uc3m | Universidad Carlos III de Madrid

Master in Statistics for Data Science
2024-2025

Master Thesis

"Optimizing Wind Turbine Placement in Wind Parks via Mixed Integer Optimization using Neural Network based Constraint Learning."

Simon Schmetz

Carlos Ruiz Mora
2nd Tutor complete name
Madrid the 29. of January

Acknowledgments

TODO Acknowledgments

ABSTRACT

This thesis combines Linear Optimization with Constraint Learning to optimize wind turbine placement for maximum performance in a predefined area under randomly distributed wind. A Neural Network is trained on simulated data to model the impact of turbine positioning on power output. This model is integrated as a constraint in a linear optimization problem, and the problem evaluated for a current state-of-the-art wind farm configuration.

CONTENTS

1

INTRODUCTION

With the clean energy transition currently taking place in Europe with ambitious targets for 2030 and beyond [11], wind energy is playing a central role in that transition, with wind energy expected to rise to 50 % in the EU energy mix. [5] With wind energy thus expected to become the main contributer to the EU's energy production and large potentials identified for both onshore and offshore parks [1] attempts to optimize all parameters of windparks with even minor power efficeny improvement can be expected to yield significant returns in absolute power due to the scale of future wind energy production.

Within the total generated power by wind energy lies the subproblem of optimizing the layout of wind farms. Here, the main goal is to reduce the negative impact that wake effects between wind turbines have on overall power generation, with yield reduction of up to 15% mainly due to reduced wind speeds in wake regions. Optimizing the farm for overall minimal wake exposure between wind turbines can thus potentially significantly increase power yield. [16] [19]

In practice, the problem reduces to placing wind turbines within a predefined zone, subject to the wind conditions as shown in 1. These wind conditions can be assumed to be deterministic or (more accurately) as random variables, by considering probability distributions for variables like wind direction and wind speed.

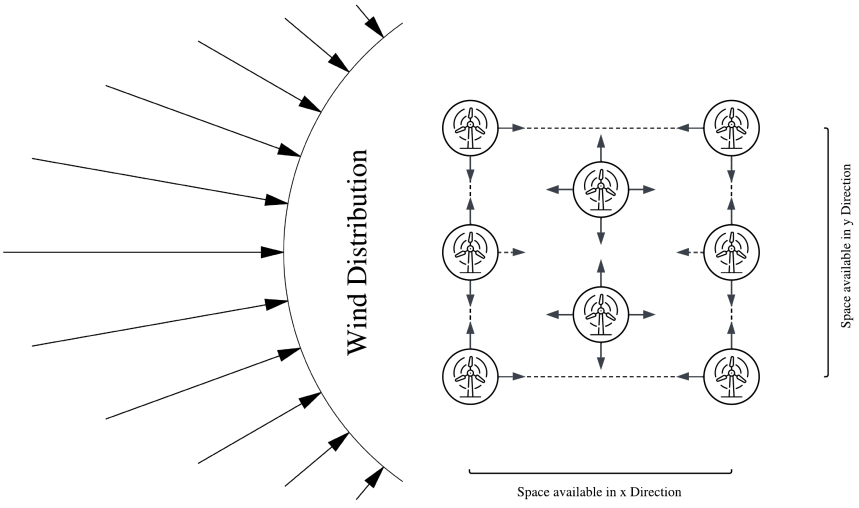

Figure 1: Optimizing the total power output of a farm reduces to placing wind turbines within a set space for the farm, subject to the wind conditions at the given location

In mathematical terms, this problem can be expressed as the attempt to maximize the sum of power output across all turbines, e.g.,

maximizing total farm output, subject to the limitations of the defined space, which is assumed to be rectangular:

$$\max_{\mathbf{x},\mathbf{y}} f_{Power}(x_i, y_i, wind_conditions) \tag{1}$$

$$\text{s.t.} \quad 0 \leq x_i \leq X_{\max} \tag{2}$$

$$0 \leq y_i \leq Y_{\max} \tag{3}$$

$$\tag{4}$$

where:

- $(\Delta x, \Delta y)$ are the relative distances of the two turbines
- $f_{Power,\mathrm{NN}}(\Delta x, \Delta y)$ is a neural network (deterministic) approximating the total power output
- $X_{\max}, Y_{\max}$ define the maximal distance the two turbines can be placed apart
- $d_{\min}$ is the minimum distance between the two turbines

This thesis is dedicated to a new approach for optimizing the placement of such fixed number of wind turbines in a predefined space, beginning with the two-turbine problem, e.g. the problem of optimally placing two turbines relative to each other. To solve this optimization problem, an extension to the Pyomo Python library is used, which allows the introduction of Neural Networks to the optimization problem as constraints. [3] This extension allows for introducing a Neural Network to model the effects of wind turbine placement relative to each other on power production for the respective wind turbines. Introducing this model to the optimization problem defined in Pyomo then allows for the optimization of overall power production across all wind turbines in the wind park. The optimization problem in its simplest form can be defined as

To create a model optimally fit to the needs of the optimization problem, the model is trained on data specifically generated with the FLORIS wind farm simulation tool for optimal coverage of the parameter space of the optimization. To simplify the problem, the surface below the turbines is assumed to be perfectly flat and an equal wind speed is assumed along the entire height of the turbines. Solving the problem can be seperated into two main Steps

1. *Farm Power Model:* Generation of simulation data covering the parameter space and training a Neural Network model with power generation as output
2. *Optimization:* Setting up optimization problem, embedding of power model and solving

This thesis is structured according to these two main steps, with a brief review of the state-of-the-art beforehand.

2

STATE OF THE ART

Since the arrival of large-scale wind turbine farm operations as part of energy infrastructure, optimizing the positioning of the individual wind turbines relative to each other to mitigate wake effects and reduce the total power output is the subject of scientific investigation. As this Thesis is an attempt to apply a novel constraint learning method introduced by Alcantara and Ruiz in [3] to the optimization of wind farm layouts, the following state-of-the-art is split into two pieces, with the first investigating the current state of the art of wind farm optimization and the second a brief introduction into recent developments made in the field of constraint learning.

Optimization of Wind Farm Layouts

As discussed in the Introduction, one of the main goals in the optimization of layouts is to reduce the negative impact wake effects between wind turbines [19]. Historically, the initial approaches were to use rule of thumb approaches by setting up the layout as a grid and with the distance between wind turbines in the dominant wind direction between 8 and 12 times the turbine rotor diameter and spacing perpendicular to the dominant wind direction 4 to 6 times the turbine rotor diameter [6] [16].

These methods have evolved to with most current research pursuing the goal of maximizing the Annual Energy Production (AEP) of wind farms in the context of stochastic optimization as done in [26] [19].

The core of any of the most recent optimizations is a wake model, which becomes part of the objective function by representing the wake effects on power output. These models can be categorized as [31]:

1. Experimental Methods
2. Numerical modeling
3. Analytical/semi-empirical modeling
4. Data-driven modeling

While experimental methods and numerical models might be the most precise models available for wake modeling, one of the challenges that come with the optimization is that the model has to be able to be introduced into the current state-of-the-art solvers as part of an objective function, leading to the prevalent use of analytical wake models like Gaussian wake model and the 3D wake model [31]. With advancements in machine learning the field of data-driven modeling is meanwhile expanding, with successful attempts of introducing Neural Networks and other Machine Learning frameworks into

optimizations of wind farm layouts. Generally, either experimental data or data from numerical modeling (more prevalent) is used to train a chosen model type. The resulting model is then introduced into the optimization problem, as done in [33] [7] [28] [29].

Constraint Learning

The term Constraint Learning, defined as "finding a set of constraints, a constraint theory, that satisfies a given dataset" by Raedt et al., is the intersection of machine learning and optimization or more in practical terms, the introduction of machine learning models into optimization problems as constraints. As the models learned (from a given data set), the constraints resulting from such a model are thus equally learned. [13]

For this thesis specifically, the novel constraint learning method of decomposing a neural network into a set of constraints using a big M approach introduced in [3] and [4] is the foundation. Similar approaches of embedding machine learning models into optimization problems have been taken for Decision Trees and Random Forests in [9] or again for Neural Networks, but without integer variables required as done in [12]. A survey performed by Fajemisin et al.[14] shows how the field is currently emerging with an increase in publications in recent years and most publications revolving around the Embedding of Neural Networks and Decision Trees/Random Forests.

3 FARM POWER MODEL

The first central component in optimizing the wind farm layout is to generate a data-driven surrogate Model that can be introduced into the optimization problem and solved by a solver. As detailed in the introduction, more specifically, the aim is to use the distCL extension to the Pyomo Python package, requiring a small Neural Network as a surrogate model. The following chapter documents the steps taken to generate such a model. To train such a Neural Network, data is required that covers the parameter space of the optimization to prevent extrapolation by the model. Therefore, the chapter starts by explaining how the open-source wind farm simulation tool FLORIS® was used to generate a dataset and what the simulations behind this dataset are. Then, the fundamentals of Neural Network architecture and training are briefly introduced, before the model of the interactions for two turbines is trained and evaluated.

3.1 DATA SOURCE

The power output of wind turbines and wind farms as a whole is fundamentally connected to the aerodynamic conditions in the airflow every wind turbine experiences, with wind speed as the biggest factor relevant to how much power a wind turbine can generate. The main effect reducing the power generated by a wind turbine is to be positioned in the wake downwind of another turbine. This power reduction is primarily a result of the reduction in windspeed joined with the increased turbulence in the wake airflow (see 2) [20]. Moving further downstream of a wind turbine, the wake gradually mixes with the outer airflow and thus again increases windspeed until the entire airflow reaches a new homogeneous air speed [22]. Thus, even if a wind turbine is positioned in the wake of another wind turbine, the greater the distance between the two, the less the second wind turbine is affected. Ideally, wind turbines are not positioned in the wake of other wind turbines at all.

Figure 2: Visible wind turbine wakes due to contrail generation [32]

With the overall goal of creating a model that models how the interaction between wind turbines (induced by the wake) is related to the wind turbines relative positioning to each other and the wind conditions as input to an optimization model, a tailor-made dataset from simulation results is the easiest to handle to ensure that the data fits the use case well. Using self-generated simulation data allows to set the parameter space of the model as required for the different variants of the optimization problems explored in the second part of this thesis. After investigating two Python-based open source wind farm simulation tools FLORIS (FLORIS official website) and PyWake (PyWake Documentation), FLORIS was chosen after a brief comparison between the two due to its apparent ease of use. The FLORIS is a wind farm simulation tool developed by the National Renewable Energy Laboratory (NREL) as one of multiple tools containing models of varying complexity available for different use cases. FLORIS is a control-systems-oriented toolbox and therefore contains a lower complexity model yielding fast simulation results. This allows the automatic generation of large quantities of data in a relatively short time. As the data is coming from simulation results, the data is deterministic even though there is an underlying uncertainty attributed which can be quantified by FLORIS if required.

For this thesis, depending on the optimization problem, FLORIS is given a grid of the following parameters:

- Position of Wind Turbines
- Wind Direction
- Wind Speed
- Wind turbulence

defining the parameter space limits with evenly spaced data points to then generate the corresponding simulation data, with the simulation outputs being the generated power by the individual wind turbines as well as by the farm as a whole (e.g. the sum of individual power outputs). The configuration of all of these parameters, as well as the number of wind turbines, can be modified at will in this setup and the result is a dataset with the rows corresponding to a specific combination of parameters and the resulting power outputs of the turbines. As a turbine type, an IEA 10-MW is used for data generation due to its repeated use as a baseline turbine in other scientific works like [21] and [18], a more exact technical specification of the turbine can be found in [10]. The configurations of the specific dataset generated for the individual optimization problem formulations are documented in the corresponding Sections of Chapter 4.

3.2 INTRODUCTION TO NEURAL NETWORKS

To model the relationship between the attributes of an incoming airflow to a wind turbine and the output generated by the same wind

turbine, many surrogate models could be chosen. As the model generated for this thesis is created to be then introduced into the Pyomo extension referenced in Chapter 1, the model has to be compatible with said extension. That is why the model chosen is a simple neural network with a limited number of hidden layers and nodes, with the size of the Neural Network being limited due to its introduction into the optimization problem.

In the following section, a brief introduction is given to how Neural Networks work and are trained, including some regularization techniques.

Architecture of a Neural Network

Fundamentally, Neural Networks represent Graph Networks consisting of function blocks as the nodes, in the case of Neural Networks called perceptrons (or Neurons as a more general term) and arcs which correspond to the inputs/outputs of a given perceptron. In simple words, the inputs to the Neurons are summed up and introduced as an argument into a function $f()$, which yields an output y, representing the output of the given perceptron. These Neurons are organized in layers, which correspond to the row-type structure Neural Networks are usually represented in and each perceptron of one layer is connected to every other perceptron of the following layer. The following layer, in this case, means in the direction of flow, in visualizations usually from left to right. The arcs thus in simple terms correspond to a single number, and the Neurons to the action of summing up all inputs and then applying the unspecified function $f()$ to generate an output. This process is repeated for all Neurons in the network, with the first layer of the network taking as input the values of the given data features (the input values to the model) and the last layer producing outputs corresponding to the output value(s) of the model. The number of Neurons in this first and last layer corresponds to the task the Neural Network is supposed to perform. If the goal is to identify handwritten numbers 1-9 from 50x50 pixel images, the input layer might have 50x50 Neurons for the value of each pixel, and the output layer 9 layers with the output value of each perceptron representing how much the network "thinks" the given image shows the corresponding numbers 1-9. A schematic of this architecture is given in Figure 3.

As shown in Figure 4 the output of a Neuron is slightly more complicated as described before, as the output y is generated by summing up the inputs x multiplied by a corresponding weight w together with a bias b and introducing this summation as an argument into an activation function $f()$. In this process, the weights w represent a weight to give importance to the individual inputs and the bias b serves to set a minimum output value that will always be reached, regardless of the inputs.

The activation function $f()$ is called the activation function, as in its simplest form it represents a step function that decides if a neuron activates or not, e.g., takes the binary values $0, 1$ for a given threshold.

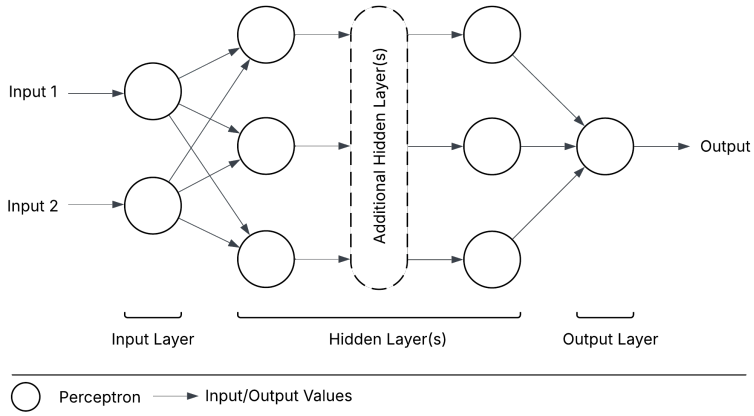

Figure 3: Schematic of Neural Network Architecture

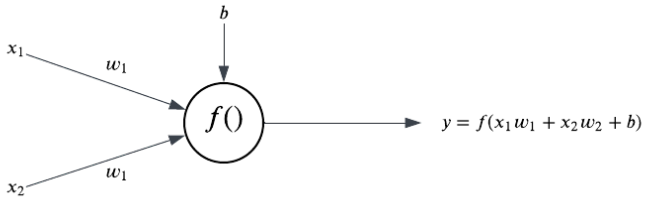

Figure 4: The output of a neuron is generated by applying the activation function to the sum as the weighted inputs $w_i x_i$ and the bias of the neuron b

Contrary to the human brain, where neurons are indeed binary, most Neural Networks resort to an activation function whose outputs are not binary but deliver continuous values between 0 and 1 to avoid the boundary issues that occur with binary thresholds. The most common function used instead of a step function is the sigmoid function, which roughly corresponds to a continuous version of the step function, with $\sigma(x) \approx 1$ for $x \to \infty$ and $\sigma(x) \approx 0$ for $x \to -\infty$.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

This relationship also becomes apparent when plotting both of those functions over each other, as shown in Figure 5. An alternative to the sigmoid as activation function is the Rectified Linear Unit (ReLu) function, or in simple words, the maximum function, defined as follows.

$$\mathrm{ReLU}(x) = \max(0, x) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{if } x \geq 0. \end{cases}$$

The main general applicable benefit of the ReLu function is that it has a constant, non-vanishing gradient for both directions (see Figure 5), leading to better gradient propagation, an attribute that is relevant in the context of the backpropagation algorithm briefly discussed in Section 3.2. [15] Beyond this, the ReLu function allows for embedding

the Neural Network into a conventional optimization problem, as will be discussed in Section 4.2.

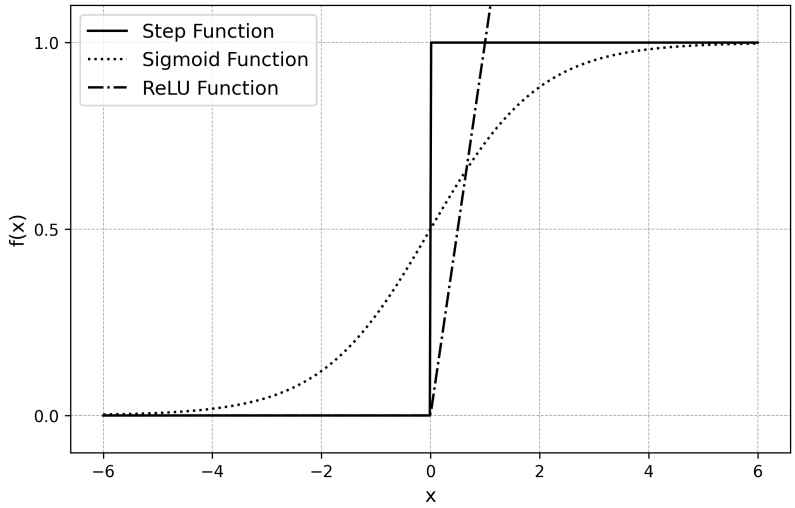

Figure 5: The Sigmoid Function being close to the Step Function without being as sensitive to slight changes in x due to its continuity. Alternatively, the Rectified Linear Unit function (ReLu) can be used as activation function.

[24]

Training a Neural Network

With the general structure set up, and assuming that a layout has been chosen for the neural network (e.g. number of layers and number of their corresponding neurons), the training of the neural network corresponds to adjusting the weights w and the biases b of each neuron in a way, that allows the model to perform the task it is given well. What it means for the model to perform well is defined by a *Loss Function*, which defines a relationship between the output of the model and the correct output (defined by training data) and gives a Loss as output, which is some sort of delta between model prediction and truth. What it means for a model to perform well heavily depends on the task (regression, binary classification, multiclass classification, etc.), but regardless of the task, the goal is always to minimize the Loss Function. A well-known Loss Function in regression is the Mean Squared Error (MSE)

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

The training of the Neural Network thus corresponds to adjusting the weights and biases in a way that minimizes the chosen loss function. The algorithm most commonly used for this is called *Backpropagation*. [24]

Before diving into how the algorithms work exactly, it makes sense to first define specifically what parameters have to be optimized, e.g., all weights and biases of the network to be optimized. A common

notation of the individual weights is as w^l_{jk} with l being the layer of nodes into which w^l_{jk} are the weights of its inputs (weighing the values from the outputs of the $(l-1)^{th}$ layer) and j as the neuron from the l^{th} layer as well as k the neuron of the $(l-1)^{th}$. Similarly, the biases are written as b^l_j with l as the layer and j the neuron in that layer as as can be seen for both the weights and bias in an example in Figure 6.

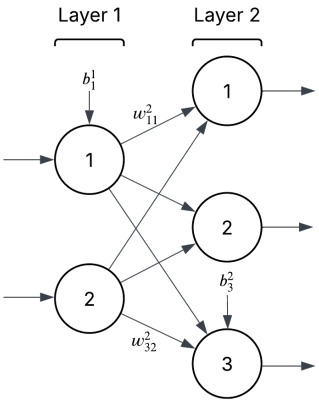

Figure 6: Notation of Neural Network weights and biases with weight w^l_{jk} and l being the input layer, j the neuron from that input layer and k the neuron of the $(l-1)^{th}$ output layer. Similarly, the bias as b^l_j with l as the layer and j the neuron.

Using this notation, the output of a given neuron a^l_j can be calculated as

$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right),$$

This expression can be further simplified by writing weights and biases in matrix form, using for each layer, a weight matrix w^l for all weights input into the given layer l and a bias vector b^l. By doing so, the output vector a^l of the entire layer l can be calculated as.

$$a^l = \sigma(w^l a^{l-1} + b^l)$$

With the notation for weights and biases established, the goal becomes tuning them in such a way as to minimize a chosen Loss function as discussed above. This is done by using training data with a known correct output and changing weights and Biases in such a way that moves the Neural Network towards outputting the true known value. The algorithm to do so essentially corresponds to:

1. Introduce Training Data into the Neural Network

2. Evaluate the Loss Function for the Training Points (with small random values for weights and biases for the first iteration) and evaluate the gradients of the Loss function for the gradients of

the individual weights and biases ($\frac{\Delta Loss}{\Delta w^l_{jk}}$, $\frac{\Delta Loss}{\Delta b^l_j}$) using backpropagation [1]

3. Update the weights and biases according to a learning rate ρ weighted by the specific weight or biases gradient like $w^l_{jk} = w^l_{jk} - \rho \frac{\delta Loss}{\Delta w^l_{jk}}$ and $b^l_j = b^l_j - \rho \frac{\Delta Loss}{\Delta b^l_j}$

4. Repeat process n Epochs, with one Epoch being the algorithm passing through all training points/batches of points once

This algorithm represents a gradient descent algorithm with learning rate ρ corresponding to the step length while evaluating the partial derivatives $\frac{\Delta Loss}{\Delta w^l_{jk}}$, $\frac{\Delta Loss}{\Delta b^l_j}$ is the critical step. Here, the backpropagation algorithm gives an efficient method of finding these partial derivatives by evaluating the individual error contributions of each Neuron and how errors accumulate as the inputs move through the Network. A more extensive explanation of how backpropagation works can be found in [25]. As is common for gradient descent, finding the global optimum for the weights and biases is not guaranteed and is even improbable if many local minima exist, as they do for the many parameters of a Neural Network. Nonetheless, using gradient descent, it is hoped to find a good local minimum. [17] [25]

Regularization

Like most machine learning models, Neural Networks run at risk of overfitting. Neural Networks are especially prone to overfitting due to their many degrees of freedom represented by the many biases and weights, potentially exceeding the number of training samples.

To prevent overfitting, regularization techniques like Cross-Validation can be applied in training, as for any other model. One very specific regularization method for Neural Networks is *Dropout Learning*, which corresponds to randomly removing Neurons from the network (by effectively setting their activation function and thus their output to 0) for each training observation. By other nodes having to "stand in" for the dropped-out nodes, nodes are prevented from developing overspecialization. [17]

Notes - many Degrees of Freedom, many local maxima - Adaptive Moment Estimation (ADAM) optimization

Estimating Distributions using Neural Networks

Neural Networks can also be used to estimate distributions of a random variable. Two possible approaches are to so-called Quantile Neural Networks and Distributional Neural Networks. While their Architecture is that of a normal Neural Network as discussed in the previous section, they differentiate in the output layer, where the Quantile Neural Network has outputs that correspond to the probability for a

1 The Gradient can either be evaluated for each training point individually or for batches of training points

specific quantile while the distributional Neural Network outputs the distribution, like in Figure 7 for a normal distribution.

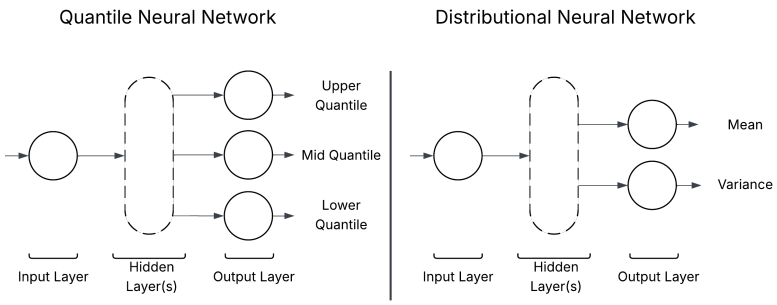

Figure 7: Neural Networks to estimate a distribution by estimating specific quantiles (left) or the parameters of a distribution (right)

The different outputs from both neural Networks require appropriate Loss functions. For Quantile Neural Networks, the Pinball Loss (Quantile loss, for more see [27]) is used, while in the case of Distributional Neural Networks, the Likelihood (like in the form of the Negative Log Likelihood) can be used as commonly used in classical statistics for parameter estimation.

[2] [23]

3.3 MODELLING PIPELINE

With the theoretical foundations laid out, this section is going to treat the actual training process of the Neural Networks to be introduced into the optimization. Part of the challenge of training Neural Networks for this specific purpose is that the objective in optimizing the Neural Network is not only to minimize the cost function, but also to minimize the number of model parameters, e.g., to minimize the total number of neurons the network consists of. This is because each neuron corresponds to an additional three linear constraints in the optimization problem, leading the number of constraints to grow as $\Delta n_{constraints} = 3 \times \Delta n_{neurons}$. In general, individual models are trained for each optimization problem seen in Chapter 4 tailored to their individual needs and parameter space. This is why this chapter will detail the main components of the modeling process and parameters that will be defined according to the optimization problem later on. The exact configuration of the models will then be defined together with the optimization problem.

The modelling pipeline contains three main components, with the elements consisting of the data generation, hyperparameter tuning (training), and the final model training as seen in in Figure 8, where data generation corresponds to running the required simulations via FLORIS for the given parameter space, hyperparameter tuning corresponds to the training and evaluation of different model configurations and final model training as the step in which the final model that is to be introduced into the optimization problem is trained.

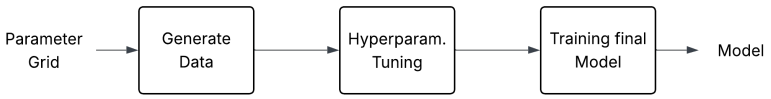

Figure 8: Flow of model generation, with parameter grid as input and the final model as output

Data Generation

The Data Generation corresponds to the creation of a dataset corresponding to a grid that is set up within the parameter space of the optimization problem. The objective is to generate a dataset that covers the parameter space with sufficient density to train a model that is able to accurately interpolate between the given data points. Each data point corresponds to an individual simulation performed using the tool FLORIS, allowing for an array of inputs and delivering an array of outputs. As inputs, the following variables can be defined as part of the parameter grid:

- x_range
- y_range
- wind_speeds
- wind_directions
- turbulence_intensities

Where x_range and y_range correspond to the relative distances of the second wind turbine to the first, wind speed corresponds to wind velocity in m/s, direction to the incoming wind angle in Degrees (with 0° corresponding to norht, 180° to south) and turbolence intensities as a measure of turbolence for the incoming airflow. Figure 9 shows an example of a simulation output.

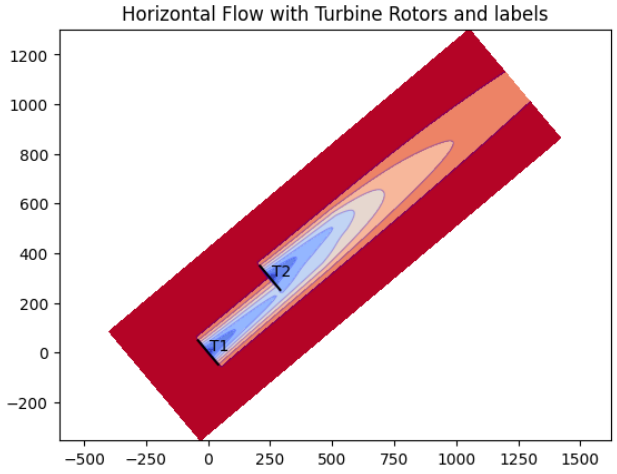

Figure 9: Example of a Simulation output from the FOLORIS Wind Turbine Simulation Tool with colormap corresponding to airflow velocity

The method implemented in (Data Generation Routine Github Link) takes a range of values together with a specified number of chosen steps for the specific parameter and performs a simulation for all combinations for the given ranges to fill the entire grid. As the number of steps or limits may be different for the different parameters, the grid is not necessarily equally spaced for all parameters, but for each parameter individually is constant, e.g. the distance between values is the same for the entire range. Depending on the configuration of the problem, some parameters may be constant. If, for example, a variable like turbulence intensity is not taken into account for a given optimization, a constant value is set, and the data is generated for that constant value. The model trained on this dataset is thus constrained in use to the assumption of that specific constant, a limitation that is important to keep in mind further down the road when the initial dataset generation might not be as conscious anymore.

The outputs of the simulation are:

- Total Power Generation
- Power generated by Turbine 1
- Power generated by Turbine 2

where the total power generation corresponds to the summation of the power generated by the individual turbines. To maximize total farm output, the total power generation will now be used as a target for the model in the following steps.

Neural Network Configuration

The next step in the process corresponds to the hyperparameter tuning of the Neural Network. According to the chosen type of Neural Network (distributional Neural Network or single output Neural Network), an appropriate loss function has to be chosen first. The data is then split into a training and an evaluation set, and the model is trained on the training partition using the algorithms detailed in 3.2, using backpropagation/gradient descent as well as dropout learning for regularization for a chosen number of times.

Once the training process is ended, the model is used to predict for the test partition of the data, and the results are evaluated by scoring the predictions using the Mean Squared Error (MSE).

This process is repeated for a grid of the hyperparameters:

- Number of Hidden Layers
- Number of Nodes in each hidden layer

and a plot like shown in Figure 10 generated to decide what the best configuration of the Neural Network is.

The final decision on what is the ideal model configuration is not only based on the best performance, but via the trade-off between complexity and output, to generate a model that is as small as possible. The kind of optimization problem the model will be used to solve is further taken into account, with complex optimization problems having less tolerance for a complex model.

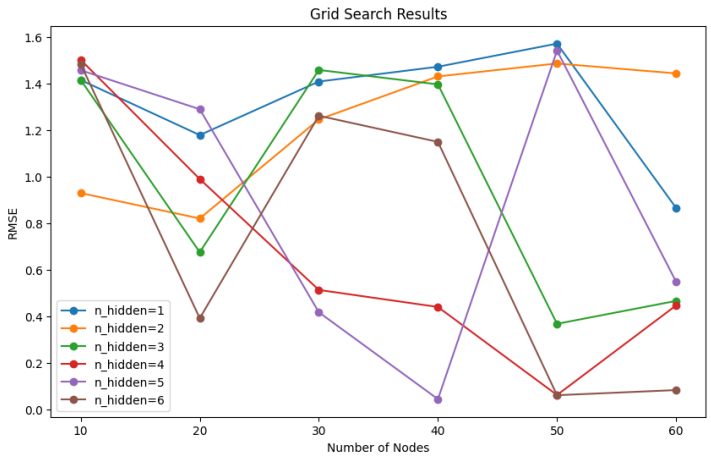

Figure 10: Exemplary Output of Hyperparameter tuning with number of hiden layers n_{hidden} and number of nodes per hidden layer

Final Model Training

After having chosen a specific model configuration, a model is then trained using that configuration. This is the model that is eventually introduced into the optimization problems detailed in Chapter 4.

3.4 POSSIBLE IMPROVEMENTS IN THE MODELLING PROCESS

As one of the two main components of the optimization process, the model generation step shows some potential for improvement in the future. The currently evenly spaced grid of data points could be replaced by a grid that is organically generated according to the gradient of the underlying function by providing a denser grid in areas of large changes in the power function to provide the Neural Network with more data in these areas. The result could be an improved accuracy and efficiency, with data points concentrated on areas of interest.

To reduce the size of the Network, the number of Neurons could be allowed to change from hidden layer to hidden layer and allow for removing expendable Neurons.

To improve the accuracy of the models, a higher-quality simulation method could be used to generate the dataset. While this might come at a computational cost, the more accurate data might lead to an increase in the accuracy of power output prediction, which, depending on the application context, might be worthwhile.

Beyond these two proposals, it is safe to assume that a wide range of methods are available to improve performance and efficiency. These might be worthwhile to pursue once the method developed in this thesis has been shown to provide a valuable new approach to wind turbine positioning optimization.

4

OPTIMIZATION

With the model of farm/turbine power established, the following section treats the embedding of the trained model(s) into a stochastic optimization problem. The underlying theory of optimization under uncertainty and constraint learning is introduced before the actual optimization problems are defined, the required models trained, and the results discussed.

4.1 OPTIMIZATION UNDER UNCERTAINTY

Like in many areas, knowledge about the uncertainty associated with certain variables can change decisions in which these variables' underlying uncertainty plays a part. That means that while optimizing anything from next year's crop to tomorrow's energy pricing, the field of Stochastic Programming is occupied with finding methods that allow for introducing these uncertainties into optimization problems.

One way to approach these uncertainties is to split the problem into scenarios. A bakery, for example, has to decide on how many Baguettes to bake for the next day to maximize its profit. Baking too few Baguettes will lead to missing out on potential sales, while baking too many baguettes will mean that the demand is fulfilled, but the money invested in the excess number of baguettes is lost. Assuming the mean number of baguettes bought every day is 1000, the price to buy a baguette is 1 and the cost to produce a baguette is $0,2$ the classical approach to solving such a problem would be by the following formulation [1]:

$$\max_x \quad (1.00 \cdot \min(x, 1000) - 0.20 \cdot x)$$

To introduce the two scenarios, additional scenarios of the demand being 10% lower (900 Baguettes) and 10% higher (1100 Baguettes) can be added. Assuming that the mean demand has a 50% probability of occurring and the 10% demand increases a 20% probability, and the decrease a 10% probability, the problem can be modified to maximize the expected profit across these three scenarios by the formulation

$$\begin{aligned}\max_x \quad & 0.5 \cdot (1.00 \cdot \min(x, 1000) - 0.20 \cdot x) \\ & + 0.3 \cdot (1.00 \cdot \min(x, 900) - 0.20 \cdot x) \\ & + 0.2 \cdot (1.00 \cdot \min(x, 1100) - 0.20 \cdot x)\end{aligned}$$

1 x of course non-negative

The result from this optimization problem would be the Expected profit and how many Baguettes are the optimal number of Baguettes to yield the maximum Expected profit across all scenarios. This would be optimal assuming there is no more information about the next day's demand, meaning that by using this approach, the total profit over a long time would be maximal. In case there is more information regarding the next day's demand, the probabilities that give the weights in this optimization might shift, with one scenario potentially reaching probability 100% if there were to be absolute certainty that the next day's demand would be, for example, 1100 baguettes. As having such exact information is very rare, the best solution will be in most cases to maximize the profit Expectation.

The obvious connection to conventional statistical analysis is that the demand is a random variable that can take multiple values, in this case, we assumed it to be a discrete random variable Y with support $900, 1000, 1100$ even though in real-world applications the demand of baguettes will move somewhere between $[0, \infty]$. Finding the Expectation for such a discrete random variable can be done as

$$\mathbb{E}[X] = \sum_i x_i \cdot \mathbb{P}(X = x_i) = \sum_i x_i p_i$$

or for continuous random variables

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x \cdot f_X(x)\, dx$$

Using these two expressions, optimization problems can thus be formulated to optimize the Expectation of objective functions containing random variables. [8]

4.2 CONSTRAINT LEARNING

Constraint learning refers to introducing a model that has learned relationships between certain variables from data into an optimization problem. In the case of constraint learning, the model is more specifically introduced into an optimization problem as part of a constraint. As many real-life relationships struggle to be represented by an explicit function to be defined as an objective function or constraint, introducing machine learning models to optimization problems opens up many new possibilities [14].

In the case of Neural Networks, one way of introducing a Network as a constraint into an optimization problem is by recognizing that when using the Rectifier Linear Unit (ReLu) Function is used as an activation function with the (linear) sum of neuron bias and weighted inputs $\tilde{v}_i^\ell$ being the function argument

$$v_i^\ell = \max(0, \tilde{v}_i^\ell) = \max(0, b_i^\ell + \sum_j w_{ij}^\ell v_j^{\ell-1}) \tag{5}$$

The function can be rewritten as the following constraints

$$v_i^\ell \geq \tilde{v}_i^\ell \tag{6}$$
$$v_i^\ell \leq \tilde{v}_i^\ell - M^{\text{low}}(1 - j_i) \tag{7}$$
$$v_i^\ell \leq M^{\text{up}} j_i \tag{8}$$

with $j_i \in \{0, 1\}$ a integer variable such that

$$j_i = \begin{cases} 0 & \text{if } \tilde{v}_i^\ell < 0 \\ 1 & \text{if } \tilde{v}_i^\ell > 0 \end{cases} \tag{9}$$

This decomposition allows for introducing a Neural Network of limited size into an optimization problem by decomposing it into a set of linear constraints of the form shown above. [3]

4.3 THE TWO TURBINE PROBLEM

Optimizing the positioning of two wind turbines can be expressed as optimizing the relative position of a second wind turbine T_2 to a fixed first turbine T_2, defined by the relative distances Δx and Δy. Both Δx and Δy are constrained by minimum distance Δ_{min} to T_2 and maximum distances $\Delta x_{max}/\Delta y_{max}$ to make the problem bounded. This problem can be visualized as shown in Figure 11.

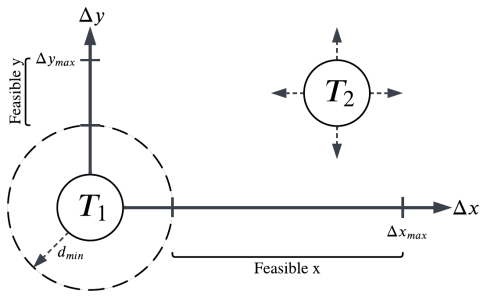

Figure 11: Optimizing the relative position $\Delta x/\Delta y$ of a second wind turbine T_2 to a fixed first turbine T_2, constrained by minimum distance d_{min} to T_1 and maximum distances $\Delta x_{max}/\Delta y_{max}$

The objective function to be optimized is the total power generation, e.g. the sum of power generated by both turbines. This objective is a function of both the position of the wind turbine as well as of the wind conditions like wind direction and wind speed.

$$f_{totalPower}(x, y, \text{windspeed}, \text{wind direction}, (...))$$

Different from the geographic coordinates, the wind condition parameters like windspeed are inherently not deterministic and follow distributions like the normal distribution as shown in Figure 12.

The two main challenges of the optimization of the two-turbine problem are thus:

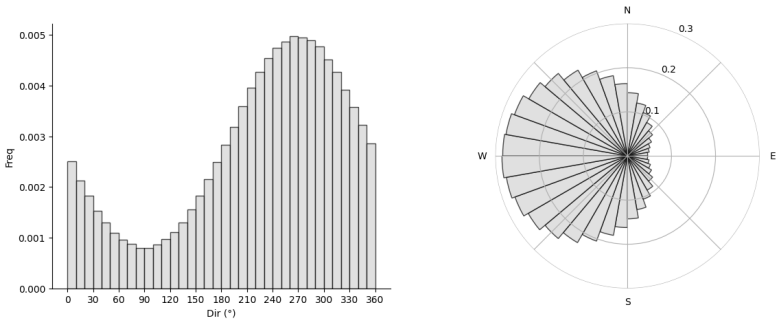

Figure 12: Histogram and Polar Plot across radiants for a normally distributed wind direction probability density function with mean West

1. Introduce the complex relationship between turbine position and wind conditions, and power output into the optimization problem

2. Introduce the non-deterministic nature of wind conditions into the optimization problem

The first of these challenges is tackled by applying the Neural Network models discussed in Section 3.2 and introducing them into the optimization problem via constraint learning as described in Section 4.2. For the second problem, multiple approaches are now explored in the following subsection.

4.3.1 *Deterministic Optimization*

To begin solving the problem, the simplest approach is to assume the wind conditions to be discrete. In application, this might be analogous to getting the Expectation of the joint probability distribution of all wind condition parameters. Taking these parameters as constant and homogeneous across the entire parameter space, the result is an objective function that is effectively only dependent on the relative positions of the turbine with the previously discussed geometrical constraints.

$$\max_{\mathbf{x},\mathbf{y}} f_{Power,\mathrm{NN}}(\Delta x, \Delta y) \tag{10}$$

$$\text{s.t.} \quad 0 \leq \Delta x \leq X_{\max} \tag{11}$$

$$0 \leq \Delta y \leq Y_{\max} \tag{12}$$

$$\sqrt{(\Delta x)^2 + (\Delta y)^2} \geq d_{\min} \tag{13}$$

where:

- $(\Delta x, \Delta y)$ are the relative distances of the two turbines
- $f_{Power,\mathrm{NN}}(\Delta x, \Delta y)$ is a neural network (deterministic) approximating the total power output

- $X_{\max}, Y_{\max}$ define the maximal distance the two turbines can be placed apart
- $d_{\min}$ is the minimum distance between the two turbines

Modelling

To begin with, the simulations to generate the dataset used to train the corresponding model for the deterministic case have to be generated. For the deterministic optimization, the position of the second turbine is variable in the bounds and an even steplength resulting from n_{Steps} as shown in Table 1. All remaining airflow characteristics remain constant (deterministic).

Table 1: Value Ranges for Deterministic Two Turbine Problem Data Set

Variable	**Const/Variable**	**Value**	Step Length
x_{turb2}	Variable	[0, 5000] m	50 m
y_{turb2}	Variable	[0, 500] m	50 m
wind_speed	Constant	8 m/s	-
wind_direction	Constant	270°	-
turbulence_intensity	Constant	0.06	-

In line with the parameter space defined in Table 1, simulations are performed as described in Section 3.3. The resulting dataset is then introduced into the Neural Network optimization method, yielding Figure 16. When considering the results, it is apparent that models with a number of layers greater or equal than two and greater or equal than 8 nodes per hidden layer are nearly identical in performance. The chosen model configuration is therefore a NN$(5 - 8^{\times 2} - 1)$, yielding good performance while limiting model size as much as possible, to be implemented in the optimization problem.

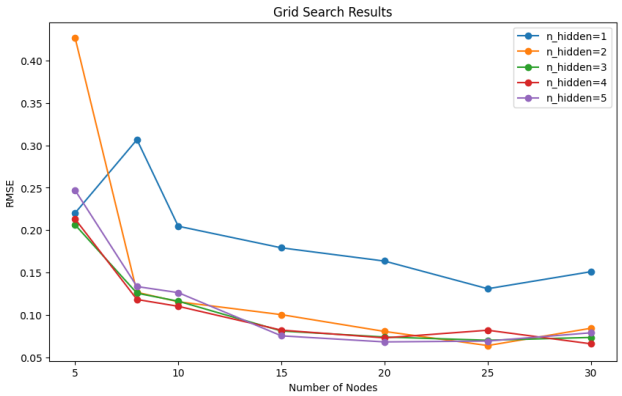

Figure 13: Grid search for Neural Network parameters for the deterministic model, showing how models with $n_{hidden} \geq 2$ and $n_{nodes} \geq 8$ yield near identical performance

To validate the chosen model and gain a deeper understanding of its predictive power, we proceed to visualize the total power output across the variable parameter space. Figure 14 shows a colormap generated from linealy interpolating between the datapoints for the simu-

lation data (Subplot 1), a finder grid of predictions of the model (Subplot 2) and a colormap of the percentage deviation between model and raw data (calculated at points and then again linearly interpolated). The color in the first two subplots represents the *total power generated of both wind turbines*, with the second turbine placed at the given x/y location We find that the model appears to overall represents the behaviour well, with only systematic deviation occuring at locations very close to the first wind turbine at the origin. As the simulation allows for placing wind turbines at the exact same location and with corresponding results unrealistic, simulations in this area are not necessarily correct in results either. For the optimization, the minimum distance constraint removes this area from the feasible region, meaning that the comparatively large deviations there do not affect the optimization.

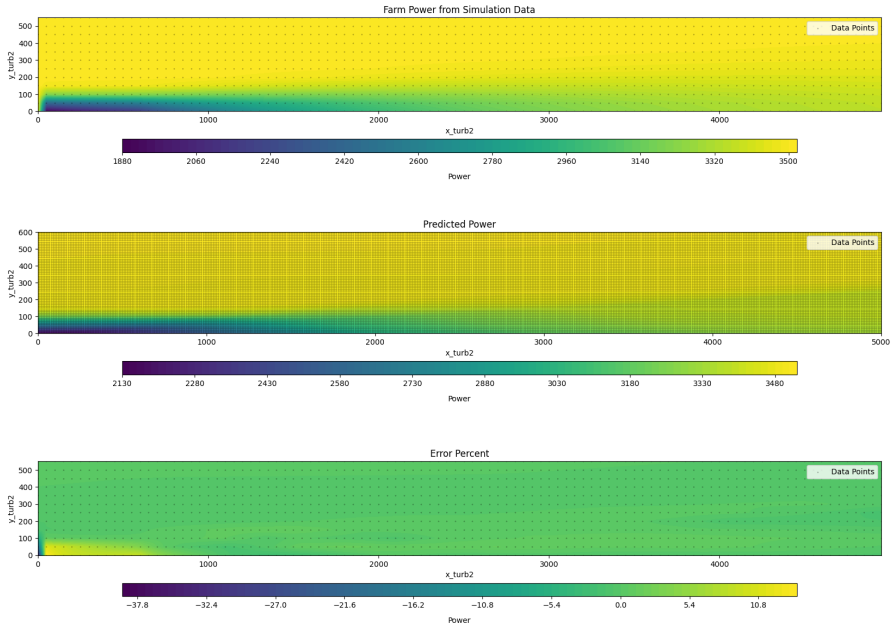

Figure 14: Colormap showing the Farm Power distribution for the second turbine being placed at any given x/y based on linear interpolation for real data, predicted data and the resulting percentage deviation, indicating overall good fit with only area of major deviation extremly close to turbine 1, an area excluded from the feasible region of the optimization problem

The model thus appears to be suitable to represent the interactions between the two turbines and we proceed to the optimization.

Optimization

With the model trained and validated, we proceed to implementing the defined optimization problem in Pyomo (Notebook Github) and embed the trained model using the approach described in Section 4.2.

When solving this problem as defined using Gurobi, a major challenge becomes apparent as the result shows that there is an large number of equally optimal solutions (e.g. degeneracy of the problem, see [30] for an exact definition of degeneracy) everywhere outside the

wake of turbine 1, as can be seen in the visualization of the optimization result found in 15.

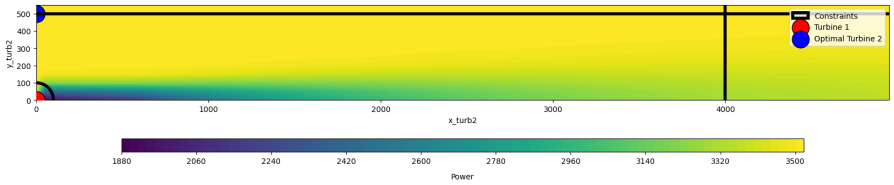

Figure 15: Deterministic optimization of the relative position of two wind turbines relative to each other with wind direction 270° and constant windspeed

The existence of a large number of optimal values thus has to be taken into account moving forward.

4.3.2 *Stochastic Optimization*

As the deterministic optimization does not take into account the distribution of wind condition parameters and having found that the deterministic approach leaves a range of infinite global optima outside of the wake of turbine 1, the next step is proceeding with the introduction of uncertainty in the form of probability distributions of wind condition parameters into the problem. To do so the objective function is modified to now represent the expected power generated by the farm, if turbine where to be placed at a given location. The Neural Network thus now has to be able to predict the power output for a given turbine 2 location taking into account the wind conditions like direction and speed at a given location. The problem is formulated as follows:

$$\max_{\mathbf{x},\mathbf{y}} \sum_{i=1}^{n} f_{Power,\mathrm{NN}}(\Delta x, \Delta y, \text{wind condition}) \cdot p_{n,\text{wind condtion combination}} \tag{14}$$

$$\text{s.t.} \quad \Delta x \leq X_{\max} \tag{15}$$

$$\Delta y \leq Y_{\max} \tag{16}$$

$$\sqrt{(\Delta x)^2 + (\Delta y)^2} \geq d_{\min} \tag{17}$$

where:

- $(\Delta x, \Delta y)$ are the relative distances of the two turbines,
- $f_{Power,\mathrm{NN}}(\Delta x, \Delta y)$ is a deterministic neural network approximating the total power output for turbine positions and wind conditions
- $X_{\max}, Y_{\max}$ define the maximal distance the two turbines can be placed apart
- $d_{\min}$ is the minimum distance between the two turbines

- n is the index of the discretized possible combinations of wind conditions

The following problem will first treat the univariate case of only wind direction being a random variable, with wind speed and turbolence intensity remaining constants.

Modelling

Like for the deterministic case, we begin by finding an fitting Neural Network model to solve that can be embedded into the optimization problem. To do so, we first again the parameter space, this time with wind direction bein variable and set up in a grid with steplength of 10°. The full parameter grid is defined in Table 2

Table 2: Value Ranges for Probabilistic Two Turbine Problem Data Set

Variable	**Constant/Variable**	**Value**	*Steplength*
x_{turb2}	Variable	[0, 5000] m	50 m
y_{turb2}	Variable	[0, 500] m	50 m
wind_speed	Constant	8 m/s	-
wind_direction	Constant	[180°, 270°]	10°
turbulence_intensity	Constant	0.06	-

We then proceed to hyperparameter tuning of the model. Due to the added complexity, the model size has to be increased significantly to yield similarly good results compared to the deterministic case. Due to limitations in compute power, this therefore yields a reduced number of possible comparisons.

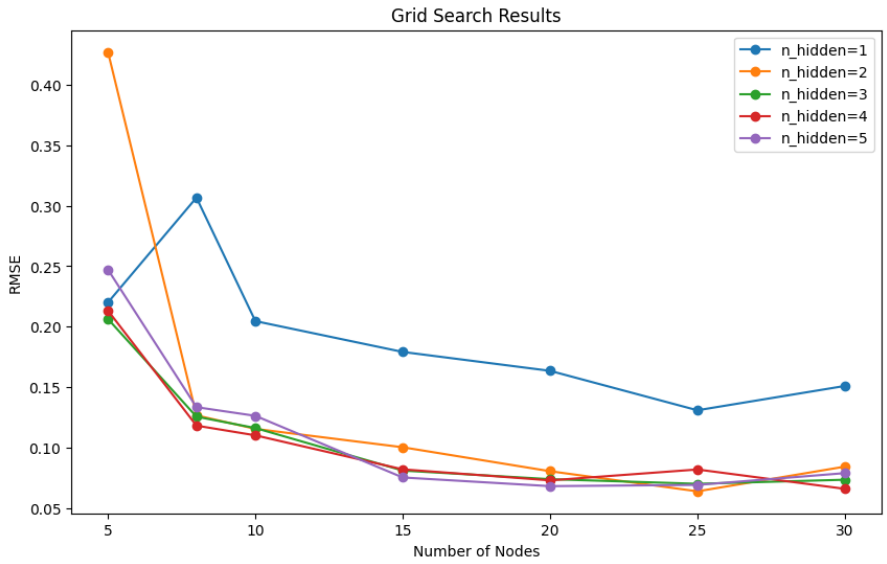

Figure 16: REPLACE FIGURE

With a model chosen, we again performe some investigation into its performance and find that even with the significantly more complex model, the results are not as good as for the model only trained for the deterministic case. While visually the shape of the wake generally

appears to fit, the deviation in percent shows are more farreaching deviation, even at greater distances away from Turbine 1, as can be seen in Figure 17 for wind direction 260° .

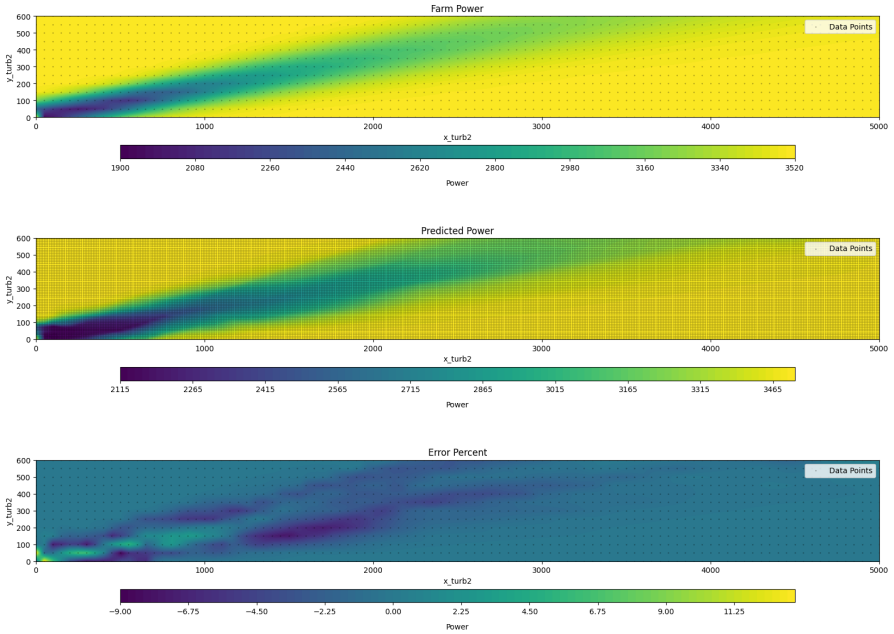

Figure 17: A heatmap of total generated farm power for a winddirection of 260° with datapoints and linear interpolation inbetween. Plot 1 shows the raw data, plot 2 the predictions of the NN$(5-25^{\times 10}-1)$, plot 3 the percentage difference between training points and predictions

Since the parameter space has now become three dimensional, conditioning on a single winddirection does not cover the full performance of the model. Figure **??** therefore shows the development of RMSE across different winddirection, indicating that the border regions and wind directions with an increased total wake lenght (e.g. wind directions closer to 270° wind direction) experience higher RMSE.

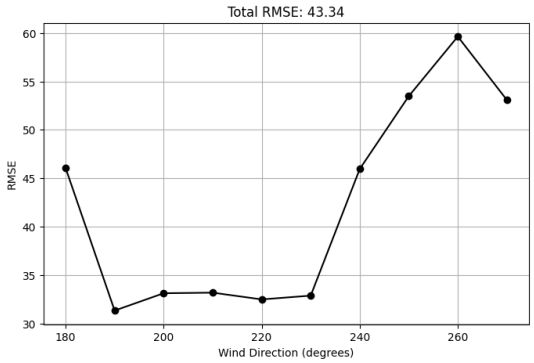

Figure 18: RMSE of NN$(5-25^{\times 10}-1)$ across the parameter space of wind direction

For optimization purposes, the individual power per wind direction however becomes less relevant, as the power outputs are weighted by their probability of occouring. To do so, we first have to define the

probability distribution of the wind directions. For simplicity sake, a Normal distribution with mean 270° and standard deviation of 5°, as shown in Figure 21.

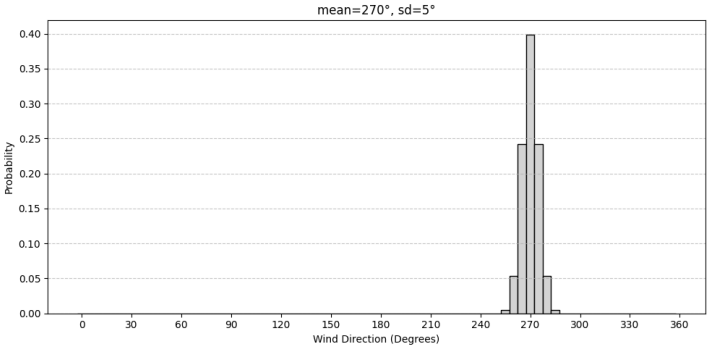

Figure 19: Discrete Wind Direction Probability Distribution with mean 270° and standard deviation 5°, discretized in intervals of 10°

Using this distribution, the expected power for the ranges of x and y coordinates can thus now be calculated by aggregating the powers for the different wind directions as a sum weighted by probability to yield the expected farm power generated at the specific coordinate. By evaluating the expected farm power both for the training data as well as for predictions for the training data, we can again compare the results and deviations as done in Figure **??**

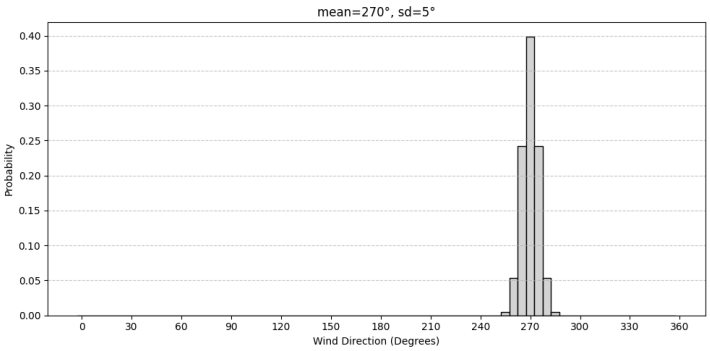

Figure 20: REPLACE FIGURE

While increasing the size of the Neural Network would undoubtetly reduce the deviations across wind directions and for the individual wind directions, the size of the Neural Network is restricted by the solvability of the optimization problem. While there is no exact threshold, solving time of the mixed integer problem that results from the embedding of the Neural Network increases roughly exponentially by adding Neurons to the Network and therefore adding binary variables to the problem. The shown Neural Network was thus found to be the maximum feasible size of Neural Network for the given computational recourses.

The same way the distribution of farm power was previously dependent on the wind direction, it is now dependent on the wind direction distribution and therefore in the case of the Normal, on its mean and variance. Changing the mean of the distribution cor-

responds to changing the direction of the principal wind direction and thus the direction of the highest expected wake losses from turbine 1. Meanwhile, increasing the variance corresponds to spreading the expected wake losses due to turbine 1 across a wider range of wind directions and therefore increasing the areaaffected by expected wake losses, while reducing the absolute reduction in expected power within the area affected by the wake. This relationship can be seen in the expected power distribution for a normal with mean MEAN and standard deviation STDEVIATION shown in Figure **??**

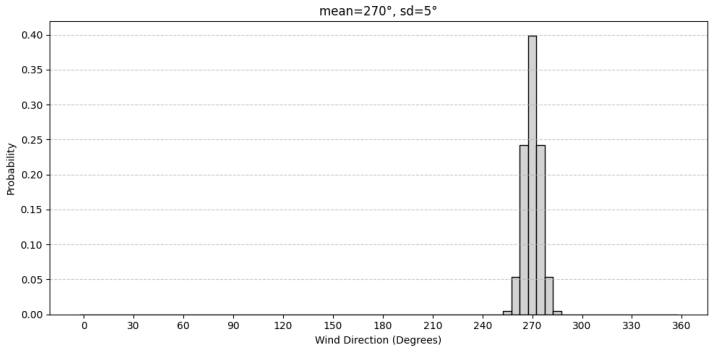

Figure 21: REPLACE FIGURE

The normal distribution is used in this thesis as a placeholder for proof of concept purposes. In practice, in practical application, the real wind distribution would have to be evaluated at the specific location.

TODO: Improvements -> train NN slightly beyondactual parameter space

Optimization

With the model set up, solving the stochastich optimization problem for a maximial expected farm power output can be attempted

The Neural Network and the wind direction distribution is again introduced into pyomo to find a global maximum. Like before, plotting the result as shown in 23, it is possible to visually find that there is a large range of more or less equally optimal points in the wind directions that the wind distribution discretization does not cover the space evenly as the parameter space for x and y of turbine 2 are not identical. The discretization in the shown scenarios thus does not sufficiently cover the whole space, leading to seemingly optimal areas where in reality significant wake losses can be expected.

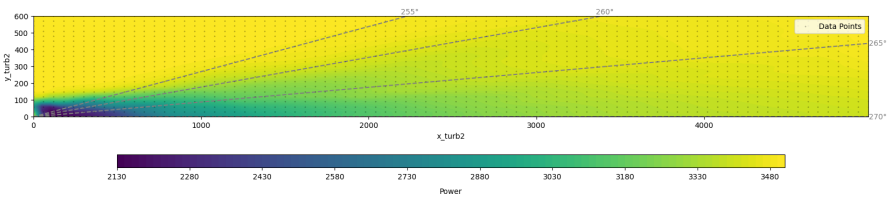

Figure 22

In practice, the size of the Neural Network required to represent the farm power function makes the identification of the global optimum challanging due to the number of integer variables added to the problem. Adding to this is the large range of nearly equal optimal solutions, leading to solvers like gurobi to require extremely large runtimes. The development of best optimal and upper limit, showing how the maximum of the objective function remains nearly identical, the upper limit to the solution converges towards the first found optima. This behaviour can bee seend in Figure **??**

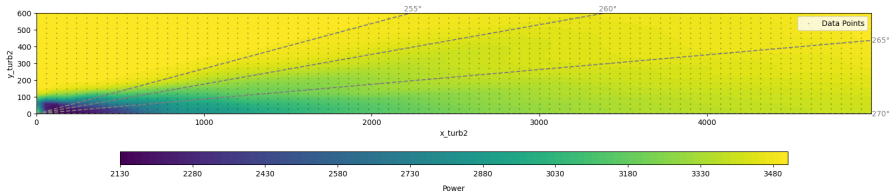

Figure 23

TODO : PLAY WITH CONSTRAINTS

A discussion of the value of this approach will be performed in the conclusion of this thesis.

Stochastic Optimization: Quantile Discretization of Wind Direction Distribution

One thing noticable in the previously shown plots of the distribution of theexpected farm power is how it is affected by the discretization of the wind directions, previously done in 10° steps. This does however mean that in the directions for which a long wake is considered (x direction/ 270°) the space in between the space between the scenarios is fairly large, as very apparent by the degree lines shwon in the solution to the probabilistic case (Figure **??**). A way to take this into account is by discretizing not with an constant step length but based on the quantiles of the distribution. The discretization can thus be performed not by a constant length of the disrcetization steps, but a constant probability within a chosen number of quantiles. For each quantile individually the mean is then calculated and the resulting expected values for the quantile are taken as discretization steps. Such an approach used for the same Normal distribution with mean 270° and standard deviation 5° degrees, fixing the scenario/quantile count to 7, yields the discretization shown in Figure 24.

Using this distribution, the expected power distribution changes for the corresponding scenarios/quantiles, with the the expected wake losses moving towards the true high-probability zone for wake losses. The optimal area thus moves out of the spaces inbetween scenarios to the true zone of low expected wake losses, as can be seen in Figure **??**

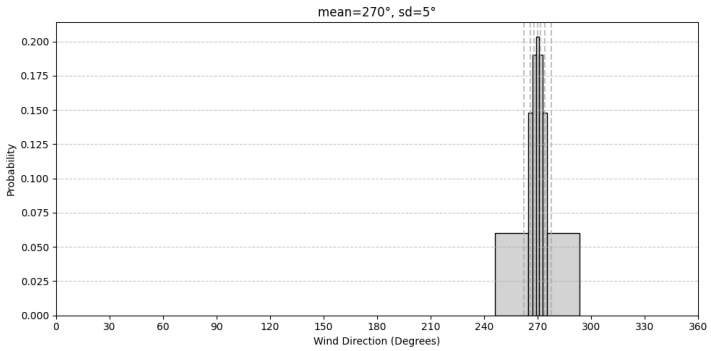

Figure 24: Quantile based discretization of Normal mean 270°, standard deviation 5°, disrcetized into 7 scenarios with equal probability for scenario and the scenario angle corresponding to the expected value for the quantile

Stochastic Optimization: Unsymmetric Wind Direction Distributions

4.4 THE THREE TURBINE PROBLEM

4.5 THE N TURBINE PROBLEM

Thoughts:

IDEA: generalize inputs in a way that allow for power calculation of each turbine individually instead of farm power

IDEA : derive basic rules from 2 turbine problem and set up new optimization problem without NN

IDEA : analogue to forward/backward selection with 2 turbine neural network

FINIDINGS:

problem degenerate if not one wind turbine fixed

4.6 POSSIBLE IMPROVEMENTS IN OPTIMIZATION

The main limitation of the approach as presented in this chapter is that the increasing the number of neurons in the Neural Network by one neuron increases the number of integer variables by one and the number of constraints by three, while adding an additional scenario requires reintroducing the entire Neural Network for that specific scenario. As computation time is a function of the number of constraints and decision variables, the computation time increases approximately exponentially with adding neurons or scenarios.

Finding a way to reduce the number of constraints/integer variables would thus be the main concern when attempting to improve the presented method.

5 COMPARISON OF METHODS AND CONCLUSION

The thesis contains the training of a Neural Network to predict the total farm power generated by a wind farm consisting of two turbines, with the inputs to the model as the relative position of the second wind turbine to the first as well as the wind direction. The resulting Neural Networks where embedded into a mixed integer linear programming problem using a novel approach for decomposing a Neural Network into linear constraints containing intenger variables and using a big M approach. Multiple optimization problems where solved using this aproach, beginning with a deterministic problem by only considering a dominant wind direction to find maxima for the total farm power and then proceeding to a stochastic scenario based optimization problem in which the expectd power generation is maximized, subject to a discreitzed distribution of the wind direction.

TODO: COMPARISON

The main challange of the work, beyond the correct implementation, is the rapid increase in the size of the optimization problem, for both an increase in Neural Network size and additional numbers of scenarios in the probabilistic case. These limitations prevented the Neural Network to be set up with completely satisfactory results and further constrained the possible number of scenarios that could be taken into account.

Meanwhile, the used was in general terms successfull in introducing a highly nonlinear relationship represented by the Neural Network into a mixed integer linear programming problem. The challenges found in doing so however appear to limit the scalability beyond the 2 turbine setup as a further increase in Neural Network size would be required if a third turbine in the form of its relative position would again increase the parameter space significantly.

Overall, the method used in this work appears was shown to be able cope with highly nonlinear relationships, but in its current form appeared to reach its limits in the problem at hand.

Among others, three potential pathways could be chosen to proceed from this point on and build on the work done in this thesis:

1. The probabilistic component of the problem could be moved from outside the model as the currently used scenario based approach could be replaced by changing the output of the Neural network to be the expected farm power. This would require applying the wind distribution to the raw simulation data first and thus remove the models independence of the wind distribution at a given location, but instead prevent the model from having to be embedded multiple times for scenarios and reduce the size of the required Neural Network as size of the parameter

2. Use the scenario based approach and attempt to replace the complex Neural Network by a simpler model type like an analytical or functional approximation

In any case, maximizing the expected farm power across the given wind direction distribution and, once computational constraints are resolved, using multivariated distribution of the wind conditions by adding potentiall interdependent random variables like windspeed seem feasible and significantly superior compared to only relying on a dominant wind direction.

ABBREVIATIONS

Abbreviation	Meaning
AEP	Annual Energy Production
MSE	Mean Squared Error
ReLu	Rectified Linear Unit
IEA	International Energy Agency
MW	Mega Watt

LIST OF FIGURES

BIBLIOGRAPHY

[1] European Environment Agency. Europe's onshore and offshore wind energy potential. Technical Report Technical report 6/2009, European Environment Agency, 2009. Accessed: 2025-04-03.

[2] Inimfon Akpabio and Serap Savari. On the construction of distribution-free prediction intervals for an image regression problem in semiconductor manufacturing. 03 2022.

[3] Antonio Alcántara and Carlos Ruiz. A neural network-based distributional constraint learning methodology for mixed-integer stochastic optimization. *Expert Systems with Applications*, 232:120895, 2023.

[4] Antonio Alcántara, Carlos Ruiz, and Calvin Tsay. A quantile neural network framework for two-stage stochastic optimization. *Expert Systems with Applications*, page 127876, 2025.

[5] Analysis and General Secretariat of the Council of the European Union Research Team. Harnessing wind power: Navigating the eu energy transition and its challenges. https://www.consilium.europa.eu/media/1kyk0wjm/2024_685_art_windpower_web.pdf, September 2024. Accessed: 2025-04-03.

[6] F. Azlan, J.C. Kurnia, B.T. Tan, and M.-Z. Ismadi. Review on optimisation methods of wind farm array under three classical wind condition problems. *Renewable and Sustainable Energy Reviews*, 135:110047, 2021.

[7] N. Bempedelis, F. Gori, A. Wynn, S. Laizet, and L. Magri. Data-driven optimisation of wind farm layout and wake steering with large-eddy simulations. *Wind Energy Science*, 9(4):869–882, 2024.

[8] John R. Birge and François Louveaux. *Introduction to Stochastic Programming*. Springer Series in Operations Research and Financial Engineering. Springer, July 1997.

[9] Alessio Bonfietti, Michele Lombardi, and Michela Milano. Embedding decision trees and random forests in constraint programming. 05 2015.

[10] Pietro Bortolotti, Helena Canet Tarrés, Katherine Dykes, Karl Merz, Latha Sethuraman, David Verelst, and Frederik Zahle. Iea wind task 37 on systems engineering in wind energy – wp2.1 reference wind turbines. Technical report, National Renewable Energy Laboratory, 2019.

[11] European Commission. Renewable energy targets. https://energy.ec.europa.eu/topics/renewable-energy/

renewable-energy-directive-targets-and-rules/renewable-energy-targets_en, 2023. Accessed: 2025-04-03.

[12] Héctor G. de Alba, Andres Tellez, Cipriano Santos, and Emmanuel Gómez. A reformulation to embedding a neural network in a linear program without integer variables, 2024.

[13] Luc De Raedt, Andrea Passerini, and Stefano Teso. Learning constraints from examples. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[14] Adejuyigbe O. Fajemisin, Donato Maragno, and Dick den Hertog. Optimization with constraint learning: A framework and survey. *European Journal of Operational Research*, 314(1):1–14, 2024.

[15] Xavier Glorot, Antoine Bordes, and Y. Bengio. Deep sparse rectifier neural networks. volume 15, 01 2010.

[16] Peng Hou, Jiangsheng Zhu, Kuichao MA, Guangya YANG, Weihao HU, and Zhe CHEN. A review of offshore wind farm layout optimization and electrical system design methods. *Journal of Modern Power Systems and Clean Energy*, 7(5):975–986, September 2019.

[17] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. *An Introduction to Statistical Learning: with Applications in Python*. Springer Texts in Statistics. Springer, 2023.

[18] Samuel Kainz, Julian Quick, Mauricio Souza de Alencar, Sebastian Sanchez Perez Moreno, Katherine Dykes, Christopher Bay, Michiel Zaaijer, and Pietro Bortolotti. Iea wind tcp task 55: The iea wind 740-10-mw reference offshore wind plants. Technical Report NREL/TP-5000-87923, National Renewable Energy Laboratory, 2024. Technical Report.

[19] Taewan Kim, Jeonghwan Song, and Donghyun You. Optimization of a wind farm layout to mitigate the wind power intermittency. *Applied Energy*, 367:123383, 2024.

[20] C.T. Kiranoudis and Z.B. Maroulis. Effective short-cut modelling of wind park efficiency. *Renewable Energy*, 11(4):439–457, 1997.

[21] Mads Madsen, Frederik Zahle, Sergio Gonzalez Horcas, Thanasis Barlas, and Niels Sørensen. Cfd-based curved tip shape design for wind turbine blades. *Wind Energy Science*, 7:1471–1501, 07 2022.

[22] M. Magnusson and A.-S. Smedman. Air flow behind wind turbines. *Journal of Wind Engineering and Industrial Aerodynamics*, 80(1):169–189, 1999.

[23] Grzegorz Marcjasz, Michał Narajewski, Rafał Weron, and Florian Ziel. Distributional neural networks for electricity price forecasting. *Energy Economics*, 125:106843, September 2023.

[24] Michael Nielsen. Neural networks and deep learning - chapter 1, 2015. Accessed: 2025-04-10.

[25] Michael Nielsen. Neural networks and deep learning - chapter 2: How the backpropagation algorithm works, 2015. Accessed: 2025-05-01.

[26] Michael Sinner and Paul Fleming. Robust wind farm layout optimization. *Journal of Physics: Conference Series*, 2767(3):032036, jun 2024.

[27] Ingo Steinwart and Andreas Christmann. Estimating conditional quantiles with the help of the pinball loss. *Bernoulli*, 17(1), February 2011.

[28] Zilong Ti, Xiao Wei Deng, and Hongxing Yang. Wake modeling of wind turbines using machine learning. *Applied Energy*, 257:114025, 2020.

[29] Zilong Ti, Xiao Wei Deng, and Mingming Zhang. Artificial neural networks based wake model for power prediction of wind farm. *Renewable Energy*, 172:618–631, 2021.

[30] Robert J. Vanderbei. *Linear Programming*, chapter 3. Springer, New York, NY, 5 edition, 2020.

[31] Li Wang, Mi Dong, Jian Yang, Lei Wang, Sifan Chen, Neven Duić, Young Hoon Joo, and Dongran Song. Wind turbine wakes modeling and applications: Past, present, and future. *Ocean Engineering*, 309:118508, 2024.

[32] Windpower Monthly. Offshore wind clusters to lower energy production – study. *Windpower Monthly*, 2019. Accessed: 2025-04-29.

[33] Kun Yang, Xiaowei Deng, Zilong Ti, Shanghui Yang, Senbin Huang, and Yuhang Wang. A data-driven layout optimization framework of large-scale wind farms based on machine learning. *Renewable Energy*, 218:119240, 2023.