricting the applicability of many traditional statistical methods that assume the observations are independent and identically distributed. In sight of this problem, many methods and techniques have been developed to handle this correlations and on this section we are trying to provide an overview of the most relevant for our project.

We can begin with probably the most elementary of this analysis which is the Autorregresions, these analyses create predictions for the current value based on the past values in the series. For this analysis, we are not dealing with mean values, but exploiting any possible correlation between the values themselves. Basically, we want to predict the current value of the series based on its value in the past. To better define this relation we must understand the Autocorrelation Function (ACF) and the Partial Autocorrelation Function (PACF). ACF considers the correlation between a value y_t and a value y_{t-x} taking into account all the values that are between t and $t-x$, whereas PACF considers only the effect that $t-x$ has over t. Formally PACF can be defined as in Equation 2.4.

$$\frac{\text{Covariance}(y, x_p|x_1, x_2...x_p)}{\sqrt{\text{Variance}(y|x_1, x_2...x_p)\text{Variance}(x_p|x_1, x_2...x_{p-1})}} \tag{2.4}$$

Autorregresive models rely on using the lags (previous periods) with the largest PACF associated for predictions to avoid problems like overfitting ('Partial Autocorrelation Function', 2022). This model can be generalized as in Equation 2.5, with p being the order of the autorregresion and c a constant.

$$\hat{y}_t = c + \phi_1 y_1 + \phi_2 y_2 + ... + \phi_p y_p \tag{2.5}$$

Moving on to Moving Averages. The basis for this analysis lies in taking advantage of the correlation between the mean of neighboring observations by replacing the series with a moving average i.e. $v_t = \frac{1}{3}w_{t-1} + w_t + w_{t+1}$. This results in a smoother series that reduces noise and de-emphasises periodic behavior. Another application of the Moving Average may be to rely on the average value of the series to make predictions and adjust the prediction based on the error of the last prediction. A generalized equation for this approach can be as in Equation 2.6, where w stands for the error between the prediction and the average, and q for the order (how many previous periods are being considered) of the moving average.

$$\hat{y}_t = \mu + w_t + \theta_1 w_{t-1} + \theta_2 w_{t-2} + ... + \theta_q w_{t-q} \tag{2.6}$$

Traditionally, while doing regression analysis, it is desirable to eliminate any kind of auto-correlation in the data to avoid biases. However, when handling cyclical data such as weather phenomena, these auto-correlations are intrinsic of the data and analyzing the error of the last forecast can give valuable information to make better ones. The models that we will be discussing further were designed for time series with no trend, constant variability, and stable correlations over time (Olson & Wu, 2020).

An evolution for these models is the Autorregressive Moving Average (ARMA) which is in essence a linear combination of the two techniques described before, and in their calculations, they consider auto-correlation, trend, and moving average terms. The models are often referred to as the ARMA(p,q) model and are described as in Equation 2.7.

$$y_t = w_t + \sum_{i=1}^{p} \phi_i y_{y-1} + \sum_{i=1}^{q} \theta_i w_{t-1} \tag{2.7}$$

From the equation alone we can observe its constituent parts, but to deepen on the most relevant ones:

1. The number of auto-correlation terms **P**, which is essentially what makes the model take advantage of the existing correlations. This is the autoregressive part of the model.

2. The number of moving average terms **Q** which serves to stabilize the data.

Finally, to cover for the inefficiencies of ARMA models in modellign non-stationary series, the Autoregressive Integrated Moving Average(ARIMA) were developed. Integrated is a measure of the amount of seasonal differences that are needed to achieve stationarity. This process is known as differencing and consist on computing the differences between consecutive observations. This process can help stabilize the mean of the time series by reducing trends and seasonality. The models are often referred to as the ARIMA(p,d.q) models, and follow the ARMA model, with **d** being the number of differencing order needed to reach stationality.

This is an approach that has a great overlap with Neural Networks, so we won't be considering it directly. However, we will be taking some ideas and implementations from previous work on this and applying its principle on our implementations.

2.3.3 Principal Component Analysis

The main tool from Unsupervised Learning we will be employing is Principal Component Analysis (PCA). PCA is the process used to compute the Principal Components (PC). These PC can be described as a low dimensional representation of the data, which simultaneously contains as much as the original data's variation. The first Principal Component is correspondent to the normalized linear combination of features with the largest variance, with the following components being the remaining combinations.

As stated by James et al. (2019): "The first principal component of a set of features $X_1, X_2, \cdots, X_p$ is the normalized linear combination of the features that has the largest variance. By normalized, we mean:

$$\sum_{j=1}^{p} \phi_{j1}^2 = 1. \tag{2.8}$$

We refer to the elements $\phi_{11}, \ldots, \phi_1$ as the loadings of the first principal component; together, the loadings make up the Principal Component loading vector, $\phi_1 = (\phi_{11}\ \phi_{21} \cdots \phi_{p1})^T$. We constrain the loadings so that their sum of squares is equal to one, since otherwise setting these elements to be arbitrarily large in absolute value could result in an arbitrarily large variance".

With the previous in mind, the first PC loading vector is the result of solving the optimization problem:

$$\underset{\phi_{11}\cdots\phi_{p1}}{maximize} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left\{ \sum_{j=1}^{p} \phi_{j1} x_{ij} \right\}^2 \right\} \tag{2.9}$$

for the values of $z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \cdots + \phi_{p1}x_{ip}$. The previous is true subject to the normalization described in Equation 2.8, for an $n * p$ data-set. Additionally, the values for $z_{11}, \cdots, z_{n1}$ are refered to as the scores of the First Principal Component.

Basically, PCA rearranges our data in a way that each Principal Component is a combination of our original data, resulting in having as many Principal Components as we had variables. The advantage of using PCA is that by analyzing the scores (or as we will be referring to it, the "Explained Variance") of each component, we can reduce the number of variables we are working with. This reduction can be done since each of the Principal Components explains a certain amount of variance in the original data, so by selecting the Components that represent together the largest amount of variance, we have a trusty representation of our original N variables in M number of components, aiming for M < N. Reducing the dimensionality of our data not only makes it more interpretable (simpler to plot) but at the same time makes it less taxing to work with in terms of computational power. We can run analysis in M variables instead of N, and as long as the explained variance for M is high enough, the result of these analysis will be as valid for N.

2.4 Neural Networks

Artificial Neural Networks (ANN or just NN) are information processing systems developed in the 1940's (although thought of much before) in attempts to recreate the workings of the nervous systems and brains of humans and animals. Although the field of Artificial Intelligence (AI) has developed considerably since the inception of NNs, the initial idea for them is still valid, where simple units work together to achieve a task by their combined effort after being properly trained.

Paraphrasing the first chapters of Kruse et al. (2011): The most basic component of a NN is called a Neuron (keeping with the anatomical parallels). Neurons are the simple unit we referred to before, and their function is to receive data, evaluate it, and transform it according to its programming. At the heart of each Neuron, we find the Activation Function. This function is determined by the programmer to transform the data as per its intention, thus it is usually used to make non-linear transformations.

The simplest NN will consist of a single unit, making linear transformations. However, that's not an efficient use of the resource, since Neurons work better when combined together. Since, ideally, different Neurons will be making different transformations on the data in order to achieve the desired result, it is important to have an adequate connection between them.

Weights and Biases are properties of the connections between Neurons that make linear transformations on the data from Neuron to Neuron and thus determine the strength between the connections. Assigning this Weights and Biases is what constitutes the Training process, and is typically done through a process known as Back-Propagation. This is an iterative process that strives to find the optimal values for the Weights and Biases by changing their value as a function of the error between the value generated by the Network's current parameters and the True value.

If we refer back to the simplest NN, it will have one Neuron with an associated Weight and Bias. Information will come into it, it will be transformed multiplicative by the Weight, additively by the Bias and non-linearly by the Neuron. This series of operations will generate a result which will be compared to the expected result. Through Back-Propagation the Network adjust the values of the Weight and Biases in order to hopefully produce a more accurate result on the next iteration.

With the understanding of how this simple NN would work, we can expand on how more complex Networks do it. NNs are typically organized in groups of Neurons that will work in parallel in what is known as a layer. Typical NN structures have at least:

[6]Image taken from: https://medium.com/fintechexplained/neural-networks-bias-and-weights-10b53e6285da

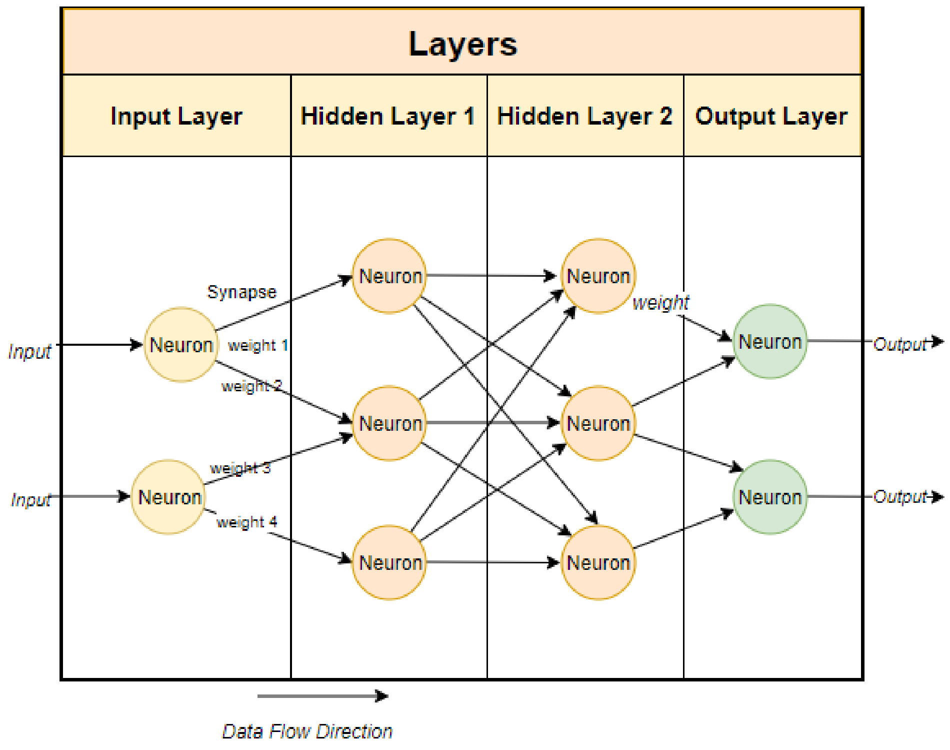


Figure 2.8: Depiction of a simple Neural Network with two input nodes, two hidden layers and two output nodes [6].

1. An Input Layer: responsible for the intake of data
2. One (or Several) Hidden Layer(s): responsible for data transformation
3. An Output Layer: responsible for providing the result from the NN

Usually, information flows through these layers in one direction: from Input layer, through the hidden layer(s) and to the Output layer. This is called a feed-forward Network. A typical visualisation of such a model can be observed in Fig. 2.8. Additionally, some NNs are designed in such a way that the information flows not only forward through the net, but back into itself; these kinds of Networks are known as Recurrent.

One of the biggest strengths of NNs are their ability to act as a universal approximator. This means that with the right configuration and with the correct training, a NN can be used to approximate any continuous function with great accuracy (Scarselli & Tsoi, 1998). This entails the ability to replicate most PDFs and use the NN to make predictions.

2.4.1 Long Short-Term Memory Networks

A particular type of Neural Networks developed by Sepp Hochreiter, first published in their 1997 paper "Long short-term memory" (Hochreiter &

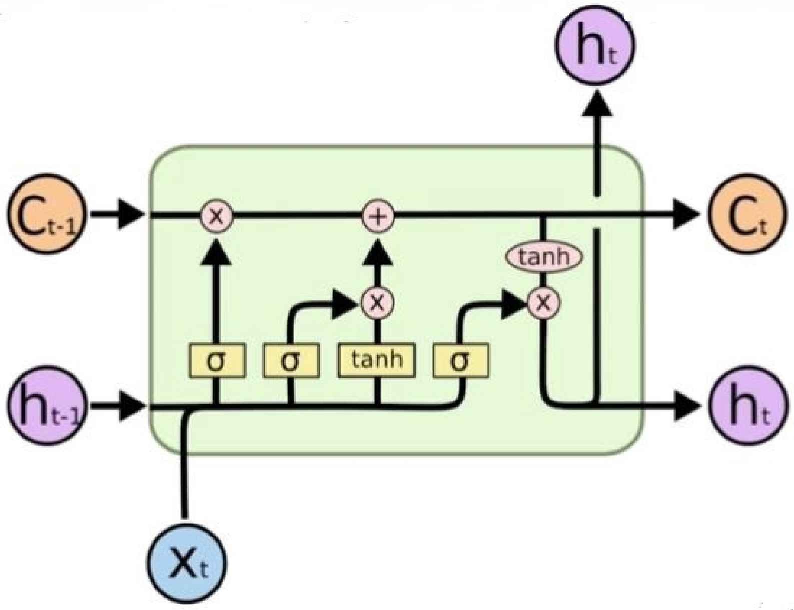


Figure 2.9: Visual representation of a LSTM node, which is the at the heart of a LSTM Network [7].

Schmidhuber, 1997). This kind of Network falls into the Recurrent Neural Network (RNN) category. One of the defining characteristics of RNNs, and maybe the reason for their widespread use and development, is their ability to encode temporal context in their feedback connections, which are capable of capturing the time-varying dynamics of the underlying system (Bianchi et al., 2017). In other words, the network has the ability to store and convey information to itself, essentially simulating memory.

The original aim of the development Long Short-Term Memory (LSTM) was to find a way to store information over extended time intervals (long-term dependencies) without the need for a long time investment in training through traditional Recurrent Back-Propagation. They manage to achieve their goal by introducing a new structure together with a gradient based method that attacked the insufficient decaying error back-flow causing the issue.

The LSTM architecture can be observed in Fig. 2.9 and works by taking an input $\boldsymbol{X}$ on an inner timestamp $\boldsymbol{t}$, and generating two outputs: it's Cell State $\boldsymbol{C}$ and it's Hidden State $\boldsymbol{h}$. The output is generated by taking $\boldsymbol{X}$ and h_{t-1} through a Forget Gate layer formed by a Sigmoid function. Through training, this gate learns which data "forget" and which to "remember". Afterward, the information is sifted and updated into the Cell State by the Input Gate layer. The Input Gate is composed of another Sigmoid layer and a tanh layer, these two individual results are multiplied together to continue the process. The old Cell State C_{t-1} is then updated by multiplying it by the result of the Forget Date and later adding the result

[7]Image taken from: https://www.iarai.ac.at/publication/mc-lstm-mass-conserving-lstm/

of the Input Gate, this gives us our Cell State C_t that will travel to the Output Gate. Additionally in the Output Gate layer a process similar to the Input Gate is executed, where X_t and h_{t-1} are passed through a Sigmoid function to be later multiplied with our current Cell State C_t. This operation produced our updated Hiden State h_t, which is the last output of our Output gate ('Understanding LSTM Networks', 2015).

The most common application for LSTM networks is in problems were order dependencies or sequencing are involved as stated by Smagulova and James (2019). With this in mind we can see why they are commonly used to address problems like ours.

2.4.2 Transformers

Transformers surfaced with Vaswani et al. (2017)'s paper "Attention is All You Need". With this publication the authors revolutionized the Natural Language Processing (NLP) field by introducing a better alternative to the traditional Recurrent/Convolutional Encoder-Decoder architecture that had dominated the field so far. The authors realized that the best performing models connect the Encoder and the Decoder through an Attention Mechanism and proposed this new architecture based only in the Attention Mechanism, ditching the recurrence and convolutions completely.

To better understand this model, one must understand how Attention works first. Attention was introduced in 2014 with the aim to create a way to improve the performance of the models used at the time for machine translation, which was constricted by fixed sentence length and limited access to information (Bahdanau et al., 2014). Usually these tasks involve an Encoder-Decoder model, where a RNN serves as an encoder taking the inputs and generating an abstract representation of the information learned from the input as a final hidden state. This hidden state is transferred to the decoder which uses it in an iterative process to generate each element of the output, using each of this predictions recurrently to generate the next one.

Attention manages it's objectives by allowing the decoder to have access to all the hidden states created by the encoder, and evaluating and assigning the relative importance of each input to rest of the inputs. Using a score function, the model can measure the similarity between two vectors, in this case the similarity between the decoder and each of the encoder's hidden states, and "concentrate" on the most relevant parts of the input sequence to learn from.

The Transformers architecture can be observed in Fig. 2.10. This architecture builds upon the traditional Encoder-Decoder model taking every aspect of it and improving it.

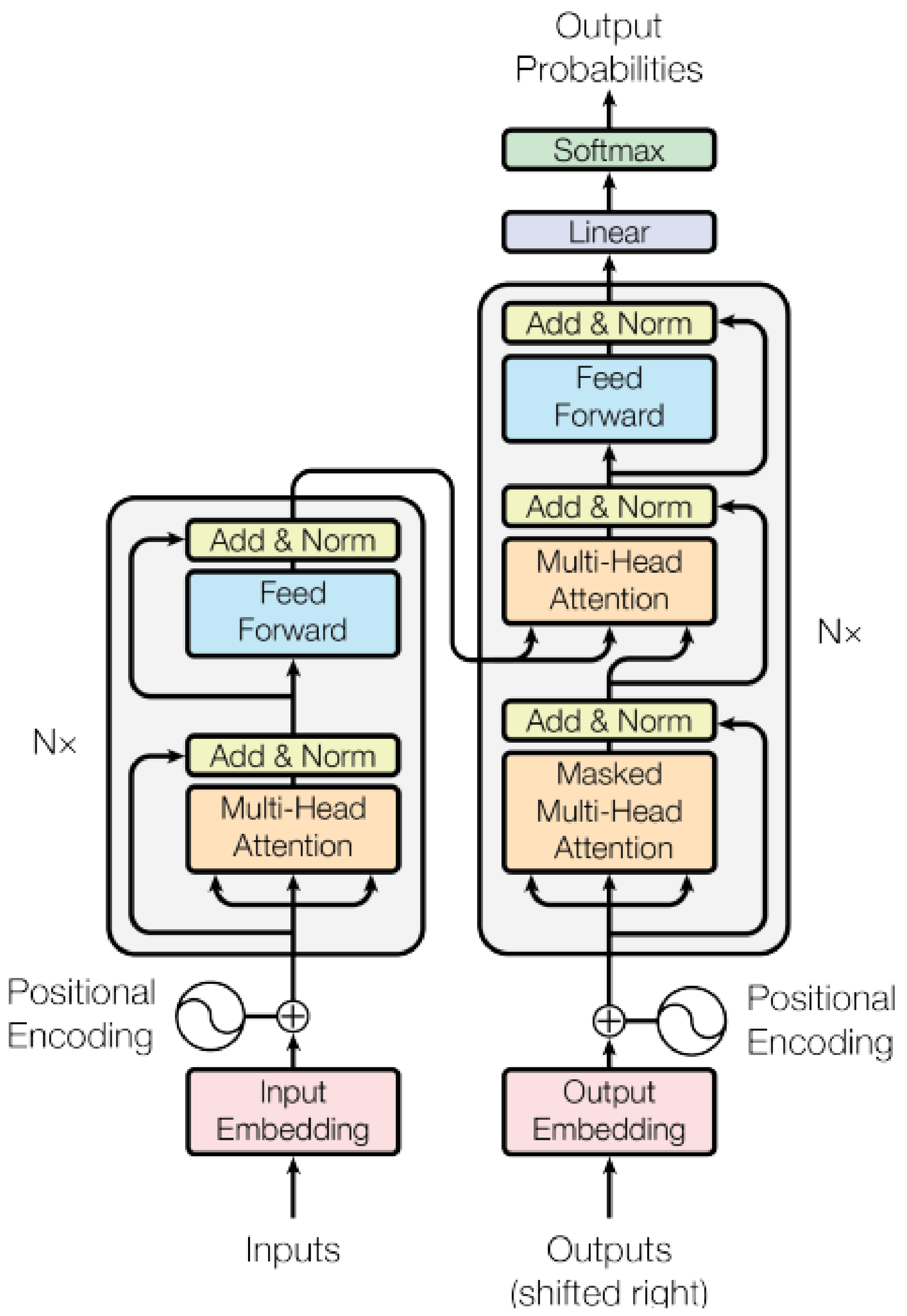


Figure 2.10: Transformers architecture as depicted in Attention is All You Need (Vaswani et al., 2017).

The model's greatest contributions can be summarized as:

- ***Positional Encoders***: Since this model doesn't depend on Recurrence, the information regarding order sequence is added "at the bottoms of the encoder and decoder stacks" (Vaswani et al., 2017) through a vector calculated as described in Equation 2.10, where *pos* is the position and *i* is the dimension. This results in a vector that can be added to the embeddings to aggregate the positional information.

$$\begin{aligned} PE_{pos^{2_i}} &= sin(pos/10000^{2_i/d_{model}}) \\ PE_{pos^{2_i+1}} &= cos(pos/10000^{2_i/d_{model}}) \end{aligned} \tag{2.10}$$

- ***Multi-headed attention***: This module allows the Transformer to apply self-attention in a way that modifies the traditional Attention method of associating each individual component in the input to the other components. This method consist of splitting the components of the traditional Atention module (key, querry and values) into different projections referred to as heads. Each head goes through the self attention process allowing the model learn something different from each, thus giving the model more representation power.

- ***Parallel Processing***: Hand in hand with the multi-headed approach, both the Encoder and Decoder can be stacked N_x times to further encode the information, where each layer has the opportunity to learn different attention representation. This model not only allows for better prediction power but it also facilitates the parallelization of the computing operations, allowing it the models to make better use of GPUs and reduce training times.

Transformers have been revolutionising the NLP field since their creation, with BERT (Devlin et al., 2018) and GPT ('GPT-3 Powers the Next Generation of Apps', 2022) being probably the most renowned ones. However, their ability to process sequence dependent data better than most of the alternatives is allowing this model to permeate to other Machine Learning fields like image and audio processing (Child et al., 2019), modeling biological properties in amino-acids and proteins (Rives et al., 2021), credit assignment in reinforced learning (Ferret et al., 2019), and modeling physical systems (Geneva & Zabaras, 2022), to name a few.

2.5 State of the art in Wind Speed modeling

While reviewing the literature around Wind Speed analysis, an interesting evolution in the methods and techniques to do so can be observed. This evolutionary path starts with the early mathematical and statistical models, then it grows into two divergent approaches, which we will call the Machine Learning approach and the Statistical Learning approach. These two approaches produce interesting results for some time. However, the most recent trends seem to indicate that these two approaches are converging into one main path again, with the latest tendencies is having hybrid approaches that include complementary techniques from both paths.

In order to understand this evolution better, we can start with R. I. Harris and Cook (2014). In this paper he starts challenging the most used methods at the time. They express that Weibull is probably the most used PDF used while analyzing wind, however it doesn't have any theoretical background in either atmospheric nor probability theory. Harris proposes an Offset Elliptic Normal (OEN) Distribution for modeling wind's mean speed, which satisfies both, being founded within probability theory and resembling a Forward Weibull Distribution. OEN's efficiency was tested using vector analysis of wind observations at two widely separated UK stations with successful results. One caveat of this model is that it takes he probability distribution of mean wind speeds taken over an averaging period between 10 minutes and 1 hour, however, this satisfies the World Meteorological Organization's (WMO) standards. This model works under the framework of wind's components having disjointed characteristics, making it a problem of "sum of ellipses" instead of a "Sum of exponentials", which itself presents the problem of determining the right amount of mechanisms without any standard procedure. At the moment of writing, it was unclear if a mixed OEN model could be used to predict extremes, which would be the source of future work.

By no means we mean to imply that the previous paper is the one that gave birth to either form of Wind Speed analysis. One could trace back this kind of work even further back with studies like the ones done by Cadenas et al. (2010), whose approach is one of the few we found reference of handling a data-set with its resolution at $1Hz$. The authors are demonstrating the early efficacy of an exponential smoothing method, particularly effective when predicting at short intervals. Another early study we could refer to is the work by Li and Shi (2010), where one of their main focused is to determine the appropriate amount of lags to use in order to obtain the best results. We are stating that we consider it one that deepens the understanding of the field in a way that we could find diverging paths from it.

One of the earliest works in the field of Statistical Learning is the work of Skittides and Früh (2014). Here the authors state that Principal Component Analysis is capable of identifying patterns in wind behavior by being able to create different representations of either just wind speed or multivariate measurements. And that by combining PCA predictions with persistence prediction at very short time scales is possible to eliminate most of the weaknesses of the previous similar methods.

We can link the previous study with the work from Geng et al. (2020), where they combine the use of PCA with an LSTM model, demonstrating the more recent tendency for mixed models. The hybrid approach consists of using PCA to process the original meteorological data. Afterwards, an LSTM network is used to obtain the predictions. Even though the author states that their future work will consist of implementing different deep learning models for this hybrid method, the results of this study demonstrate that the current model requires less training data than other models, while simultaneously improving the accuracy of forecasting predictions.

Going back to the trend of challenging the accepted norm, while still keeping with the more traditional Statistical Learning approach we can refer to J. Yu et al. (2019). In their paper the focus on analysing Wind Speed at a sampling rate of 1 second using Weibull, Nakagami, Rician and Rayleigh distributions. We might remark this is an important source for our project, since like us, Yu's publication is one of the few using data with a 1 second sampling rate. They concluded that in most cases the R^2 and the Root Mean Square Deviation (RSME) values for the Nakagami distribution presented the best results, and Rayleigh had the poorest ones. However, this distribution has only one parameter, which allows for a quicker calculation. Overall, every distribution was suitable to model wind speed distributions, thus enabling a proper assessment and classification. This study wasn't able to accurately predict periods with high percentages of null wind which can be expanded on following studies, but overall provides a good basis for our project.

Continuing and highlighting one of the most interesting papers in the Statistical Learning branch we can mention Salim et al. (2019)'s work. The main contribution of this paper is the ability to predict the wind PDF (Weibull) for one location based on the knowledge of wind speed at other locations. This is done through the proposed bi-variate wind Weibull. The method was developed after showing that there are existing structural similarities between the wind Weibull versus normal distribution models arising from the fact Weibull reflects a natural phenomenon with some restrictions, and adapting Gaussian known multivariate functions. The results overall were positive, getting the best results when the correlation factor between the two data-sets is high. This usually happened when the two locations are not very far from each other (< 20 km). A new proposed moving sample window is suggested to overcome large distances forecasting errors. This results in lower Mean Average Percentual Error in the prediction for wind speed, compared to the traditional models used. As per the correlation, if the correlation factor between data-sets is decreased, the prediction error will be increased accordingly. The authors suggest that their findings are significant enough to include into a real-life existing system or used to evaluate possible sites for wind farms. The development of a module for MatLab is proposed as a possible next step to continuing this work.

A link between the traditional 10 minute interval analysis can be found in Milan et al. (2014)'s work highlighting the use of the Langevin process to model wind data at a $1Hz$ resolution. This model works by extracting two functions: the drift function, which describes the data's deterministic behavior through time, and the diffusion function which describes the stochastic fluctuations in the system. The authors conclude that the drift coefficient is able to best estimate the behavior of the data. Additionally, the authors develop a method to estimate $1Hz$ wind dynamics from data collected at 10-minute intervals.

When talking about approaches leaning more towards Machine Learning , the literature is vast due to their increasing popularity. However, we would like to highlight the work by N. Wu et al. (2020). They introduce the VMD-ADE-TFT model, that is able to take account of not only historical wind speed but other meteorological data as well in order to make predictions. The results point at the model being able to outperform nearly every other comparable model in different indicators, in addition to contributing to making the results more interpretable than these other alternatives. Stating that the forecasts are satisfactory both in stability and accuracy. Finally, the authors express their desire of testing the model over different time scales to prove its effectiveness.

A trend we think is noteworthy in this kind of approach is the focus on the spatial and temporal features, which we did not observe at the same degree on the mathematical approaches. Here we can mention Khodayar and Wang (2018). They produced a scalable graph convolutional deep learning architecture (GCDLA) that manages to extract both temporal and spatial features from a wind speed time-series from whole graph nodes. This approach combines Deep Learning and Graph Theory to achieve results that rival other DL architectures which overlook spatial-temporal features, like Deep Belief Networks and Stacked Auto-encoder networks. After the temporal features have been extracted by a LSTM model and modeled as a graph, they are fed to a convolutional deep learning architecture. This results in an architecture that excels at handling the inherent noise and uncertainties present in wind speed data.

More recently on the same lane we have the work by M. Yu et al. (2020). They propose a Superposition Graph Neural Network (SGNN) to tackle the challenge of extracting the aforementioned spatial and temporal features in an effort to utilize data from an entire wind farm, with emphasis on the extraction of spatial features from each turbine to make accurate predictions. Although successful at reducing the prediction error when compared to other methods, the authors acknowledge that in many practical applications one does not need to predict for the entire farm. However, we interpret the existence of the SGNN model as evidence that incorporating spatial features into wind speed analysis is a trend deemed worthy researchers working on the subject and very likely to continue.

Starting to see the attempts of bridging and reconcile the two methodologies, and perhaps the biggest reason this project is taking its current route, we can refer to P. G. Lind et al. (2017)'s work. In order to evaluate the suitability of a stochastic approach to model normal wind behavior, the authors compare this method with a NN. The NN utilized was a recurrent Non-linear Auto-Regressive with exogenous inputs (NARX) consisting of a three node input layer, a hidden layer with a sigmoid activation function and a linear output layer. The Langevin model was used for the stochastic approach. Many metrics were used to compare the results provided by the NN and the Langevin model, which encompassed analyzing not only the error of prediction but the different statistical moments (mean, standard deviation, skewness and kurtosis) as well. For those metrics, the authors conclude that the stochastic approach has comparable or even better results than the NN, especially excelling in modeling non-Gaussian fluctuations. Furthermore, the Langevin model provides a better reproduction of the data's original variance and demonstrated better performance with data sampled at a higher frequency.

One of the most remarkable papers, and one at the beginning of the new hybrid approach tendency is Islam et al. (2017)'s work. This article proposes using two hybrid ANN systems, using genetic algorithms (GA-NN) and particle swarm optimization (PSO-NN) for the extrapolation of wind speed, opposed to the traditional methods based on logarithmic law and power law. The model was utilized to extrapolate wind speed at higher heights based on the measured speed at lower heights, obtaining results that outperformed traditional methods on a daily, monthly, and yearly scale. Arguing that it is an arduous, and most of the time inconvenient, task to find the right Back Propagation (BP) technique for wind speed profile estimation due to its high fluctuating behavior, the authors propose this technique. Using genetic algorithms to determine the weights and biases for the ANN can be a valuable alternative, resulting on better parameters and less time invested, with the use of Levenberg–Marquardt (LM) algorithm to minimize the Mean Squared Error between the ANN results and the actual wind speed. The data used was ten minute averaged wind speed data at 10, 20, 30 and 40 meters from Juaymah meteorological station in Saudi Arabia for a year's period, with a 80-20 train-test split. The data was filtered as to only use the records where lower height wind speed must be lower than the upper height speed. Predicted data was feed back into the ANN and used to extrapolate to even higher altitudes. For the ANN: the number of hidden neurons was twice the number of inputs, using just one hidden layer. Ten populations were used in the Genetic Algorithm to represent the weights and biases (with ranges between -30 and +30), with 150 iterations and a 0.80 crossover factor.

Another remarkable product of the hybrid approach era is Wang et al. (2018)'s paper. With wind speeds measured at 10 meters with a sampling interval of 10 minutes from three locations, this paper proposes a new hybrid model for wind speed forecasting. The model combines Extreme learning Machine (ELM) with improved complementary ensemble empirical mode decomposition with adaptive noise (ICEEMDAN) and ARIMA. The autoregressive model is used to determine the best input variables, while the ICEEMDAN is used to improve the prediction accuracy by post-processing the errors, and this is fed to the ELM model to obtain the wind predictions. Through the process of arriving to this hybrid model, the authors tested each component by stages (i.e. ARIMA as a single model, later adding ICEEDMAN as pre-processing, and finally adding the post-processing) and with different configurations, concluding that: (a) it is feasible using a variation of the ARIMA for pre-processing, (b) the results of hybrid models surpass those of the single models in term of prediction accuracy and (c) and post-processing models have higher accuracy than base ELM models as evidence by the reduction in the mean absolute percentual error.

Finally, we feel the need to state that even if there is a great influx of good results coming from hybrid methods, this is not to say that those are the only alternatives. Studies like the ones elaborated by Liu et al. (2021) and Z. Zhang et al. (2019), demonstrate conclusively that under certain circumstances the Statistical Approach is still king. Clearly stating that models like ARIMA and their variants are still outperforming other methods, even Machine Learning and Hybrid, when it comes to short-length wind speed forecasting. That is not to say that these are the ultimate solution to every Wind Speed forecasting problem, but that there is no one-fits all solution, and the field is still constantly evolving and improving.

Chapter 3

Data and Methods

3.1 Data set from FINO1

The data analyzed in this thesis covers a time span of 15 months from October 2015 to December 2016. As mentioned in Section 1, it was collected at an offshore facility on the coast of Germany. The recordings were done at a 1 second interval, and collected at 8 different altitudes, starting at 30 m above sea level and having one every 10 meters until reaching 100 meters. Unfortunately, we do not possess a measurement of the speed at sea level (or closer to it than 30 meters), so we might not observe the full effect of drag along the wind profile.

To better understand the data, we decided to start by plotting it in several different manners and computing it's Statistical Moments. We felt this process gave us valuable insight, which aided in the selection of the methodology. Some of these figures give information that helps nuance the results from the Neural Networks, as such they will be included in Section 4.2. To have a full panoramic picture of the information we started plotting the speed through time. A sample of this time series can be observed in Fig. 3.1, where we are just depicting some of the heights. In these time series we could observe the stochastic behavior of the wind speed on our data-set, which doesn't show any discernible patterns or seasonality. Furthermore, we could observe that there are certain periods where there is missing data from one or more of the measuring units. This missing information was our first indication as to the importance of feeding the Network not only the wind speeds but the time-stamp on which they were recorded as well.

Once we understood the general behavior of the data through time, we realized the importance of how the speed changes through the different heights. A visualization of the PDF for the changes in speed can be observed in Fig. 3.2, where we observe that as the distance between the measurement increases, the curves tend to be wider, evidencing an increase on their standard deviation. Furthermore, higher altitudes have usually higher means, which corresponds to the theoretical wind profile. From here we understood how speed and height can be correlated and it further reinforced our selected methodology choices.

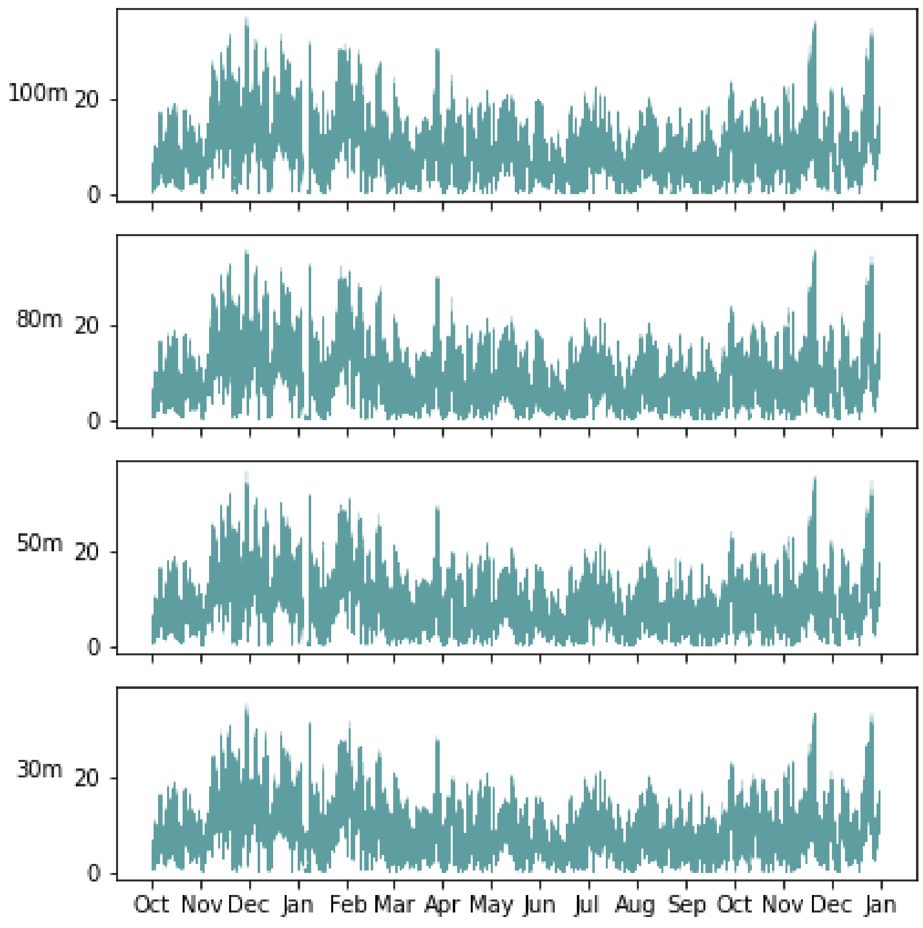


Figure 3.1: Time series for wind speed at different heights. A similar but not identical behavior can be observed throughout the different heights. Additionally, some periods of missing information are identifiable.

Theoretical wind profiles, as stated by LeHau (1959), behave in a way that lower speeds near the ground and increase in a logarithmic way as the altitude increases. As visualized in Fig. 3.2, this would be the case for our data-set. However, we created an animated visualization that allowed us to observe the behavior of the wind profiles in different time periods. A sample of this visualization can be observed in Fig. 3.3 it can be observed that individual wind profiles do not necessarily follow theoretical behavior, but have in fact a less defined profile. As we will be analyzing and predicting individual wind profiles, understanding this behavior was a step that would also shape the selected methods and how we approached the problem.

As per the general status of the data, most of the recordings are in place and uniformly labeled. The speed is recorded throughout the entire data-set with five decimals of precision. However, while exploring the data we observed various unusual behavior.

In addition to the missing data (observable in Fig. 3.1 between January

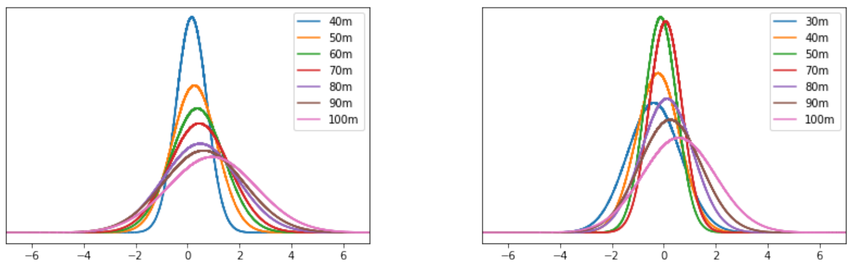


Figure 3.2: Depiction of the PDF for change of speed as a function of a height differential. (Right) Base at 30m (Left) base at 60m. Greater changes of speed and more variety in these changes can be observed as the height difference increases.

and February as an example of a large amount missing), spread through the data-set we found some instances where the speed recording was exactly zero, and some other where the difference of speed between two time stamps is exactly zero. As instructed by the data provider, we attributed both occurrences to some fault or malfunction in the equipment, or the connection to it. We replaced these records with a NaN (not a number) so as to not interfere with our results, as per Scheffer (2002). We opted for this alternative over trying to infer the missing data since it is not a common occurrence (less than 1% of the data) that occurs at random. Additionally, the time-stamps of the data are being fed into the model as a variable, so the gap created for the missing data is being taken into consideration during analysis and inference.

While analyzing the Statistical Moments for the data-set and performing early predictions, we got some results that were dissonant with the theory. These particular results led us to realize that (and as it might be observed in Fig. 1.1) due to the tower's architecture, the anemometer placed at the 100 meter height is exposed to wind on a different manner than the rest of them. This difference comes from there is less equipment and infrastructure higher up, leading to this particular anemometer not being as shielded from certain angles as the rest, thus presenting different statistical behaviour. We opted to not consider these readings for our models and predictions in order to avoid introducing unnecessary noise to the models.

The vast majority of these data analysis, modifications, and transformations were performed using Python as a programming language. The tools used were mainly Numpy (C. R. Harris et al., 2020), and Pandas (McKinney et al., 2010).

3.2 Methodology

While facing the challenge of approaching a well-researched subject under different premises or with different goals in mind, one can get the feeling similar to trying to find the best hiking trails in your area while looking at a map depicting the bus routes. One is bound to find information that is relatable and might spark some good ideas, but it will not be completely

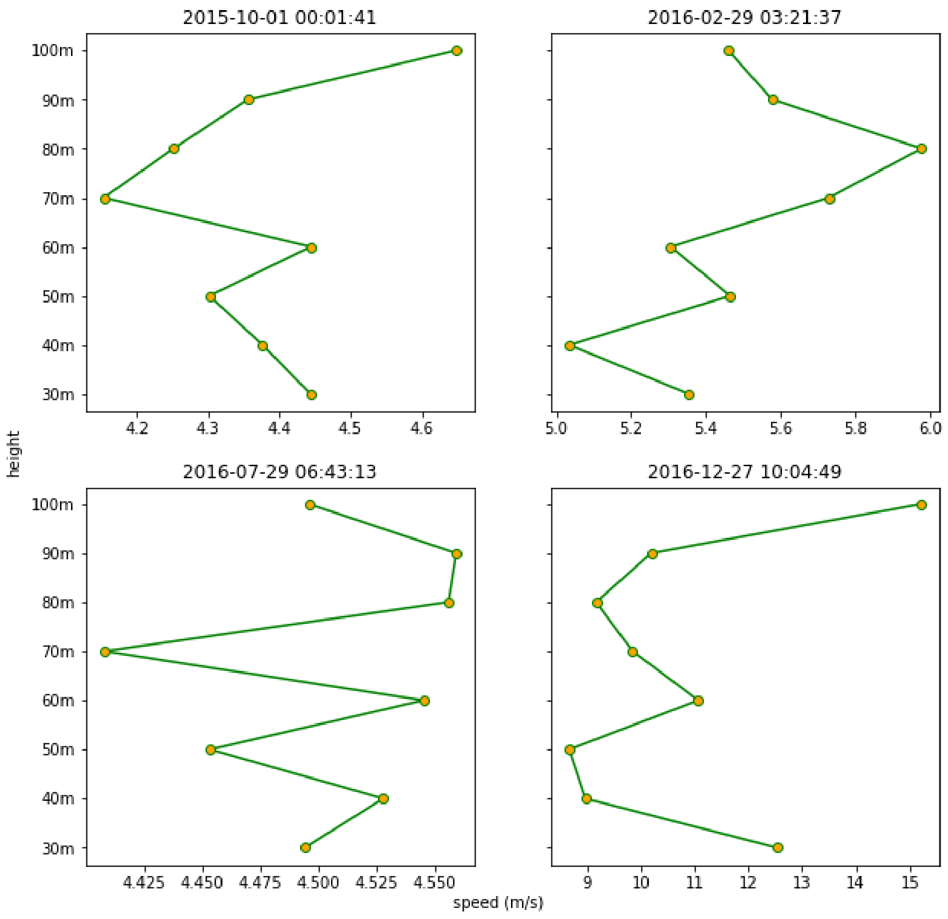


Figure 3.3: Depiction of Wind Profiles at FINO1 at four different timestamps. It can be observed that when analyzed at this time resolution, wind profiles tend to show great variability and non-uniformity.

useful or relatable. Remembering that our goal is to be able to produce a model capable of predicting wind profiles at a one-second sampling rate; while reviewing the most relevant publications related to our project, we experienced the aforementioned feeling. There is an abundant source of studies relating to the subject of forecasting wind behavior. However, the vast majority of these efforts are trying to find a solution to forecasting wind speed in ten-minute intervals, which is the current industry standard for meteorology and energy production. By steering clear of using averages and trying to create accurate forecasts with wind speed measurements taken every second, we are introducing more variability to the problem, which in turn can affect the reliability and effectiveness of these well-researched methods.

As discussed in Section 2.5, the current research trends in modeling Wind Profiles seem to point towards combining Machine Learning and Mathematical methods to obtain the best results. While researching for the aforementioned section we couldn't, bar some exceptions, find many published instances of the combination of methods at this level of detail (e.g. the work by Cadenas et al. (2010), and Milan et al. (2014)). Furthermore, we found more instances of using linear or mathematical solutions as the time resolution got closer to ours. As we looked deeper into the general problem and the currently proposed solutions, we developed a secondary hypothesis: machine learning methods seem to excel at solving the industry standard problem, while mathematical models seem to be better suited to handle more variability, as the one we will be facing due to the increased resolution of our data.

This secondary hypothesis is what gave birth to our way of approaching the problem. We decided that our plan of action would be to create intermediate data-sets simulating different time resolutions from our data. Testing different approaches with the same starting conditions (except the data's resolution), would allow us to make a comparative analysis of the results among the different time-windows used. Using this approach would let us replicate some of the industry's practices and test their effectiveness at predicting on our data at the resolution they were initially designed for. Additionally, this approach gives us comparable results for the different methods, allowing us to evaluate their performance across different time resolutions.

We would start by simulating from our data "industry-standard" database by averaging our second-to-second data into ten-minute intervals. The other data-sets were simulated with the following resolutions: 5 minutes, 2 minutes, 1 minute, 30 seconds, 10 seconds and 5 seconds. We settled for those time-frames since we felt we would be able to observe and evidence any trends on the performance while increasing the level of detail and variability in the data. We are going to be referring to the different time intervals used to create these data-sets as T. They were computed through the whole data-set as:

$$\langle v_h(t) \rangle_T = \frac{1}{T} \sum_{k=t}^{t+T-1} v_h(k) \tag{3.1}$$

where v_t is the velocity at the given timestamp, h is the height of the measurement and T is the size of the interval we used to average (from 5 seconds to 600 seconds).

Some examples of how the wind profiles vary depending on the values for T can be observed in Fig. 3.4. From here it can be observed that the wind profiles corresponding to the data-sets with higher values fot T (i.e. 2, 5 and 10 minutes) tend to have a smoother profile, while the contrary can be said about the data-sets with higher resolutions. This reinforces our belief to evaluate the chosen methods at the different time resolutions, as the variability of the data increases.

A more generalized image can be observed in Fig. 3.5. Here we depict the PDF for the data at each height for each value for T, to observe how these two variables affect the data. We plotted as well their respective first and second Statistical Moments (Mean and Standard Deviation), to evidence how they vary through the different T and heights. This expanded view allows us to understand to a better degree how the

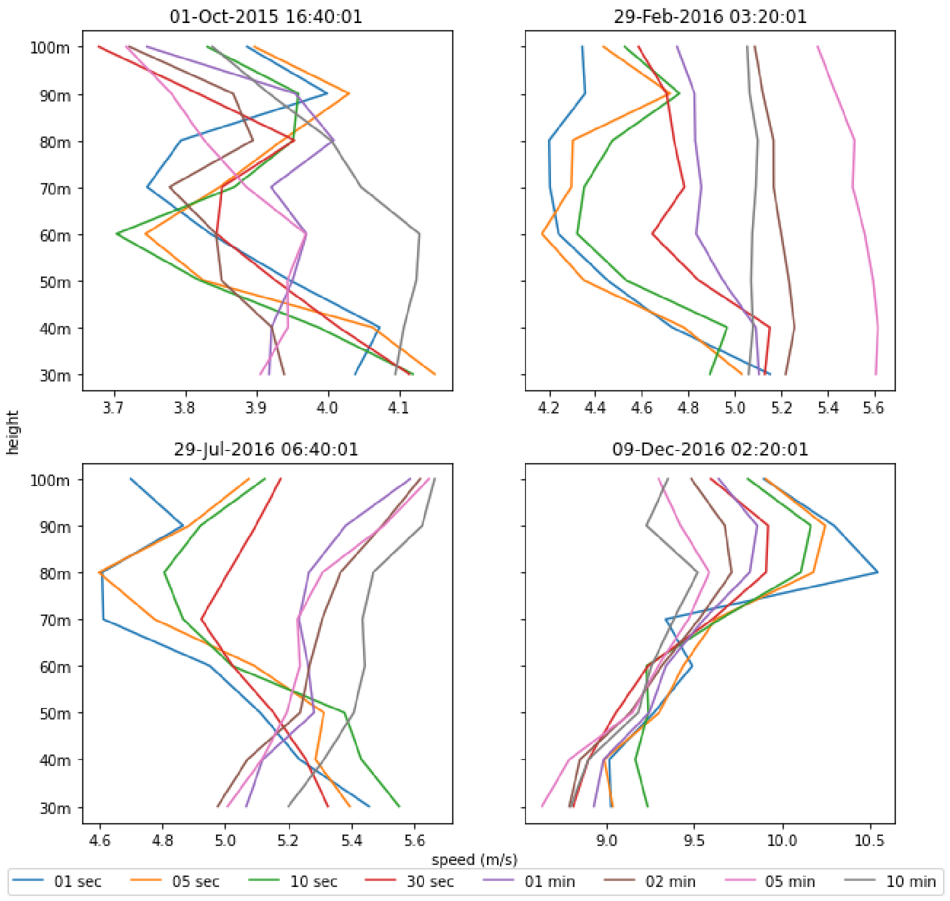


Figure 3.4: Example of how Wind Profiles change when averaged at different time intervals. The Profiles corresponding to data averaged at larger intervals show a smoother profile.

different speeds composing the wind profile are distributed along our data-sets. In general, we observe longer tails as the height increases and the tau decreases, and higher taller curves at lower altitudes, as well as variations in the Statistical Moments throughout the board. All this reinforces the need to create these intermediate data-sets, as these changes are progressive as the data resolution changes.

Focusing back on predicting the wind profiles; branching off from the work of Sim et al. (2018) we selected a secondary approach to attack our problem and produce comparable results. In addition to predicting the absolute speeds that will conform to the wind profile in future time signatures, we can compute and predict the speed differential between the current measurement and a future measurement. This second approach gives a different perspective to the problem, since it changes the data-set from having a Weibull distribution to one that resembles closer to a Gaussian.

This secondary approach entailed the creation of what we will be referring to as *tau* data-sets, where τ corresponds to the distance between the two measurements in seconds. To have comparable results we chose

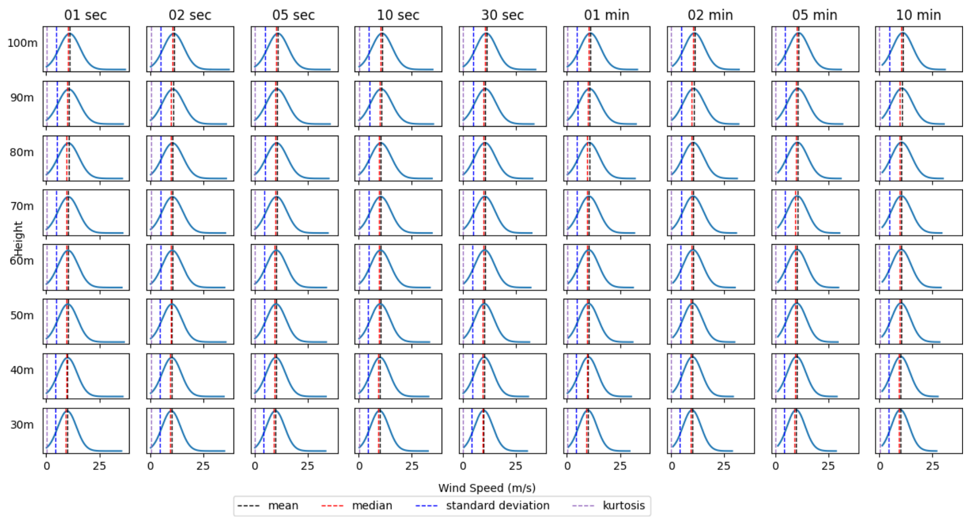


Figure 3.5: Distribution of Wind Speed for each height for every value of tau, where slight differences in curve shape, tail lengths and Statistical Moments can be observed.

to have values of τ equal to the T values (5, 10, 30, 60, 120, 300 and 600 seconds) used in our initial approach. As such, these data-sets where created with τ defined as:

$$\Delta_\tau v_h(t) = v_h(t+\tau) - v_h(t)\,. \tag{3.2}$$

With this approach, regardless of the value for τ the resulting data-set has a shape that resembles more a Gaussian than a Weibull distribution discussed in Subsection 2.1.2. A visualization can be observed in Fig. 3.6. There we can observe the changes in the probability of extreme events for both series changes depending on the value of τ and the values for T, having a lower probability for these for larger values of τ. Additionally, it can be observed for both cases that the deviation of the data tends to be larger as the time-window increases. All in all, these visualizations reinforce our believe that the methodology of creating supplementary data-sets can increase our understanding of how the methods' performance can be affected by the resolution and distribution of the data they are being trained upon.

To close this section, we would like to reiterate explicitly on why are we choosing this double approach with our methods. Although, assuming good performance for our Neural Network models, both approaches will generate working and useful predictions for our many data-sets. We expect that some of our methods will have more success than others depending on the data-set's distribution and the time resolution. By choosing to approach the problem by two different angles, we are doubling the amount of analysis to complete but we are also increasing our probability of getting results that will deepen our knowledge on the subject. We are deeming this outcome worthy of the extra resources used on the additional analysis.

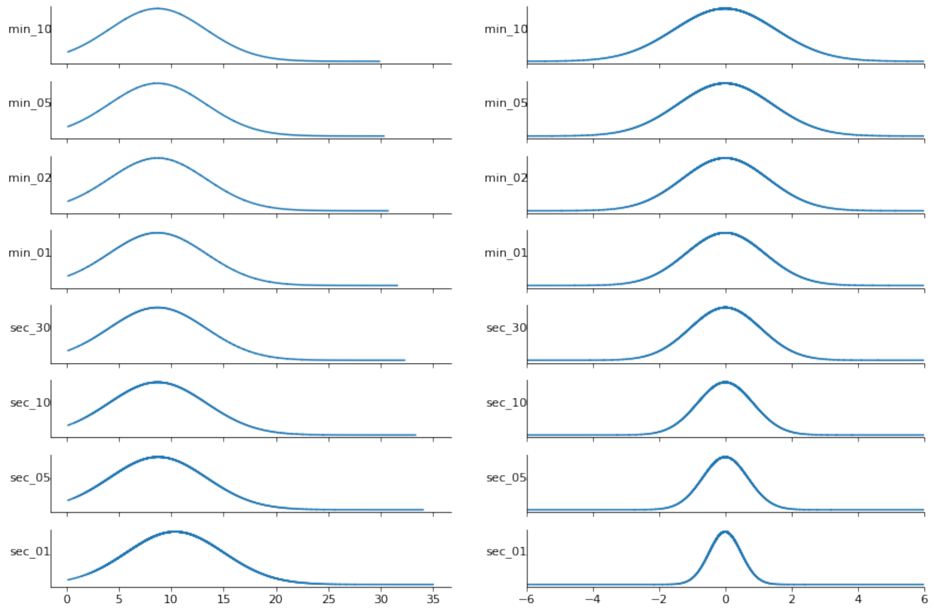


Figure 3.6: Visualization of the Probability Density Function for the different simulated data-sets.(Left: for T data-sets)(Right: for τ data-sets).

3.3 Methods, Models and Their Evaluation

For this project we will be employing the following methods in the following order. We have chosen these methods since they represent a good compromise between what is working best within the latest research and what we can achieve with the resources we currently have available. Furthermore, some of these methods are closely related to each other, allowing us to have a natural progression in the robustness of the tools we use that matches the progression of the complexity of the analysis we will be making. Finally, one of our methods (depending on it's results) could provide us with even more information to support the remaining ones.

We will be building our networks using the framework privided by Pytorch (Paszke et al., 2019). We selected this framework due to its combination of ease of use, accessibility, familiarity and available tools. Additionally, some of the publications we will be consulting have an available repository which we can consult and the ones we are most interested in are utilizing it as well.

3.3.1 Principal Component Analysis (PCA)

Dealing with multivariate analysis can be not only complicated, but also resource craving. By finding ways to describe our data accurately with less variables will help us with faster processing times and increased comprehension of the interrelation within our variables. As stated in Subsection 2.3.3, PCA is the ideal way of achieving this dimension reduction. Our aim is to achieve through the use of PCA a way to represent our data-sets with a reduced number of variables, by selecting the principal components that have the most explained variance. For this task, we will be utilizing the framework and routines available in the Scipy (Virtanen et al., 2020) package in Python, due to our familiarity with it and its proven effectiveness (as is the case for Ilie et al. (2017)'s work).

We will be taking ideas from Skittides and Früh (2014) and Geng et al. (2020) who had promising results utilizing PCA to reduce the amount of other variables related to wind-forecasting in order to improve their predictions.

We are hoping that this reduction in the number of analyzed variables might increase the success of the models in their prediction. Additionally, we are interested in feed our models with the entire data-set resulting from the PCA analysis, not just the components with the most explained variance. This could provide interesting results given that each of these new variables will not be correlated with each other, as opposed to our original data set.

3.3.2 Long Short Term Memory Neural Networks (LSTM NNs)

We are selecting this kind of NN as a start point because they are powerful enough to model our data, but simple enough to allow us to concentrate on the analytical aspect of the testing, rather than the coding side. Additionally, as stated in Chapter 2 they were created and popularized because of their ability to capture time dependencies within the data. The plan is to start with a simple model on which we can test the process' pipeline and then work on making it more robust later on. This will give us a baseline of how this kind of NN behaves while processing our different data-sets, allowing us to test our hypothesis on their behavior at lower time resolutions. We will be utilizing the work by Geng et al. (2020) as a reference for our architecture. However, the bulk of this project will come through experimentation to find the hyper-parameters that work best for our data-set. Additional manipulations to the data-sets will need to be performed in order to use them as valid inputs to the model, such as creating DataLoaders to manage batch sizes and ensuring the set is structured in an input-output fashion.

3.3.3 Transformers

After developing a satisfactory LSTM model, we are using that knowledge to develop the more robust model which we consider is a Transformer model. Their ability to capture dependencies way beyond what is possible with RNNs should provide some more insights from our data. Being an evolved form of the LSTM models, the transition towards transformers should present less issues from a coding point of view. This potentially smooth transition will allow us to concentrate in deepening our understanding of this model's strengths and weaknesses, to be able to leverage them adequately when it comes to building the final Transformers model. We will be referencing the work by N. Wu et al. (2020) and B. Wu et al. (2022) for our implementation.

3.3.4 Evaluation

In order to identify the methods and the parameters that give the most satisfactory results, we need a measurement that allow us to compare how the models performed on their own and against each other. Additionally, this measurement should be generalized enough that lets us understand the performance of the predictions, even if the different models' input and output have different sizes. As one of our first steps is to create data-sets that represent our data in different time intervals, we need to take into account that the amount of predictions will vary for each data-set. Additionally, we will be predicting wind profiles, which entails that the output of the models will be a group of seven predictions instead of a single number. In addition to measuring Training and Test loss, we measured the relative error of each prediction though the entire Wind Profile. This error is defined as:

$$\Delta = \frac{1}{h * \delta} \sum_{i=1}^{\delta} \sum_{j=1}^{h} \left| \frac{y^j_{predicted^i} - y^j_{true^i}}{y^j_{true^i}} \right| \tag{3.3}$$

with δ being the size of the data-set used and h the number of heights that conform the wind profile. The value of h will be typically seven, one for each height excluding 100m. However, due to the results of the PCA analysis and their possibility of reducing the number of variables, this number might decrease. For the majority of studies relating to wind speed forecasting, the accuracy is calculated utilizing the Mean Absolute Error (MAE), calculated as:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}. \tag{3.4}$$

Equation 3.3 is an adaptation of Equation 3.4 that suits the shape of or data-set.

At the end, we should be able to not only elaborate a comparative study between these methods, but have a good idea of which combination of PCA and NN could give the best result if combined to create a mixed model. As discussed in Section 2.5, combining Statistical Learning methods with Neural Networks is a rising trend in this field with a track of increasingly good results.

Chapter 4

Assessing the Evolution of Wind Profiles

Statistical Analysis and AI-Based Approaches

4.1 Principal Component Analysis

The first step of our analysis was trying to find a way to reduce the dimensionality of our problem. As such, after creating the supplementary data-sets T and τ, we ran a Principal Component Analysis over all sixteen of them. For this task, we utilized the framework and routines available in the Scipy (Virtanen et al., 2020) package in Python. With this package we can obtain the resulting data with one principal component (PC) for each of the variables of the original data-set, in addition to other data that is of interest to us like: the eigenvalues which we will be referring to as the explained variance, and the loadings which help us interpret how much the variables of the original data impact each PC.

Our first point of interest while analyzing the results from the PCA was the explained variance of the components. This is a measurement of which percent of the variance of the original data is represented by each PC. As we are interested in finding a way to reduce the number of variables we are working with on our problem, a high percentage of explained variance in a low number of PCs is what we are looking for.

For the T data-set, the results were excellent. As we hoped, regardless of the value for T, the explained variance for the first PC is above 95%, which can be visualized in Fig. 4.1(Upper plot). These results in essence mean that we can reliably use the only the first PC as a replacement for the original data in our models and have equivalent results. Focusing more on the results, a faint although clear correlation between the explained variance and the value of T can be observed, where the representativeness for the Fist PC increases as the value for T does as well.

Additionally, while further inspecting the results for the first PC, we analyzed its loadings (visualized in Fig. 4.1(Bottom plot)). From these results we observe a homogeneous contribution of each height to the component along the values of T. From this, we can interpret that all the heights play a similar role in describing the data-set, and there is none that

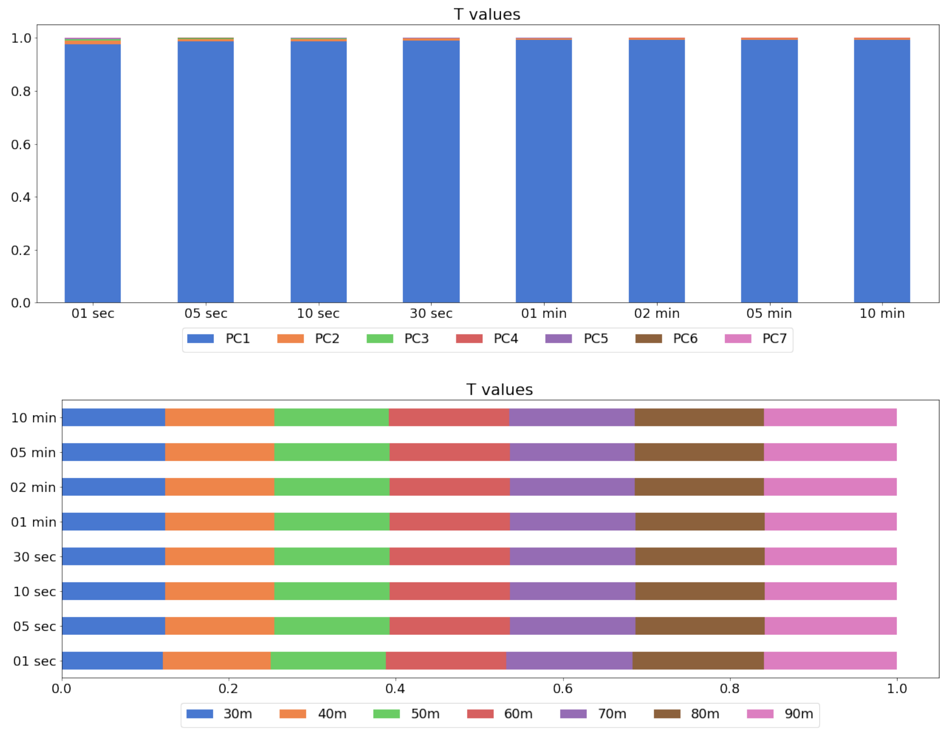
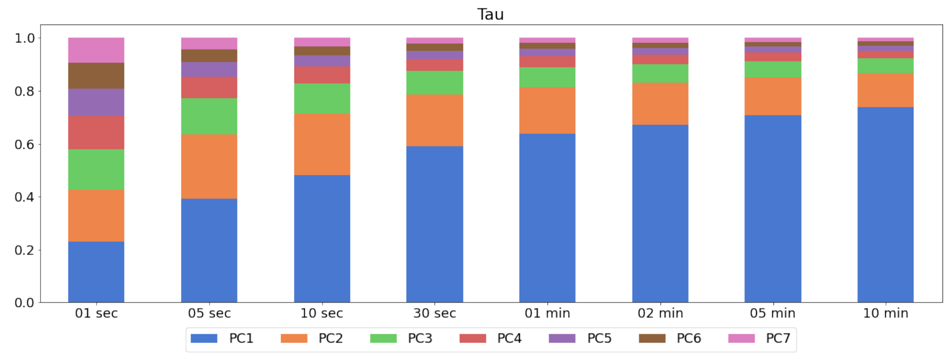


Figure 4.1: Results of PCA over the T data-set. (Above) Cumulative explained variance for each value of T. (Bottom) Loadings for the first PC for each value of T. The relevance of the first PC and the homogeneous importance of each height can be observed.

is significantly more representative than the rest.

Conversely, the results for PCA on the τ data-set weren't as positive as the ones obtained for T. Analyzing the explained variance for the τ data-sets we observe a similar behavior to the results for T, where the relevance of the first PC increases along with the value of τ. For this data-set however, the values are significantly lower. These results can be visualized in Fig. 4.2(Upper plot). In contrast with the previous results, we never get a percentage of explained variance for the first PC which we would feel comfortable using as a replacement for the original data. In fact, to get similar levels of representation (above 90%), we would need to use six of the seven principal components, which we would consider a non-significant dimensionality reduction.

Taking a look at the loadings for the Fist PC for the τ data-set (visualized in visualized in Fig. 4.2(Bottom plot)) we can get an insight on the reason for the lower representative properties of the first PC. The value for how much each height influences each component varies a lot through the different values of τ. It appears to be a correlation between the homogeneity of the importance of each height and the explained variance for the first principal component. The best results for explained variance are observed when the loadings for the first PC resemble those observed for on the T data-set. Looking back at how the data-set was constructed



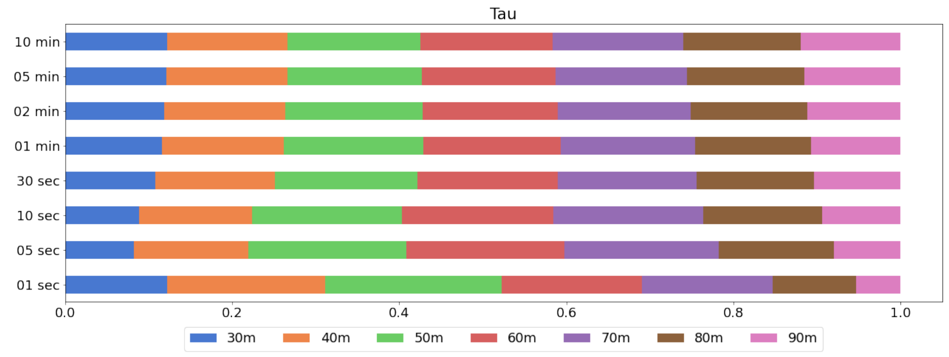


Figure 4.2: Results of PCA over the τ data-set. (Up) Cumulative explained variance for each value of τ. (Bottom) Loadings for the first PC for each value of τ. A relatively low relevance of the first PC is present through the different values of τ.

and its PDF (from Fig. 3.6) we could interpret that the distribution of the values is too similar to be able to easily represent its variance, therefore producing principal components with low associated explained variance.

At last, even if we didn't find a representation that manages to reduce the dimensionality for the τ data-set through the use of Principal Component Analysis, analyzing the data-set this way gives valuable insight. Through the innate properties of the construction of the principal components we obtained a representation of the data-set where each variable is independent from each other. This representation could lead to different conclusions as their statistical properties differ from the ones to the data-set it was produced from.

Summarizing this section we can say that we found through the use of PCA a way to represent one of our data-sets in a way that would reduce the dimensionality of the problem. Additionally, if we consider both resulting data-sets from this analysis we have more ways to represent our original $1Hz$ data in a manner that captures its essential characteristics but has different statistical properties. Exploring both of these data-set could lead to interesting findings, but in order to increase focus on answering our research questions by constricting the amount of potential results to analyze, we will be continuing this project focusing only on the T and τ data-sets, as well as the resulting data from the PCA for the T data-set due

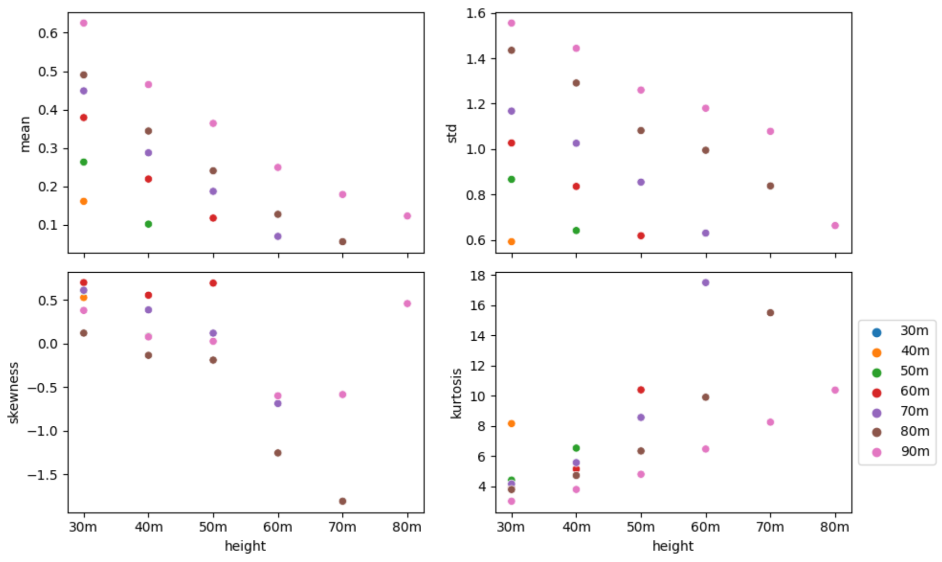


Figure 4.3: Visualization for the statistical moments for height differentials.

to its high representative prowess on its first PC.

4.2 Statistical Analysis

After having defined the data-sets we will be working with, we focused on understanding the Statistical characteristics of each one. While working on this analysis, we continuously got results from the anemometer placed at 100m which were uncharacteristically different from the rest. These results were present in every data-set and on most of the Statistical Moments, which we referenced in Section 3.1. This prompted us to review the data and FINO1's architecture and discovered that due to the different layout of equipment at this altitude, the wind speed measured from certain directions will always be different here than on the rest of the tower. To avoid any chance of this situation impacting the results, we decided to not take into consideration these measurements.

Deepening what we observed in Section 3.1, we started our Statistical section by analyzing our original data-set at $1Hz$ (already preprocessed as described in the aforementioned Section) by observing the changes in its moments as a function of a height differential, which was defined as:

$$\Delta_{h'-h} v_h(t) = v_{h'}(t) - v_h(t) \tag{4.1}$$

where $h' > h$. A visualization for these results can be observed in Fig. 4.3. To simplify the interpretation, we are omitting the results for when $h' = h$, since everything would be zero or infinite, depending on the Moment. We are also omitting the results for $h' < h$, as they are equal to the results presented but multiplied by -1. These results provide an insight into how the average Wind Profile behaves. We can observe that both the mean and the standard deviation of the Profile increase consistently as the

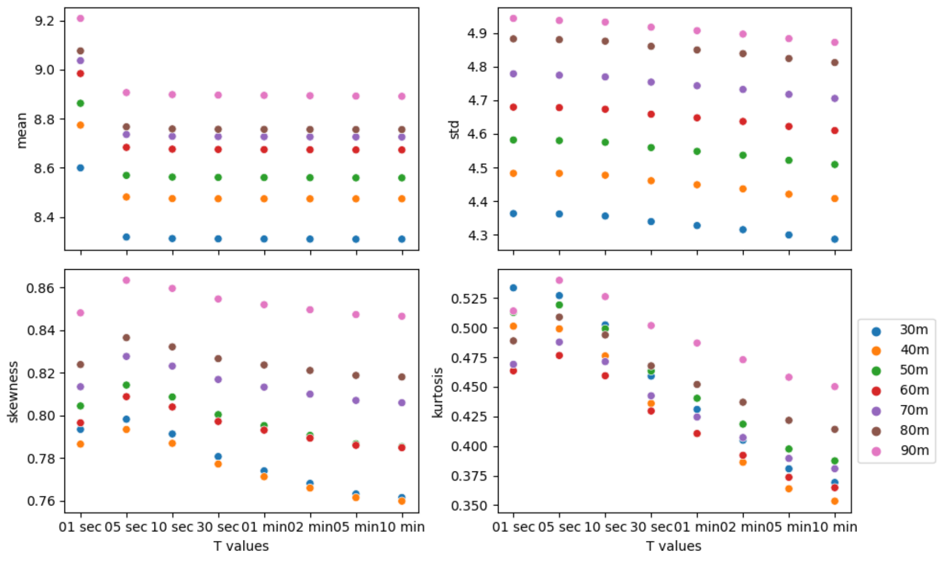


Figure 4.4: Visualization for the statistical moments for the **different T values**.

distance between the compared heights is larger, meaning that the average Wind Profile of our data-set behaves as the theoretical one. When it comes to skewness, it tends towards negative values at the two uppermost heights signaling a tendency for the higher velocity at the top of the Profile, once again consistent with the theory. Finally, when looking at the resulting values for the excess kurtosis, all values present a leptokurtic behavior, with a tendency for these values to increase along with the height. All these results point toward our original data-set behaving within the norm, thus allowing us to have results that can be generalized to typical data-sets of this kind.

After getting an idea of how the original data-set behaves, we proceeded to analyze how our simulated data-sets differ from the original data. Analyzing the results for the T data-sets, we observe a slight increase of both mean and standard deviation with height, pointing back towards the results from h' and signalling that even the resolution of the data changes, the basic Statistical Characteristics remain comparable. When observing particularly the results for mean, we can observe that the values for the original data-set (1 *sec*) are higher than the rest for all heights hinting at a dilution of extreme effects due to the averaging process. This is supported by the decreasing values of standard deviation as T increases, evidencing more homogeneous data. The values for skewness are all positive as we expected from a typical Weibull distribution. They don't present any visible correlation with height and, with the exception of the original data-set, seem to decrease as T increases, hinting at a more normal distribution but not enough to be significant. This last behavior is replicated for the values for excess kurtosis, however here we observe a non-trivial minimum at 60m. These results can be visualized in Fig. 4.4. In general, the results are non-atypical for a data-set with a Weibull distribution.

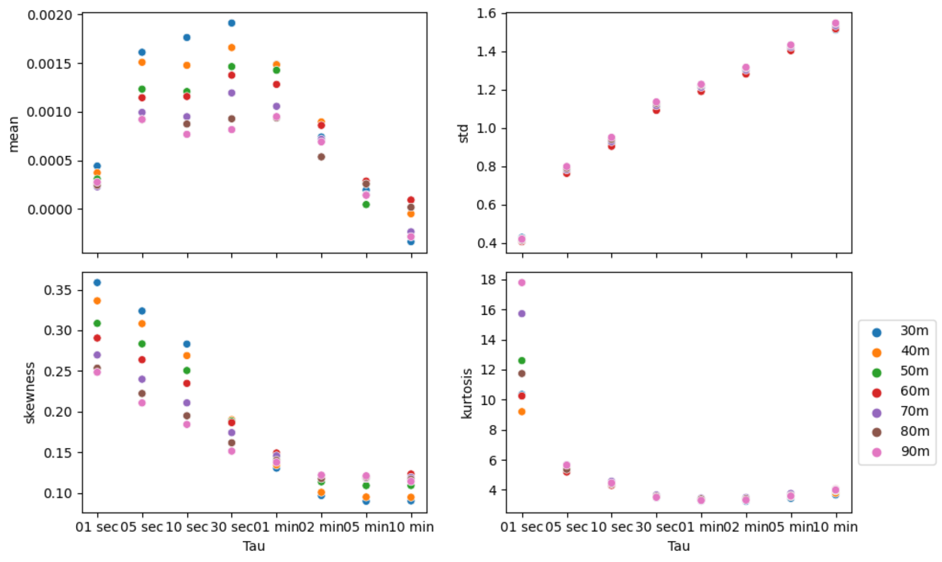


Figure 4.5: Visualization for the statistical moments for the **different τ values**.

Dissecting the Statistical Moments for the τ data-set, observed in Fig. 4.5. The values for mean seem to be independent from the height which evidences non-uniform changes of speed across the Wind Profile. Additionally, these values begin by increasing along with τ until 2 minutes, when they start to decrease until they reach a minimum at 10 minutes. It is important to note that even though patterns are observed, the resulting values are relatively small, pointing towards a stable Wind Speed. Likewise, the differences between the standard deviation between heights are small but they presents a non-trivial minimum at 60 m. Analysing across the data-sets, the standard deviation seems to be correlated with the value of τ pointing towards bigger deviations from the mean as τ increases. The skewness tends to decrease with height and with τ as well, giving the impression of a more centered curve. Excess kurtosis is the highest when $\tau = 1$ second, pointing towards a higher probability of bigger changes in speed between these measurements than on the rest. These values decrease both at lower heights and at higher values of τ. All these results combined lead us to assess that these data-sets present characteristics more similar to a Normal distribution than the T data-sets, which fall right into our expectations when creating them, and hinting towards different performance when fed to the NN models.

Finally we move to examining the Moments for the data-sets resulting from the PCA analysis over the T data-set. The value for mean seems to be independent from the order of the components. These values however, have a tendency to converge towards zero as the value of T increases. The standard deviation is highest for the first PC as expected, since this is the component that holds the majority of the variance for all the data-sets regardless of the value of T, otherwise the standard deviation for the rest of the components is doesn't have big variations. The values for skewness tend to show larger variations the larger the value of T and the order PC.

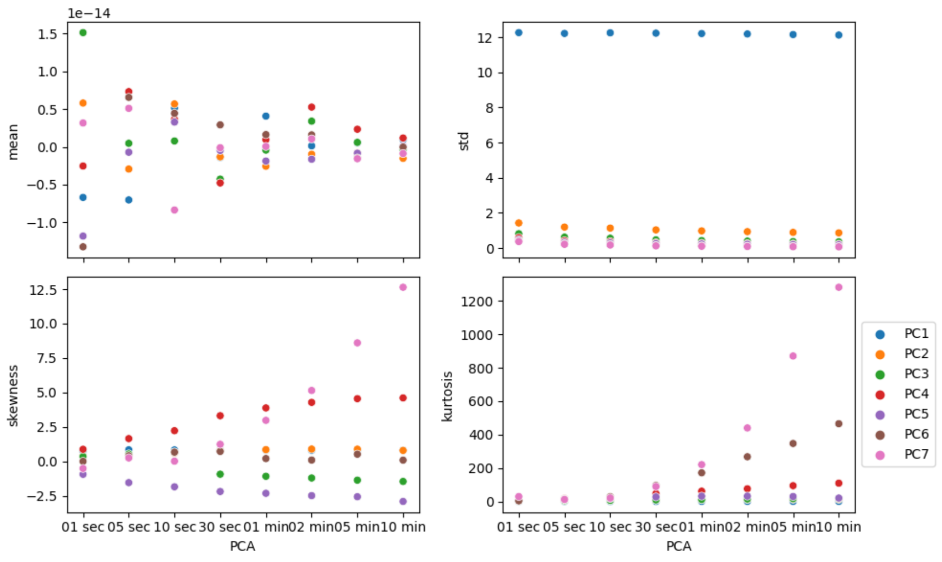


Figure 4.6: Visualization for the statistical moments for the **different PCA results** corresponding to the various T values.

Focusing on the skewnnes for the first PC, it is the closest to zero from all PCs, signalling a curve with symetric tails. The excess kurtosis increases with the order of the principal component and with T, going from almost mesokurtic, to completely Leptokurtic. Again, focusing on the first PC, the values are close to zero, which again are typical of a Normal distribution. All these results are visualized in Fig. 4.6. These results point towards a distribution having a behavior resembling a Normal distribution, with different characteristics than the τ data-sets.

After these analyses we find that our generated data-sets behave statistically as we expected, with the T data-sets presenting the characteristics of a Weibull distribution and the τ data-sets being closer to a Normal distribution. Additionally, we got a better insight into the statistical properties of the data-set resulting from the Principal Component Analysis. At this point we are satisfied with the data-sets and their Statistical Moments variety, which work in our favor while facing the next challenge which is running predictions over them. Now we are left with three data-sets with very distinct properties who are still related to each other and pointing at the same direction.

4.3 Modeling Wind Profiles using LSTM Architectures

As mentioned in Section 3.3, we built our network utilizing Pytorch as our framework. A visualization for the complete network can be observed in Fig. 4.7.

[1]We ran a small batch of information sized (100x7) to get the visualization.

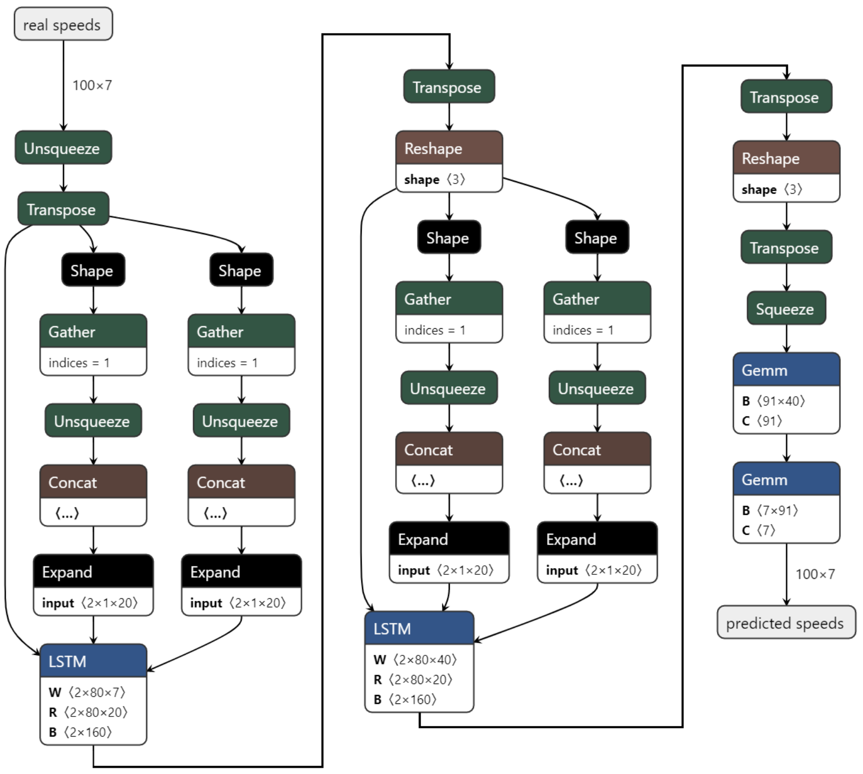


Figure 4.7: Visualization of the final LSTM model [1].

The final version consists of a two layered, bidirectional LSTM layer that takes with one node for each of the heights of one height profile. In addition, we have two fully connected layers. The first of these layers is responsible of transforming the output (just the prediction, not the hidden state) from the LSTM layer, and transform it by an expansion factor. The second fully connected layer takes the output from the first and reshapes it to the size of the wind profile. The final values for the variables like number of nodes and layers for the LSTM layer, the direction on which the information flows (unidirectional or bidirectional), the transformation factor, learning rate and batch sized were determined through experimentation, using the principles stated by S. L. Smith et al. (2017) and L. N. Smith (2018). A summary of these parameters can be reviewed in Table 4.1.

4.3.1 Available Information for the Network

One of the most important variables regarding the training and testing process, was to determine the amount of data we would allow the network to have access to in order to predict the next wind profile. To determine this amount of data to feed our network with, we used several sized of time windows to try to predict the next time step and we named this window

Parameter	**Value**
Batch Size	As close to 24h as the memory permits
Learning Rate	0.001
Optimizer	Adam
Loss Function	MEA
Number of LSTM Layers	2
Bidirectional	True
LSTM Nodes	20
Expansion Factor	13

Table 4.1: Summary of parameters used for the training cycle for the LSTM Network model.

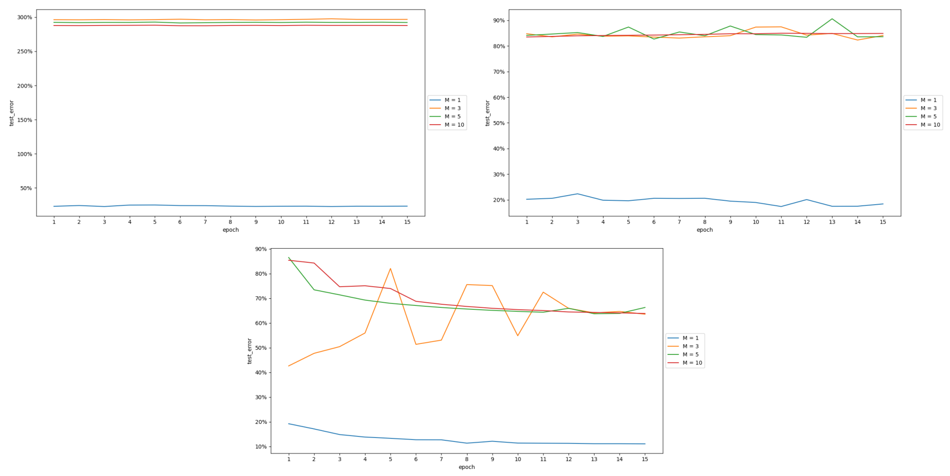


Figure 4.8: Comparative results for the LSTM architecture while being trained with different sizes of inputs (M) for different sizes of T to predict the next wind profile in the series. (Left) Trained with $T = 01$ second. (Middle) Trained with $T = 01$ minute. (Right) Trained with $T = 10$ minutes. It is apparent that the best results are obtained with lower values of M. Training done under the same parameters, with similar sized data-frames.

size "M". On a base with a T equal to 1 minute, an M equal to 10 means that we fed the net 10 wind profiles corresponding to the last 10 minutes of data to predict what the wind profile will be the next minute. Functionally, we modified the data-set as to have M number of records (corresponding to each of the previous M timestamps) as independent variables for each profile we were aiming to predict. Through the training we measured the error in the predictions using Equation 3.3.

Fig. 4.8 depicts the results for the accuracy of our net at predicting with different values of M. All the results have been normalized by dividing them by the standard deviation of the data-set used in order to make them comparable. From the figures a correlation between larger testing errors and larger values of M can be observed. Furthermore, an inverse correlation between the values for T and the testing errors can be observed, with the largest errors occurring on lower values of T. These behaviour is

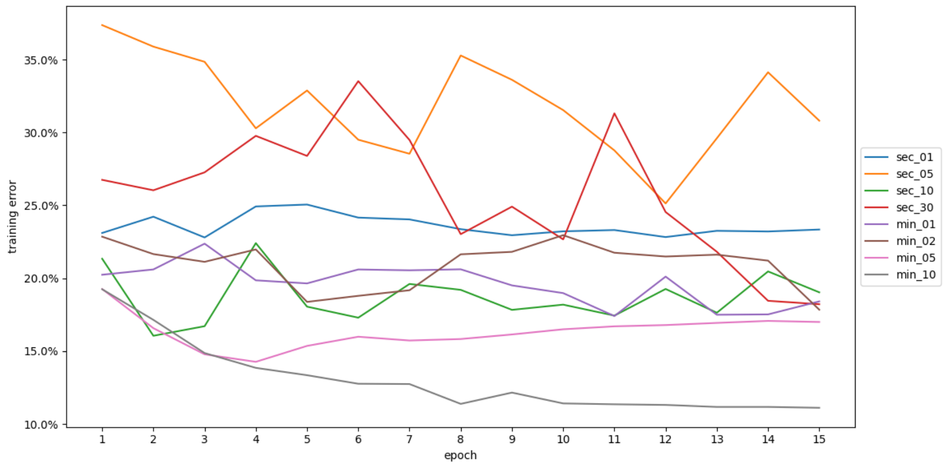


Figure 4.9: Normalized results for the LSTM network training over the T data-sets.

present equally on the training set. Additionally, the results for training loss reflect a similar tendency, with smaller loss values for smaller values of M and larger values for T.

4.3.2 Training Results

Once the value for M had been established, we used that knowledge to train the LSTM network with the different data-sets. To have comparable results we utilized the same parameters for each training cycle. Additionally, for all cases we used 75% of the data-set for training and 25% for testing the network, and due to the importance of trends along the data we didn't perform a shuffle before separating it. The results can be observed in Fig. 4.9. To make the results comparable among the different data-sets, we divided the raw prediction error over the standard deviation corresponding to the value of T groups of data over the data-set at $1Hz$.

A tendency towards better predicting capability can be more commonly observed for higher values of T than at lower ones, however no clear correlation can be distinguished. Additionally, data-sets a higher variation between the results from epoch to epoch can be observed on the data-sets where the network has higher errors. Finally, the network performed best over the $T = 10$ minutes data-set, not only having the best prediction power but also demonstrating a tendency to better results as the training went on. Contrasting these results with their corresponding Statistical Moments, we cannot observe any correlation between them. The changes on the performance for the model are above any change in any of the moments, leading us to believe that the difference increased prediction error in related to the behavior of the data-set instead of its statistical properties.

Subsequently, we proceeded training the network with the τ data-set, without making any adaptations to its architecture as to be able to have comparable results. These results can be visualized in Fig. 4.10, were we

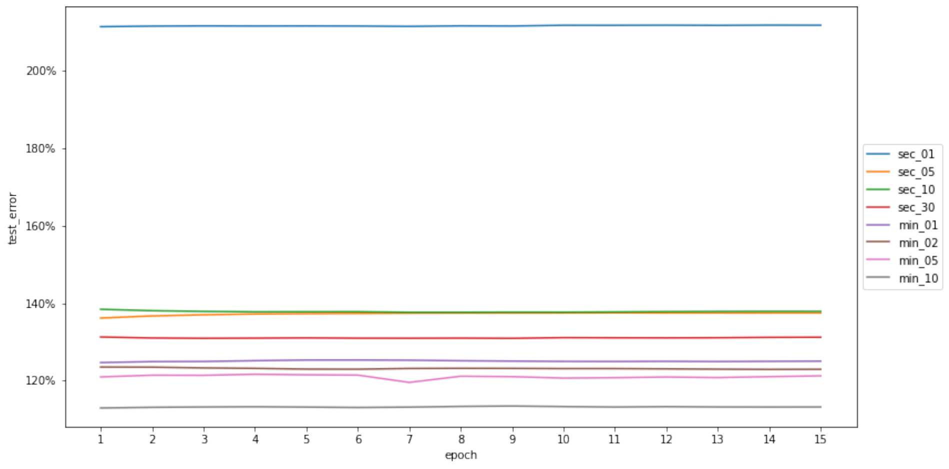


Figure 4.10: Normalized results for the LSTM network training over the τ data-sets.

can observe some similarities with the results from the T data-sets, as well as some notable differences.

The most notable result are the higher overall error values. Where as the worst results from the T data-sets were around 30%, we encounter errors over 100% for the τ data-set. Additionally, as with the previous results, the lowest error values are found on the higher values of τ. However, in this case there is visible although small correlation between the value of τ and the magnitude of the error. One more thing to note is the overall tendency to show stagnant results, regardless of the size of τ, the network doesn't seem to be able to learn from the inputs to generate better predictions. Again, making connecting these results with their respective Statistical Moments, the high relative error can be related to the small values the network is predicting upon. As observed in Fig.4.5 the mean for the different data-sets is close to zero, as well as having a low standard deviation. When calculating the error described in as 3.3, the small denominator can result in bigger relative error.

Finally, we present the results from training over the PCA data-set resulting from the T data-set. We had two different training sessions with this data-set, one with the whole output and one just utilizing the first PC, since as we noted in Section 4.1 it contains a satisfying amount of the information of the complete data-set. The results are visualized in Fig. 4.11, on the Left for the complete data-set and the Right for the first PC.

It is noteworthy observing both the differences and the similitude between these two training sessions. While the results over the complete data-set have a behavior similar to the T data-sets results (in shape, not magnitude), the results over just the first PC are significantly better and even comparable to the one from the τ data-set. The results for the training using only the first principal component are congruent with our analysis of the explained variance for it, thus the similitude on the results. As per their worsen performance, we could attribute them to a similar situation as the

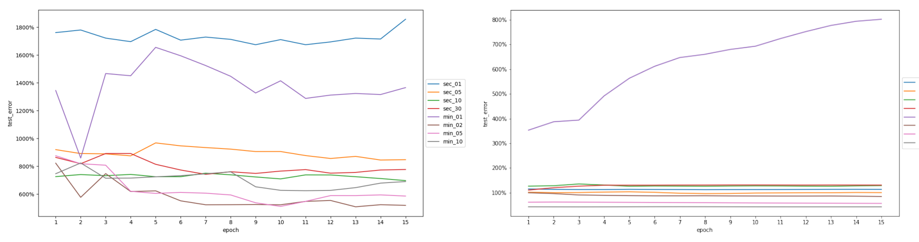


Figure 4.11: Normalized results for the LSTM network training over the PCA data-sets.(Left) Complete data-set. (Right) Utilizing only the first PC.

τ data-set, where the mean is closer to zero and the values are all relatively small. Additionally, we could relate the decrease in performance upon the data-set with only the first PC, to the big standard deviation from the set, pointing towards many values that deviate from the mean, thus making it harder to predict. The results for both data-sets do not seem to demonstrate any correlation between the value of T and the error. While the complete set demonstrates a tendency to improve the prediction, the first PC has a stagnant tendency for the most part. The results for $T = 1$ minute for the first PC set are notable as they are extremely different both in tendency and values from the rest of the data-sets.

4.4 Modelling Wind Profiles using Transformers

As we did for the LSTM model, we constructed our Transformers model utilizing the framework provided by Pytorch. For this implementation, we based our code heavily on the work by B. Wu et al. (2022), and referencing B. Wu et al. (2022) repository as it help clarifying some aspects during the implementation. The model doesn't differ greatly in architecture from the original model from Attention Is All You Need (Vaswani et al., 2017), consisting on an Encoder with four layers with a Positional Encoder layer [2]between the input and the encoding, and a Decoder. Where the Encoder takes the information and creates a memory, which is transferred to the Decoder, which in turn is responsible for making a prediction. Similarly to the LSTM model, the values for batch size, encoder/decoder size, and learning rate were determined through experimentation. As for the values for number of layers, number of attention heads, dropout rates, number of neurons, and the expansion factor for the data, we took the values that adapted best to our data-set from the works we referenced. A summary of these parameters can be observed in Table 4.2. Additionally a depiction of the model is included in Appendix A.

4.4.1 Measuring the Prediction Error

Our Transformers model, as many traditional models of this kind, uses the information from the Encoder to create a Prediction in the Decoder. This

[2]This was created according to the Torch documentation ('Language Modeling with nn.Transformer and TorchText', 2023).

Parameter	**Value**
Batch Size	As close to 24h as the memory permits
Learning Rate	0.001
Optimizer	Adam
Loss Function	MEA
Encoder Proportion	90%
Learning Rate	0.001
Expansion Factor	512
No. Encoder Layers	4
No. Decoder Layers	4
No. Attention Heads	8
No, Neurons	2048

Table 4.2: Summary of parameters used for the training cycle for the Transformers model.

prediction can vary is computed by the Decoder by analyzing a set of data that proceeds the set fed to the Encoder, and generating a prediction set of equal size. This prediction set, on an ideal scenario, will be almost identical to the set fed to the Decoder, except for the last register in the set. This last register would correspond to the prediction made by the decoder over data it has never observed. This gives the chance to measure error over the whole output of the Decoder, or only over the last register generated from the Decoder.

To illustrate the previous situation we can give a practical example: timestamps 1 to 10 are inputted in the Encoder where it is processed and the output is transferred to the Decoder to be used as a memory. The Decoder receives timestamps 11 to 14 as an input, and with the aid of the memory creates an output corresponding to timestamps 12 to 15. Since the Decoder has access to timestamps 12 to 14, timestamp 15 becomes the prediction of the model over unseen data. However, predicted timestamps 12 to 14 have gone to the same transformation an inference process that 15 has gone through, making them part of the model's prediction as well.

For our results, we have decided to present the results for the error over the last predicted profile (timestamp 15 in the previous example). We opted to present this results for two main reasons. The first being that doing this describes more accurately the prediction power of our model, as we are measuring only the error of predicting over unknown information. The second reason, and probably the most relevant, is that measuring the error in this manner let us compare these training results with the ones from the LSTM model one-to-one, without any transformations or equivalences. This error is calculated using Equation 3.3.

4.4.2 Training Results

Similarly to the process with the LSTM model, we trained the Transformers model with all our data-sets (T, τ, and PCA). Correspondingly, to have comparable results we utilized the same parameters for each training cycle

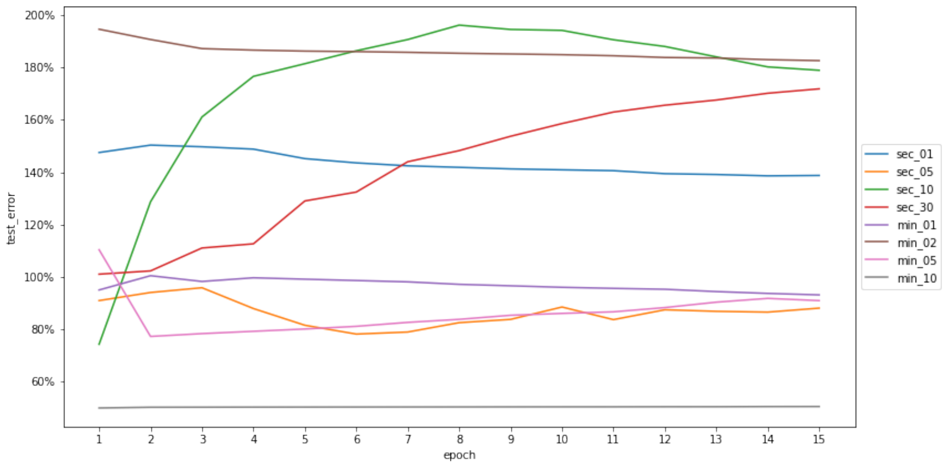


Figure 4.12: Normalized results for the Transformers model training over the T data-sets.

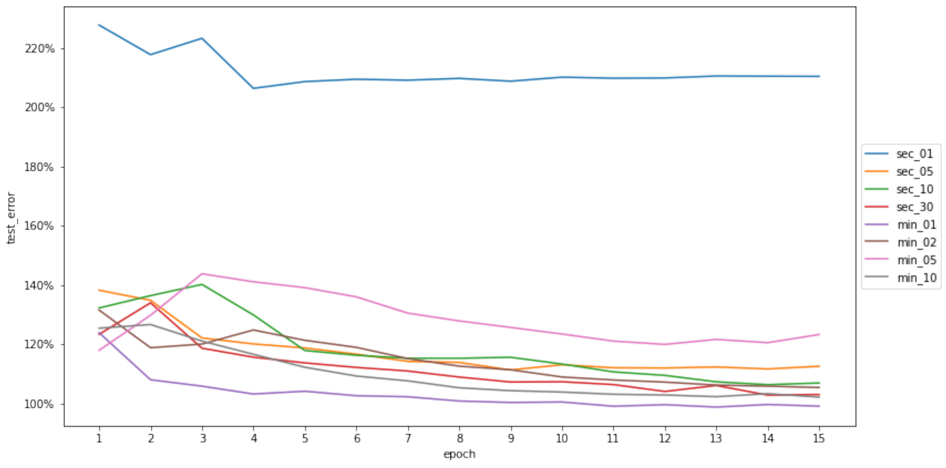


Figure 4.13: Normalized results for the Transformers model training over the τ data-sets.

regardless of the data-set. Finally, for all cases we used 75% of the data-set for training and 25% for testing the network, and due to the importance of trends along the data we didn't perform a shuffle before separating it. To make the results comparable among the different data-sets, we divided the raw prediction error over the standard deviation corresponding to the value of T groups of data over the data-set at $1Hz$.

The normalized results fro the T data-sets show no correlation between the value of T and the prediction error, these can be visualized in Fig. 4.12. Additionally, we are unable to observe any relationship between the value of T and the model's ability to improve its prediction capability with the training cycles, where for some values of T this ability improves, and for some other it deteriorates or remains stagnant. It is noteworthy that the data-set corresponding to $T = 10$ minutes produces the best results.

The results for the τ data-sets show a more consistent behavior. With

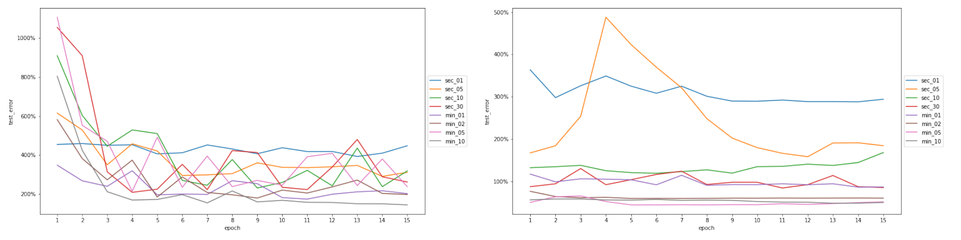


Figure 4.14: Normalized results for the Transformers model training over the PCA data-sets.(Left) Complete data-set. (Right) Utilizing only the first PC.

the exception of the data-set where $\tau = 1$ second, all of them present similar prediction error values across the whole training cycle, without any clear correlation between the magnitude of the error and the value for τ. Additionally, the model demonstrates a tendency to improve its predicting capacity through these same data sets. The only data-set where the model produces significantly different results is for $\tau = 1$ second, where the error is approximately doubled as from the other data-sets, and after some epochs the results stop improving. These results are visualized in Fig. 4.13.

Finally, observing the results for the PCA data-sets, visualized in Fig.4.14, we can compare and contrast between using the whole data-set and just the first principal component. The results are worse when observing the errors for the complete data-set in relation to just the first PC. For the results of the complete data-set, there is a slight tendency for the error to decrease as the value for T increases, as well as a heightened ability for the model to improve its results as the training cycle advances. In contrast, the results for the first PC are for the most part stagnant, with some of them worsening along the training cycle. For this data-set there is also a tendency for higher errors at lower T values. Overall, the prediction errors are larger when utilizing the complete data-set in contrast to only the first PC.

Chapter 5

Discussion and Conclusions

Generalizing our results we could conclude that this thesis points in the same direction as many of the other works in the field. As we state, when utilized to predict Wind Speed Profiles, Neural Networks demonstrate a lower prediction capability when analyzing data at a higher resolution than what's used in meteorology and in the energy industry. However, in contrast to the current work, this thesis puts a spotlight in comparing this performance in a staggered manner from the industry's standard to a $1Hz$ resolution. Next, we discuss the steps that lead us to that conclusion.

Our results from the Principal Component Analysis revealed that for the original data and the additional data-sets generated from it, T data-sets, allow for a low dimensional representation merging all seven heights into one single component describing the overall wind profile. Similar to what Skittides and Früh (2014) did, we were able to utilize this representation to train a model, in our case a neural network, that could make predictions which in turn could be generalized back to the original set. However, in contrast with Skittides and Früh (2014), we got more adequate results from training with the original data-set. As per reducing the dimensionality of our problem: when comparing the training results between using just the principal components with the most explained variance against using all the principal components to make predictions, our results were the contrary to Skittides and Früh (2014) and Y. Zhang et al. (2019). We got better results when utilizing all the Principal Components as opposed to only the most representative for the data. We feel there are too many variables to pinpoint the exact reason for this occurrence but to name some of the ones with the highest probability could be: the explained variance for the first principal component, the homogeneous loadings for it, and the architecture of the models. Another plausible explanation is that the information missing from the first principal component mostly relates to the extreme events, which neural networks struggle to model since they are best represented through the third and fourth moments in statistics. For these moments stochastic approaches seem to be more reliable as demonstrated by P. G. Lind et al. (2017), thus the need for our models to use the complete data-set instead of just the first principal component.

The statistical analysis outcome was as expected, for the original data and the additional T data-sets we generated from it. All these sets have the distinct characteristics of a Weibull distribution, much in accordance with the theory as stated by R. I. Harris and Cook (2014). Additionally, the generated τ data-sets present a distribution more similar to a Gaussian as it did for Sim et al. (2018). This combination confirmed both a correct processing of the data and an alignment with our desire to train the models with data-sets that can lead to the same conclusions but have different intrinsic statistical characteristics.

As per the AI-based approaches, the results were not as promising. With respect to the LSTM architecture, when determining the amount of information we should use as an input to get the most accurate predictions, our experimental results mirror the ones of Skittides and Früh (2014), and Geng et al. (2020). The results demonstrate a clear increase in performance when utilizing only one lag to make predictions, in contrast to inputting as much information as manageable by the experimenter's computing power. The results are somewhat similar to those of Li and Shi (2010), where this number is still relatively small but larger than one due to their data-set having a strong relationship between the analyzed measurement and the previous six as demonstrated by its Auto-correlation Function. This propensity could be the reason that Neural Networks do not thrive in the prediction of wind speeds at short time intervals and we are getting the increased use of mixed models, which are mainly AI-based approaches that incorporate Statistical Learning techniques through data processing or the NN itself, as stated in Section 2.5.

Observing the results from predicting wind profiles with a wide lens we can say that all of them support the hypothesis of decaying results as the time resolution increases. In other words, as the data gets further away from being the industry's standard of 10 minute averages, the prediction capabilities of our models get significantly worse. This phenomenon is visualized in Figs. 4.9, to 4.14, where a tendency for comparatively lower prediction errors can be observed for the data-sets with larger values of T or τ, in comparison to smaller ones. Comparing our results with the work from Cadenas et al. (2010), our results corroborate theirs at stating that when dealing with such reduced time horizon NN seem to have a decreased performance when compared to their performance when analyzing data-sets resembling the industry's standard. This phenomenon is more evident in our results when comparing our results from predicting over the τ data-set, where the data's distribution more resembles theirs. Additionally, and as stated by Liu et al. (2021) and Wang et al. (2018), when dealing with short time wind predictions, often Statistical approaches work better than neural networks, which contextualized our results.

After several training sessions over different configurations of different models, we feel confident over the parameters chosen for the models are the ones that produce the best results for our architectures. That being said, in general our prediction error was above those obtained by Geng et al. (2020) and Liu et al. (2021) when utilizing the LSTM model, and those of N. Wu et al. (2020) and B. Wu et al. (2022) when implementing the transformers. This can be attributed to differences in the model's parameters, length of training sessions, computing power, or the statistical properties of the data-set. However, we couldn't find evidence that these models were tested under the same conditions as in our experiments, with data-sets at different time resolutions, thus making direct comparisons with our whole results difficult.

With our aim of modeling the stochastic evolution of wind profiles we proposed two questions. From the results obtained through our experiments and the writing of this thesis we would like to answer as follows:

1. *What are the main statistical and dynamical features of wind profiles?*

   When analyzing the behavior of the profiles through time by analyzing the probability distribution of the data-set, it forms a unimodal bell curve with a positive skew, and a relatively low excess kurtosis. This behavior classifies the distribution as a Weibull distribution, and the distribution is maintained through all heights with slight increases to its mean as the height increases. The long tail of this pdf signalizes the presence of extreme events, although with a small probability.

   Further dissecting the change of speed through the heights, each height is more similar to the one just above and the one immediately below it when it comes to the four statistical moments. When plotting the pdf for the change in speed over height, a distribution more akin to a Gaussian is observed. The velocities increase as the height increases, leaving an average profile that resembles a logarithmic function.

   After observing the behavior of wind profiles at different time resolutions, a more stable behavior can be observed in general as the resolution decreases. The four moments present lower values. The probability of extreme events is reduced and the tails of the distribution aren't as long when dealing with averaged data.

   Finally, this behavior leads to wind profiles being a perfect candidate for a dimensionality reduction through Principal Component Analysis. Thanks to their almost uniform and predictable behavior through all the heights, all the measurements can be reliably represented through a single principal component that contains more than 95% of the variance of the whole profile.

2. *How effective is an AI-based approach to model and predict such evolution of wind profiles?*

   As per our research, AI-based approaches can have moderate to high success at modeling wind speed when under the right conditions. However, based on our experimental work, the results were not that promising for the modeling of a complete wind profile.

   Under our conditions, the selected architectures tend to perform better when predicting using only the previous measurement instead of more information. In other words, the models produce better results when only considering the changes in speed from one wind profile to the next, making the prediction of extreme events worse. To back up this claim, the results seem to be worse when predicting over data with values of skewness and kurtosis pointing towards a higher likelihood of extreme events.

Additionally, these AI-based approaches demonstrate a decreasing performance when the time resolution of the data is increased, where in contrast stochastic methods seem to thrive. In our case, the LSTM model outperformed the more complex transformers model, giving under certain conditions errors as low as ten percent. Nevertheless, we cannot conclude that an AI-based approach could be effective at predicting the stochastic evolution of wind profiles at a short time scale when analyzing them on a high time resolution.

In a broad sense, we can state that our results mostly mirror in behavior the vast amount of results found in our literature search. Overall, perhaps the major difference between our results and the ones we found in the literature could be summarized as the increased error measurements we are reporting and the contrasting results between using only the principal components with the most explained variance. Additionally, the literature has more cases of multivariate wind speed predictions with the inclusion of other measurements other than just Speed (i.e. direction, temperature, atmospheric pressure). Nonetheless, we feel that our results succeed in demonstrating the decreasing performance of a couple of the most common neural network models over predicting Wind Profiles at a high time resolution. We feel that the industry standard of ten (or higher) minute average, aside from being well explored and developed, functions well to cover the current need. Nevertheless, the industry could vastly benefit from having more precise predictions.

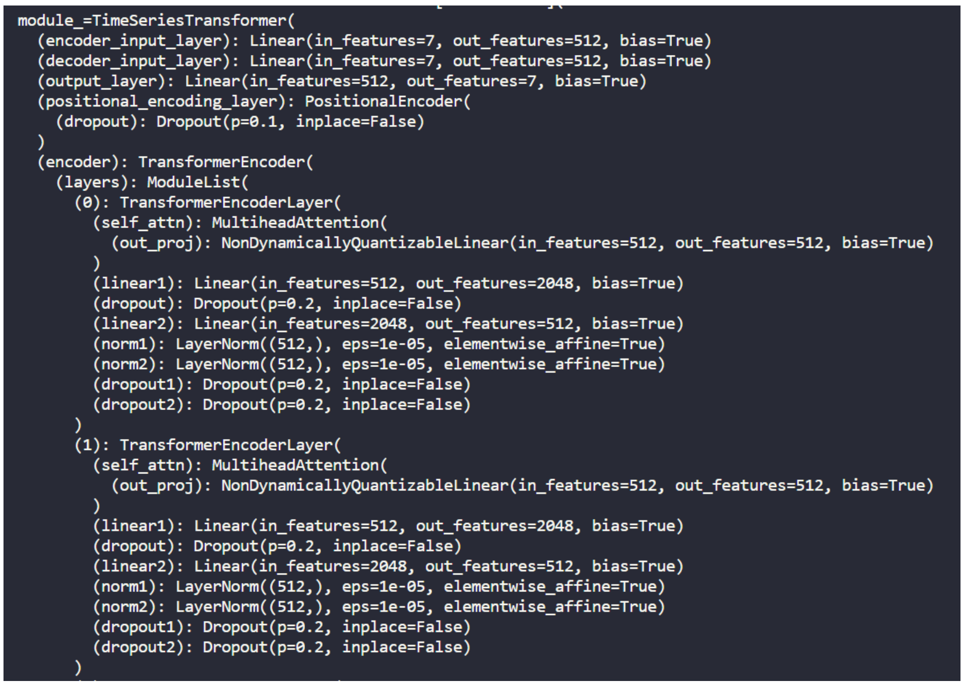
That being said, we would be remiss to not mention the aspects of this project that could be improved to provide more concrete results and give more conclusive evidence of our claims. The first would be utilizing a better method of determining the optimal parameters for the models than trial en error. Additionally, we could include other Neural Networks or Machine Learning approaches as GANS, SVM, or pre-trained models, which all have proven successful under the correct circumstances in the literature. A good example is the work by P. Lind et al. (n.d.), where they conclude that Markov processes are able to model these kind of processes while simultaneously being simpler than AI-based approaches. Moreover, another improvement could come as creating a more contrasting picture by having the representation of more Statistical/Mathematical (as the work by Liu et al. (2021), Wang et al. (2018), and the aforementioned P. Lind et al. (n.d.)) methods to predict wind speeds at short horizons. This inclusion could lead to a stronger statement if these results improved as the time resolution increases as opposed to our current results.

All these potential improvements could be a natural continuation this Thesis, aiming towards giving even more robust evidence of the declining efficiency of Neural Networks at predicting wind speeds on a short time scale. Additionally, this project presents other interesting opportunities to branch off from. One of the most interesting possibilities is the development of a method whose performance doesn't drastically deteriorate as the resolution of the data increases, prioritizing its ease of use. Most of the solutions we found that work at this resolution is almost tailor-made, making a transition from using the industry's methods complicated. By focusing on producing a tool that is flexible enough can be packaged and distributed as a single tool, we could encourage the use of data with high resolution for wind forecasting. This model most likely would be a mixed model including Machine and Statistical Learning. Finally, as the field is getting a constant influx of methods that mix Neural Networks and Mathematical approaches, it could be interesting to go against the current research and develop a Neural Network model that could outperform these methods. This is again with the aim of increasing its availability outside the scientific and research community.

Appendix A

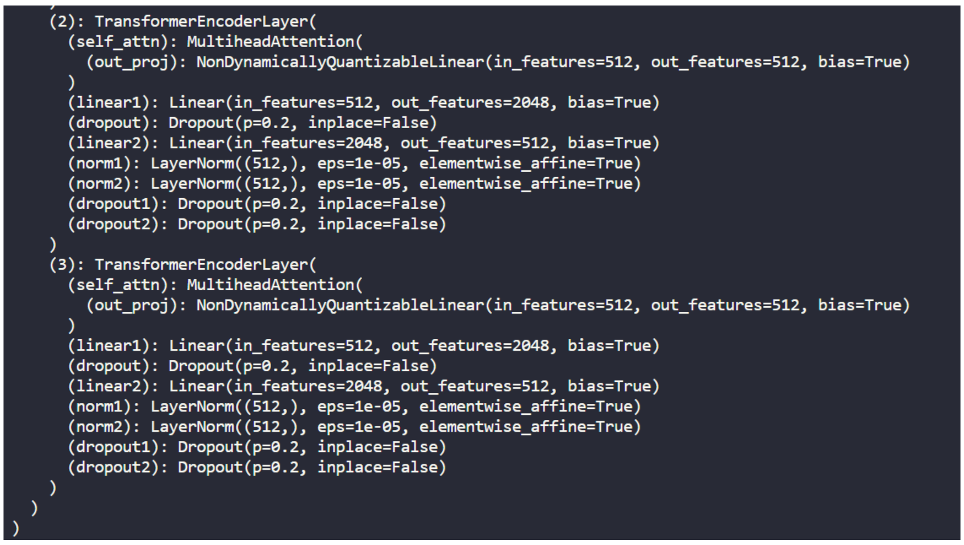
Transformers Depiction

In this appendix we present a depiction of our transformers model mentioned in Chapter 4. We decided to present it this way, as it gives a view over the entire model which was too complex to depict as we did for the LSTM model in Fig.4.7. As mentioned in the Chapter, the model starts with an Encoder section, formed by four encoding layers. Each of this layers consisting of input, positional encoding, encoder. Then the outputs from the encoder are transferred to the decoder, which again is composed of four layers.



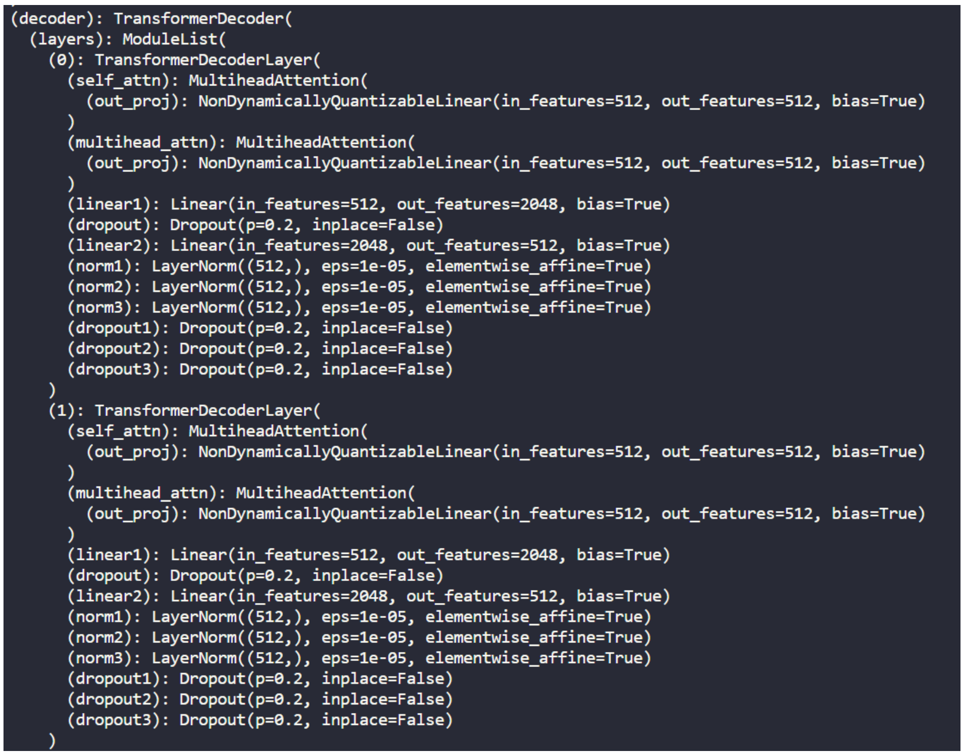
```
module_=TimeSeriesTransformer(
  (encoder_input_layer): Linear(in_features=7, out_features=512, bias=True)
  (decoder_input_layer): Linear(in_features=7, out_features=512, bias=True)
  (output_layer): Linear(in_features=512, out_features=7, bias=True)
  (positional_encoding_layer): PositionalEncoder(
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (encoder): TransformerEncoder(
    (layers): ModuleList(
      (0): TransformerEncoderLayer(
        (self_attn): MultiheadAttention(
          (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
        )
        (linear1): Linear(in_features=512, out_features=2048, bias=True)
        (dropout): Dropout(p=0.2, inplace=False)
        (linear2): Linear(in_features=2048, out_features=512, bias=True)
        (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        (dropout1): Dropout(p=0.2, inplace=False)
        (dropout2): Dropout(p=0.2, inplace=False)
      )
      (1): TransformerEncoderLayer(
        (self_attn): MultiheadAttention(
          (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
        )
        (linear1): Linear(in_features=512, out_features=2048, bias=True)
        (dropout): Dropout(p=0.2, inplace=False)
        (linear2): Linear(in_features=2048, out_features=512, bias=True)
        (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        (dropout1): Dropout(p=0.2, inplace=False)
        (dropout2): Dropout(p=0.2, inplace=False)
      )
```

Figure A.1: Overview of the general structure of the model and its parameters for the first two layers in the encoder.



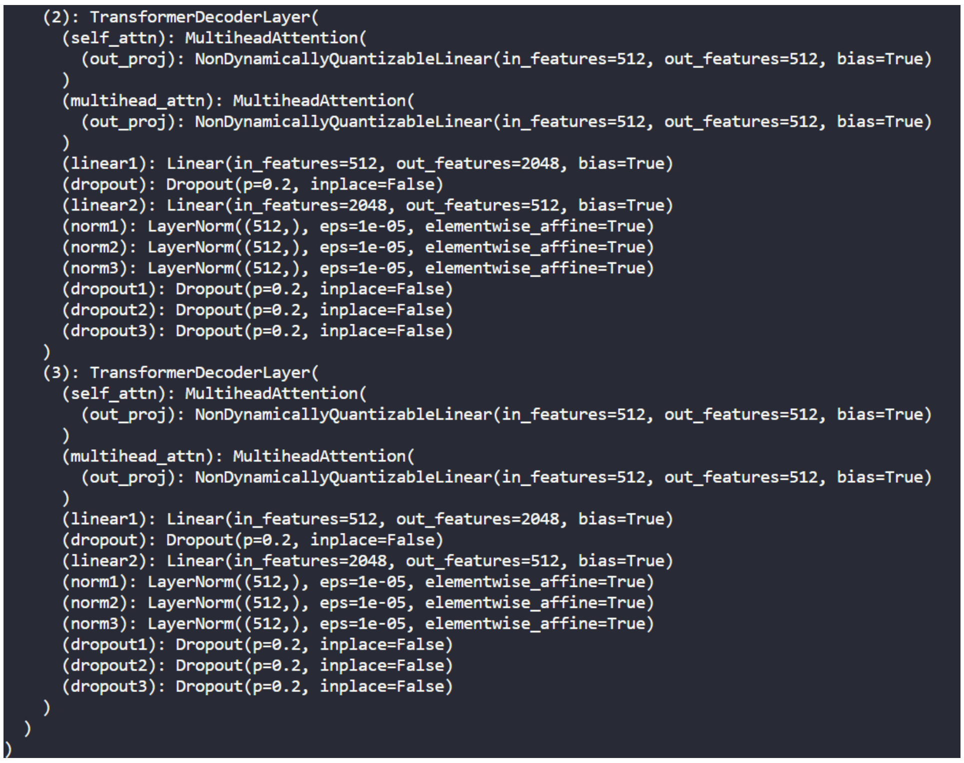
```
    (2): TransformerEncoderLayer(
      (self_attn): MultiheadAttention(
        (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
      )
      (linear1): Linear(in_features=512, out_features=2048, bias=True)
      (dropout): Dropout(p=0.2, inplace=False)
      (linear2): Linear(in_features=2048, out_features=512, bias=True)
      (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (dropout1): Dropout(p=0.2, inplace=False)
      (dropout2): Dropout(p=0.2, inplace=False)
    )
    (3): TransformerEncoderLayer(
      (self_attn): MultiheadAttention(
        (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
      )
      (linear1): Linear(in_features=512, out_features=2048, bias=True)
      (dropout): Dropout(p=0.2, inplace=False)
      (linear2): Linear(in_features=2048, out_features=512, bias=True)
      (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (dropout1): Dropout(p=0.2, inplace=False)
      (dropout2): Dropout(p=0.2, inplace=False)
    )
  )
)
```

Figure A.2: Overview of the model's parameters for the remainder two layers in the encoder.



```
(decoder): TransformerDecoder(
  (layers): ModuleList(
    (0): TransformerDecoderLayer(
      (self_attn): MultiheadAttention(
        (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
      )
      (multihead_attn): MultiheadAttention(
        (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
      )
      (linear1): Linear(in_features=512, out_features=2048, bias=True)
      (dropout): Dropout(p=0.2, inplace=False)
      (linear2): Linear(in_features=2048, out_features=512, bias=True)
      (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (norm3): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (dropout1): Dropout(p=0.2, inplace=False)
      (dropout2): Dropout(p=0.2, inplace=False)
      (dropout3): Dropout(p=0.2, inplace=False)
    )
    (1): TransformerDecoderLayer(
      (self_attn): MultiheadAttention(
        (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
      )
      (multihead_attn): MultiheadAttention(
        (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
      )
      (linear1): Linear(in_features=512, out_features=2048, bias=True)
      (dropout): Dropout(p=0.2, inplace=False)
      (linear2): Linear(in_features=2048, out_features=512, bias=True)
      (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (norm3): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (dropout1): Dropout(p=0.2, inplace=False)
      (dropout2): Dropout(p=0.2, inplace=False)
      (dropout3): Dropout(p=0.2, inplace=False)
    )
```

Figure A.3: Overview of the model's parameters for the first two layers in the decoder.



```
    (2): TransformerDecoderLayer(
      (self_attn): MultiheadAttention(
        (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
      )
      (multihead_attn): MultiheadAttention(
        (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
      )
      (linear1): Linear(in_features=512, out_features=2048, bias=True)
      (dropout): Dropout(p=0.2, inplace=False)
      (linear2): Linear(in_features=2048, out_features=512, bias=True)
      (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (norm3): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (dropout1): Dropout(p=0.2, inplace=False)
      (dropout2): Dropout(p=0.2, inplace=False)
      (dropout3): Dropout(p=0.2, inplace=False)
    )
    (3): TransformerDecoderLayer(
      (self_attn): MultiheadAttention(
        (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
      )
      (multihead_attn): MultiheadAttention(
        (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
      )
      (linear1): Linear(in_features=512, out_features=2048, bias=True)
      (dropout): Dropout(p=0.2, inplace=False)
      (linear2): Linear(in_features=2048, out_features=512, bias=True)
      (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (norm3): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (dropout1): Dropout(p=0.2, inplace=False)
      (dropout2): Dropout(p=0.2, inplace=False)
      (dropout3): Dropout(p=0.2, inplace=False)
    )
  )
)
```

Figure A.4: Overview of the model's parameters for the last two layers in the decoder.

References

Wind power. (2022). Retrieved January 12, 2022, from https://www.statkraft.com/what-we-do/wind-power/

Data science workshop - wind energy analytics, a practical approach. (2021). Retrieved January 10, 2022, from https://www.youtube.com/watch?v=5QcbOS0LMjo&t=410s

The fino1 project. (2022). Retrieved February 1, 2022, from https://www.fino1.de/en/about-fino1.html

What is drag? (2021). Retrieved February 3, 2022, from https://www.grc.nasa.gov/www/k-12/airplane/drag1.html

Rieutord, M., Dubrulle, B. & Lévêque, E. (2006). An introduction to turbulence in fluids, and modelling aspects. *European Astronomical Society Publications Series*, *21*, 7–42.

Wind turbines: The bigger, the better. (2021). Retrieved February 7, 2022, from https://www.energy.gov/eere/articles/wind-turbines-bigger-better

Pennstate: Stat 414 | introduction to probability theory. (2022). Retrieved January 19, 2022, from https://online.stat.psu.edu/stat414/lesson/14/14.1

Wolfram mathworld: Kurtosis. (2022). Retrieved January 19, 2022, from https://mathworld.wolfram.com/Kurtosis.html

Devore, J. L., Berk, K. N. & Carlton, M. A. (2012). *Modern mathematical statistics with applications*. Springer.

Harris, R. I. & Cook, N. J. (2014). The parent wind speed distribution: Why weibull? *Journal of wind engineering and industrial aerodynamics*, *131*, 72–87.

World meteorological organization. (2022). Retrieved May 4, 2022, from https://public.wmo.int/en/about-us

World meteorological organization - library. (2022). Retrieved May 4, 2022, from https://public.wmo.int/en/our-mandate/what-we-do

World meteorological organization - library. (2018). Retrieved May 4, 2022, from https://library.wmo.int/doc_num.php?explnum_id=3177

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2019). *An introduction to statistical learning: With applications in r*. Springer. https://faculty.marshall.usc.edu/gareth-james/ISL/

Greene, W. H. (n.d.). *Maximum likelihood estimation*. https://dx.doi.org/10.4135/9781526421036878705

Genschel, U. & Meeker, W. Q. (2010). A comparison of maximum likelihood and median-rank regression for weibull estimation. *Quality Engineering*, *22*(4), 236–255.

Shumway, R. H. & Stoffer, D. S. (2000). *Time series analysis and its applications* (Vol. 3). Springer.

Partial autocorrelation function. (2022). Retrieved May 18, 2022, from https://online.stat.psu.edu/stat510/lesson/2/2.2

Olson, D. L. & Wu, D. (2020). Autoregressive models. *Predictive data mining models* (pp. 79–93). Springer Singapore. https://doi.org/10.1007/978-981-13-9664-9_6

Kruse, R., Borgelt, C., Braune, C., Mostaghim, S., Steinbrecher, M., Klawonn, F. & Moewes, C. (2011). *Computational intelligence*. Springer.

Scarselli, F. & Tsoi, A. C. (1998). Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results. *Neural networks*, *11*(1), 15–37.

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi, A. & Jenssen, R. (2017). *Recurrent neural networks for short-term load forecasting: An overview and comparative analysis*. Springer.

Understanding lstm networks. (2015). Retrieved May 12, 2022, from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Smagulova, K. & James, A. P. (2019). A survey on lstm memristive neural network architectures and applications. *The European Physical Journal Special Topics*, *228*(10), 2313–2324.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Bahdanau, D., Cho, K. & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Gpt-3 powers the next generation of apps. (2022). Retrieved 2022, from https://openai.com/blog/gpt-3-apps/

Child, R., Gray, S., Radford, A. & Sutskever, I. (2019). Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J. et al. (2021). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, *118*(15).

Ferret, J., Marinier, R., Geist, M. & Pietquin, O. (2019). Self-attentional credit assignment for transfer in reinforcement learning. *arXiv preprint arXiv:1907.08027*.

Geneva, N. & Zabaras, N. (2022). Transformers for modeling physical systems. *Neural Networks*, *146*, 272–289.

Cadenas, E., Jaramillo, O. A. & Rivera, W. (2010). Analysis and forecasting of wind velocity in chetumal, quintana roo, using the single exponential smoothing method. *Renewable Energy*, *35*(5), 925–930.

Li, G. & Shi, J. (2010). On comparing three artificial neural networks for wind speed forecasting. *Applied Energy*, *87*(7), 2313–2320.

Skittides, C. & Früh, W.-G. (2014). Wind forecasting using principal component analysis. *Renewable Energy*, *69*, 365–374.

Geng, D., Zhang, H. & Wu, H. (2020). Short-term wind speed prediction based on principal component analysis and lstm. *Applied Sciences*, *10*(13), 4416.

Yu, J., Fu, Y., Yu, Y., Wu, S., Wu, Y., You, M., Guo, S. & Li, M. (2019). Assessment of offshore wind characteristics and wind energy potential in bohai bay, china. *Energies*, *12*(15), 2879.

Salim, O. M., Dorrah, H. T. & Hassan, M. A. (2019). Wind speed estimation based on a novel multivariate weibull distribution. *IET Renewable Power Generation*, *13*(15), 2762–2773.

Milan, P., Wächter, M. & Peinke, J. (2014). Stochastic modeling and performance monitoring of wind farm power production. *Journal of Renewable and Sustainable Energy*, *6*(3), 033119.

Wu, N., Green, B., Ben, X. & O'Banion, S. (2020). Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317*.

Khodayar, M. & Wang, J. (2018). Spatio-temporal graph deep neural network for short-term wind speed forecasting. *IEEE Transactions on Sustainable Energy*, *10*(2), 670–681.

Yu, M., Zhang, Z., Li, X., Yu, J., Gao, J., Liu, Z., You, B., Zheng, X. & Yu, R. (2020). Superposition graph neural network for offshore wind power prediction. *Future Generation Computer Systems*, *113*, 145–157.

Lind, P. G., Vera-Tudela, L., Wächter, M., Kühn, M. & Peinke, J. (2017). Normal behaviour models for wind turbine vibrations: Comparison of neural networks and a stochastic approach. *Energies*, *10*(12), 1944.

Islam, M., Mohandes, M., Rehman, S. et al. (2017). Vertical extrapolation of wind speed using artificial neural network hybrid system. *Neural Computing and Applications*, *28*(8), 2351–2361.

Wang, L., Li, X. & Bai, Y. (2018). Short-term wind speed prediction using an extreme learning machine model with error correction. *Energy Conversion and Management*, *162*, 239–250.

Liu, X., Lin, Z. & Feng, Z. (2021). Short-term offshore wind speed forecast by seasonal arima-a comparison against gru and lstm. *Energy*, *227*, 120492.

Zhang, Z., Ye, L., Qin, H., Liu, Y., Wang, C., Yu, X., Yin, X. & Li, J. (2019). Wind speed prediction method using shared weight long short-term memory network and gaussian process regression. *Applied energy*, *247*, 270–284.

LeHau, H. H. (1959). Wind profile, surface stress and geostrophic drag coefficients in the atmospheric surface layer. *Advances in geophysics* (pp. 241–257). Elsevier.

Scheffer, J. (2002). Dealing with missing data. *Research Letters in the Information and Mathematical Sciences*, *3*, 153–160.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., . . . Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

McKinney, W. et al. (2010). Data structures for statistical computing in python. *Proceedings of the 9th Python in Science Conference*, *445*, 51–56.

Sim, S.-K., Maass, P. & Lind, P. G. (2018). Wind speed modeling by nested arima processes. *Energies*, *12*(1), 69.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., . . . Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., . . . SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Ilie, A. G., Scarisoareanu, M., Morjan, I., Dutu, E., Badiceanu, M. & Mihailescu, I. (2017). Principal component analysis of raman spectra for tio2 nanoparticle characterization. *Applied Surface Science*, *417*, 93–103.

Wu, B., Wang, L. & Zeng, Y.-R. (2022). Interpretable wind speed prediction with multivariate time series and temporal fusion transformers. *Energy*, *252*, 123990.

Smith, S. L., Kindermans, P.-J., Ying, C. & Le, Q. V. (2017). Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*.

Language modeling with nn.transformer and torchtext. (2023). Retrieved September 18, 2022, from https://pytorch.org/tutorials/beginner/transformer_tutorial.html

Zhang, Y., Chen, B., Pan, G. & Zhao, Y. (2019). A novel hybrid model based on vmd-wt and pca-bp-rbf neural network for short-term wind speed forecasting. *Energy Conversion and Management*, *195*, 180–197.

Lind, P., Lencastre, P., Gjersdal, M., Yazidi, A. & Gorjão, L. R. (n.d.). Modern ai versus century-old mathematical models: How far can we go with generative adversarial networks to reproduce stochastic processes? *Available at SSRN 4305494*.