uc3m | Universidad Carlos III de Madrid

Master in Statistics for Data Science
2024-2025

Master Thesis

"Mixed-Integer Stochastic Optimization of the Relative Position of Two Wind Turbines Using Neural Network-Based Constraint Learning"

Simon Schmetz

Carlos Ruiz Mora

Madrid the 21. of June 2025

ABSTRACT

This following Thesis combines mathematical Optimization with Constraint Learning to optimize the relative dicstance between two wind turbines in a predefined area under randomly distributed wind conditions to maximize the total power output. A Neural Network is trained on simulated data to model the impact of turbine positioning on power output. This model is embedded as a set of constraints in a mixed-integer 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 [1], wind energy is playing a central role in that transition and expected to rise to 50 % in the EU energy mix [2]. With wind energy thus expected to become one of the main contributors to the EU's energy production and large potentials identified for both onshore and offshore parks [3], attempts to optimize all parameters of windparks that result in even minor power efficiency improvement can be expected to yield significant returns in absolute power due to the scale of future wind energy production.

Within the main wind energy challenges lies the problem 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 the power output [4, 5].

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

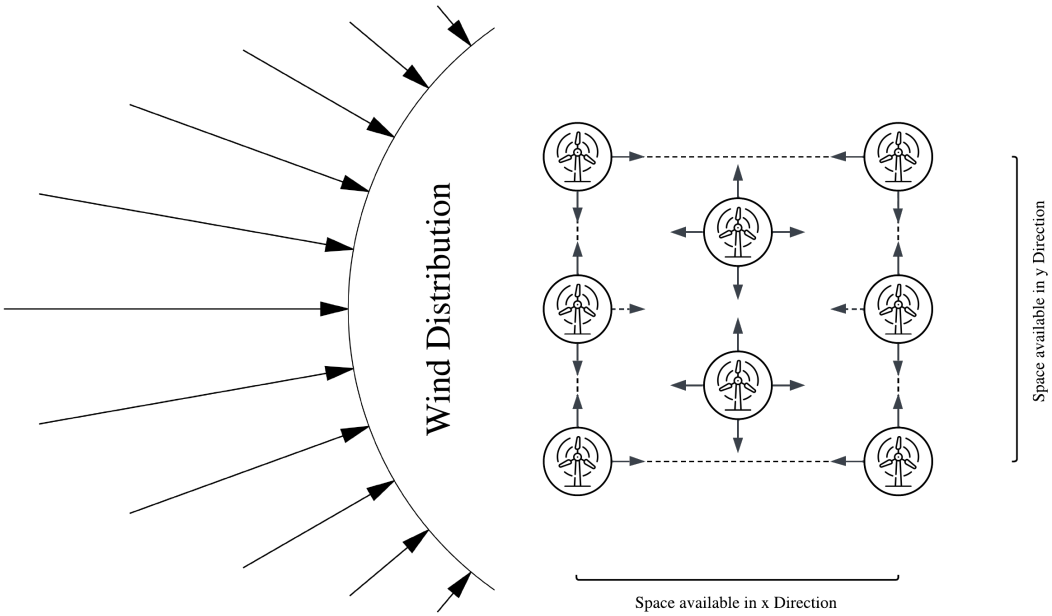

Figure 1: Optimizing the total power output of a farm reduces to placing wind turbines within a set space for the farm, subject to the wind conditions at the given location

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:

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

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

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

$$\tag{4}$$

where:

- (x_i, y_i) are the positions of the turbines
- $f_{Power,\text{NN}}(\cdot)$ is a neural network approximating the farm power output, e.g. the sum of power generated by all individual turbines
- $X_{\max}, Y_{\max}$ define the maximal extension of the space in which the turbines can be placed
- *wind conditions* are the the random variables that represent the wind conditions at a specific location, like wind direction and wind speed with their respective probability distributions

This thesis is dedicated to a new approach for optimizing the placement of a fixed number of wind turbines in a predefined space, beginning with the two-turbine problem, 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 embedding of Neural Networks into the optimization problem as a set of constraints [6]. This extension allows the modelling of the effects of wind turbine placement relative to each other on power production. Introducing this model into the problem then allows for the optimization of overall power production across all wind turbines in the wind park.

To create a model that optimally fits to the needs of the optimization problem, the model is trained on data specifically generated with the FLORIS [7] wind farm simulation tool for optimal coverage of the optimizations parameter space. 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 separated 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 in wind turbine placement optimization and constraint learning 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 maximize the total power output is the subject of scientific investigation. As this Thesis is an attempt to apply a novel constraint learning method introduced in [6] to the optimization of wind farm layouts, the following state-of-the-art is split into two pieces: The first investigates the current relevant publications of wind farm optimization and the second one presents a brief introduction into recent developments 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 wind farm layouts is to reduce the negative impact of wake effects between wind turbines [5]. Historically, rule of thumb approaches were used by setting up the layout as a grid with the distance between wind turbines, with spacing 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 [8, 4].

These methods have evolved to pursuing the goal of maximizing the Annual Energy Production (AEP) of wind farms in the context of stochastic optimization as done in [9, 5].

The core of any of the most recent optimization models is a wake model, which becomes part of the objective function to represent the wake effects on power output. These models can be categorized as [10]:

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 the Gaussian wake model and the 3D wake model [10]. With advancements in Machine Learning, the field of data-driven modeling is meanwhile expanding, with successful attempts in 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 [11, 12, 13, 14].

Constraint Learning

The term Constraint Learning, defined as "finding a set of constraints, a constraint theory, that satisfies a given dataset" in [15], 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 in Machine Learning the models learn from a given data set, the constraints resulting from such a model are equally learned. [15]

For this thesis specifically, we consider the uncertainty aware constraint learning method of decomposing a neural network into a set of mixed-integer linear constraints [6, 16]. Similar approaches of embedding Machine Learning models into optimization problems have been taken for Decision Trees and Random Forests in [17] or for Neural Networks (without integer variables required), as done in [18]. A survey performed by Fajemisin et al. [19] 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, the goal is to use the distCL extension [20] to the Pyomo Python package, using a small Neural Network as a surrogate model. The following chapter documents the steps taken to generate such a model. To train 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 [7] was used to generate a dataset. Then, the fundamentals of Neural Network architecture and training are briefly introduced, before the pipeline that yields the final model is presented.

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 wind speed joined with the increased turbulence in the wake airflow (see Figure 2) [21]. 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 [23]

With the overall goal of creating a model on 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 way to ensure that the data fits the use

case well. After investigating two Python-based open source wind farm simulation tools FLORIS (FLORIS official website [24]) and PyWake (PyWake Documentation [25]), FLORIS was chosen due to its apparent ease of use. FLORIS is a wind farm simulation tool developed by the National Renewable Energy Laboratory (NREL) as one of multiple tools to run simulations of varying complexity. FLORIS is a control-systems-oriented model 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, FLORIS is given a grid of the following parameters which are dependent on the specific optimization problem:

- Position of Wind Turbines
- Wind Direction
- Wind Speed
- Wind Turbulence

We define 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 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 [26] and [27] and a more exact technical specification of the turbine can be found in [28]. 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 embedded into the Pyomo extension as explaind 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, as large models increase the number of constraints required to decompose the model.

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 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 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 of the Neural Network. 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 as 9 layers with the output value of each of those last perceptrons 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.

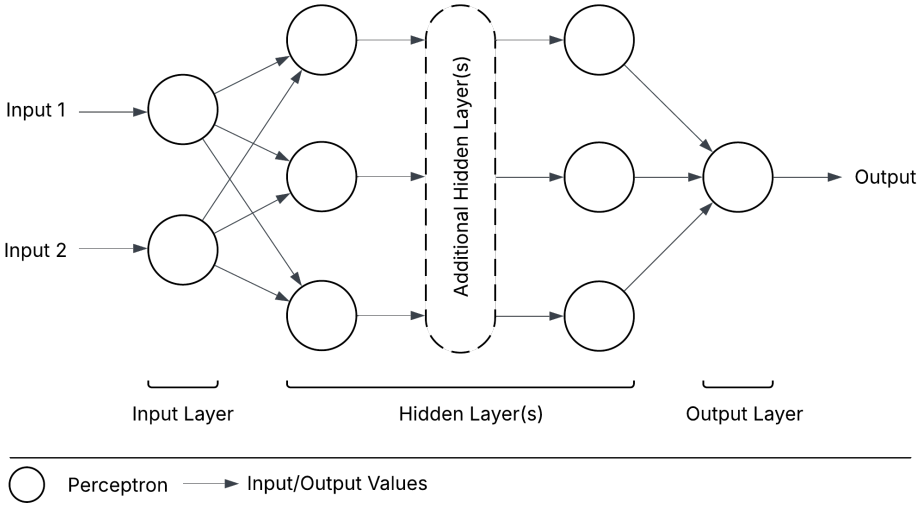

Figure 3: Schematic of Neural Network Architecture

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 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. 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 thresholds. The most common function used in place 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 $x \to -\infty$.

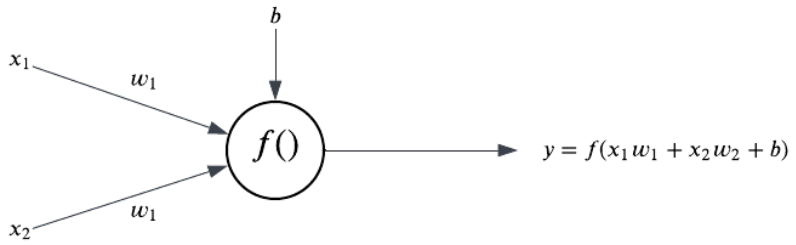

Figure 4: The output of a neuron is generated by applying the activation function to the sum as the weighted inputs $w_i x_i$ and the bias of the neuron b

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

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

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

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 [29, 30]. Beyond this, the ReLu function allows for embedding the Neural Network into a optimization problem, as will be discussed in Section 4.2.

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 quantifies a relationship between the prediction of the model and the correct output (defined by training data) and gives a Loss as output, which is a sort of difference measure between model prediction and truth. Whether a model is performing well in a specific case is heavily dependent 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*. [30]

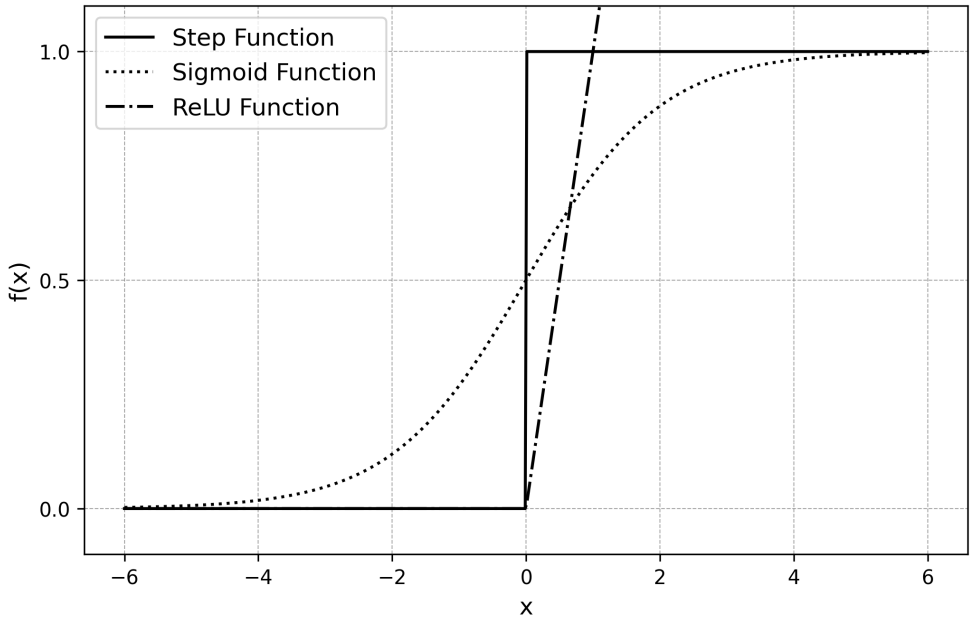

Figure 5: The Sigmoid Function being close to the Step Function without being as sensitive to slight changes in x due to its continuity. Alternatively, the Rectified Linear Unit function (ReLu) can be used as activation function.

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 can be seen for both the weights and bias in an example in Figure 6.

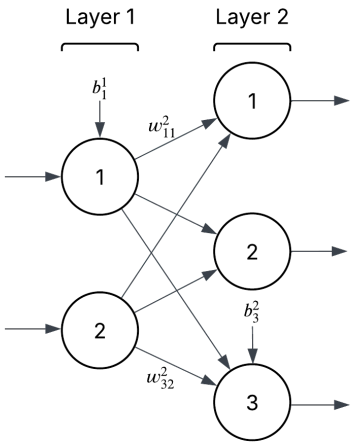

Figure 6: Notation of Neural Network weights and biases with weight w^l_{jk} and l being the input layer, j the neuron from that input layer and k the neuron of the $(l-1)^{th}$ output layer. Similarly, the bias as b^l_j with l as the layer and j the neuron.

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

$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right),$$

This expression can be further simplified by writing weights and biases in matrix form, using 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. This is done by using training data with a known true value for the prediction variable 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 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 ($\frac{\Delta Loss}{\Delta w_{jk}^l}$, $\frac{\Delta Loss}{\Delta b_j^l}$) using backpropagation [1]
3. Update the weights and biases according to a learning rate ρ weighted by the specific weight or biases gradient like $w_{jk}^l = w_{jk}^l - \rho\frac{\Delta Loss}{\Delta w_{jk}^l}$ and $b_j^l = b_j^l - \rho\frac{\Delta Loss}{\Delta b_j^l}$
4. Repeat process n Epochs, with one Epoch being the algorithm passing through all training points once

This algorithm represents a gradient descent algorithm with learning rate ρ corresponding to the step length while evaluating the partial derivatives $\frac{\Delta Loss}{\Delta w_{jk}^l}$, $\frac{\Delta Loss}{\Delta b_j^l}$ 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 [31]. 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 reasonable to hoped to find a good local minimum as solution [32, 31].

Regularization

Like most Machine Learning models, Neural Networks run at risk of overfitting. They are especially prone to overfitting due to their many degrees of freedom, represented by the many biases and weights, at times 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

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

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. [32]

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 objective is a result of each neuron corresponding 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 spaces. 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, and the final model training as seen in Figure 7. In this case, 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 (again).

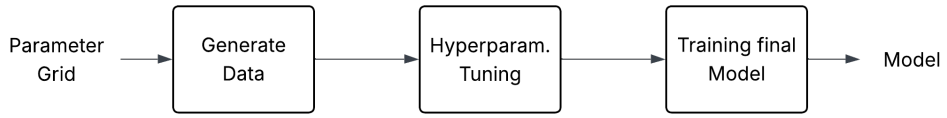

Figure 7: Flow of model generation, with parameter grid as input and the final model as output

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 [24], 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 north, 180° to south) and turbulence intensities as a measure of turbulence for the incoming airflow. Figure 8 shows an example of a simulation output.

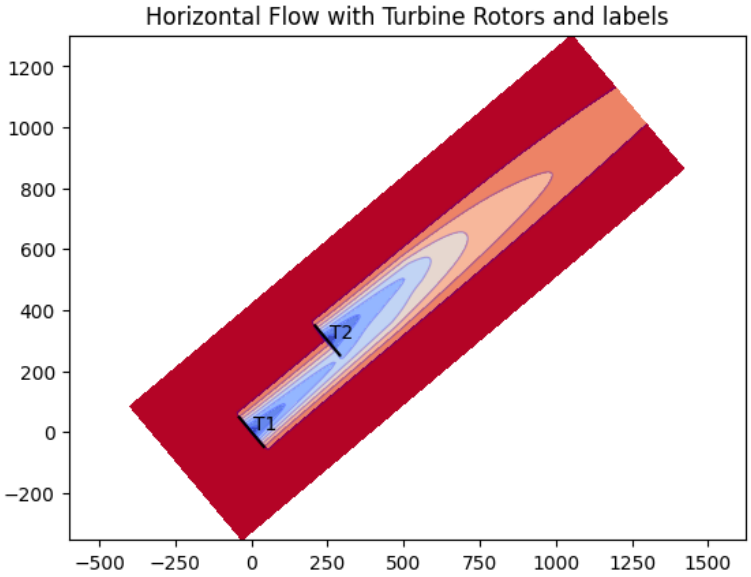

Figure 8: Example of a Simulation output from the FLORIS Wind Turbine Simulation Tool [7] with colormap corresponding to airflow velocity

The method implemented in (Data Generation Notebook [33]) 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 is constant for each parameter individually, e.g. the distance between values is the same for the entire range. Depending on the configuration of the problem, some parameters themselves 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:

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

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. First, an appropriate loss function has to be chosen. The data is then split into a training and a test 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 Root Mean Squared Error (RMSE).

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 9 generated to decide what the best configuration of the Neural Network is.

Figure 9: Exemplary Output of Hyperparameter tuning with number of hidden layers n_{hidden} and number of nodes per hidden layer

The final decision on what the ideal model configuration is not only based on the best performance, but via the trade-off between complexity and achieved Loss, to generate a model that is as small as possible.

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 one of 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 volatile 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 to 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 theoretical and pratical foundations of the model of farm/turbine power established, the following section treats the embedding of the trained model(s) into a deterministic/stochastic optimization problem. The underlying theory of optimization under uncertainty and constraint learning is introduced before the actual optimization problems for the two turbines 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. For example, 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 represent uncertainty, 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, the 10% demand increases a 20% probability and the 10% decrease a 30% probability, the problem can be modified to maximize the expected profit across these three scenarios by the formulation

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

Assuming only these three scenarios are possible, 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

1 x of course non-negative

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. [34]

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 a set of constraints. As many real-life relationships are difficult to represent by a simple analytical function, introducing Machine Learning models to optimization problems opens up many new possibilities [19].

In the case of Neural Networks, one way of introducing a Neural Network as a constraint into an optimization problem is by recognizing that when using the Rectifier Linear Unit (ReLu) Function as 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 $j_i \in \{0, 1\}$ an 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 into an optimization problem by decomposing it into the shown set of mixed-integer linear constraints [6].

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_1, 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 10.

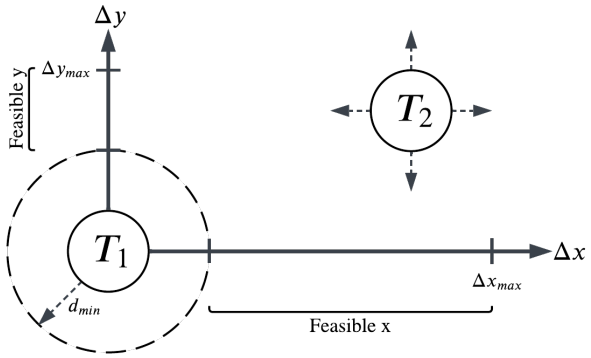

Figure 10: Optimizing the relative position $\Delta x/\Delta y$ of a second wind turbine T_2 to a fixed first turbine T_2, constrained by minimum distance d_{min} to T_1 and maximum distances $\Delta x_{max}/\Delta y_{max}$

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 a probability model like the normal distribution as shown in Figure 11.

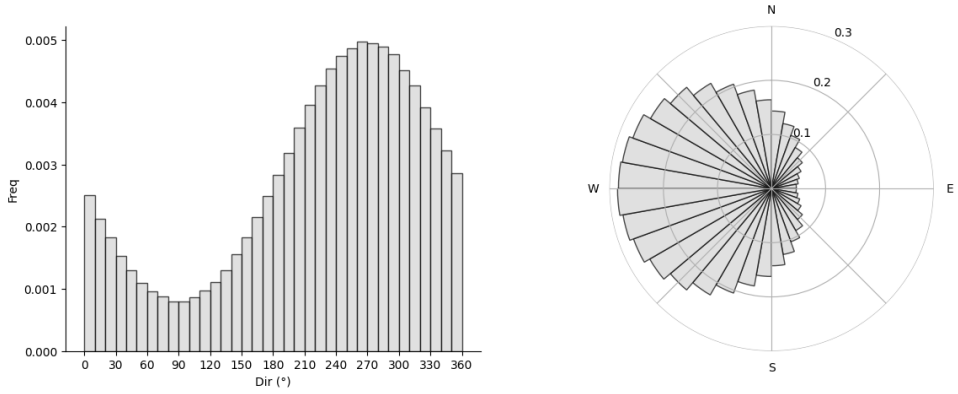

Figure 11: Histogram and Polar Plot across radiants for a normally distributed wind direction probability density function with mean West

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

1. Introduce the complex relationship between turbine position, 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 **??** 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 subsections.

In the following formulations, Pytorch [35] is used to set up the Neural Network Architecture and train the models, the Python Pyomo Package [36] used to formulate the underlying optimization problems, a modified version of the Constraint Learning algorithm developed in [37] used to embed the Neural Network into the optimization Problemthe Python packages and Gurobi [38] is used to solve the optimization Problems.

4.3.1 *Deterministic Optimization*

To begin solving the problem, the simplest approach is to assume the wind conditions to be deterministic, e.g to assume the wind comes only out of one direction. In application, this might be analogous to using the Expectations of the joint probability distribution of all wind condition parameters, which in practical terms would be considered the "principal wind direction". 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 farm 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

The Notebook belonging to this formulation can be found in Deterministic Formulation Notebook [39]

Modeling

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 given bounds and with an even step length 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
Δx_{turb2}	Variable	[0, 5000] m	50 m
Δy_{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 12. 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 reducing model size as much as possible, to be implemented in the optimization problem.

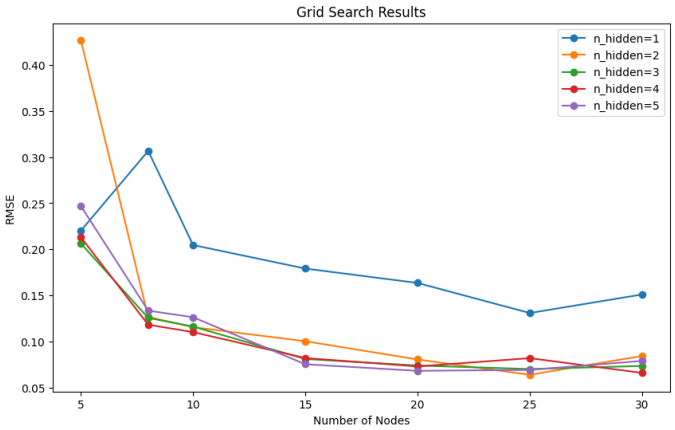

Figure 12: Grid search for Neural Network parameters for the deterministic model, showing how models with $n_{hidden} \geq 2$ and $n_{nodes} \geq 8$ yield near identical performance

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 13 shows a colormap generated from linearly interpolating between the datapoints for the simulation data (Subplot 1), a finer grid of predictions of the model (Subplot 2) and a colormap of the percentage deviation between model prediction and true simulation values (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 occurring 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.

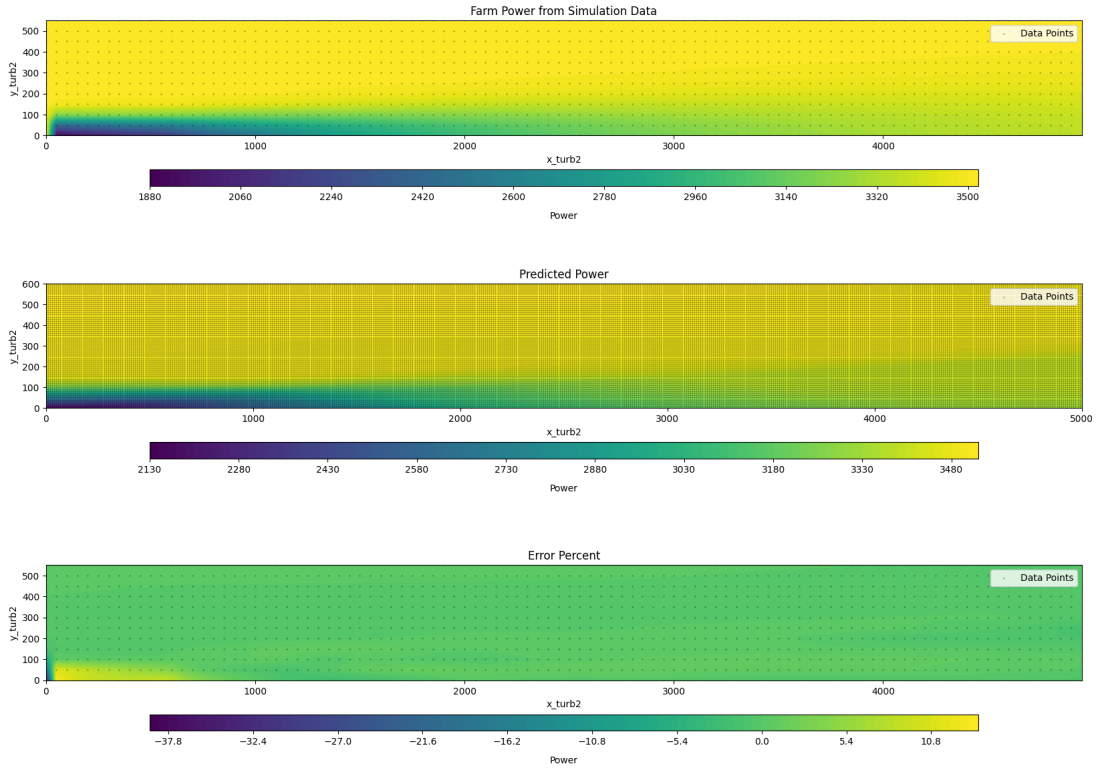

Figure 13: Colormap showing the Farm Power distribution for the second turbine being placed at any given x/y based on linear interpolation for real data, predicted data and the resulting percentage deviation, indicating overall good fit with only area of major deviation extremely close to turbine 1, 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 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 a large number of equally optimal solutions (e.g. degeneracy of the problem, see [40] 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 Figure 14.

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

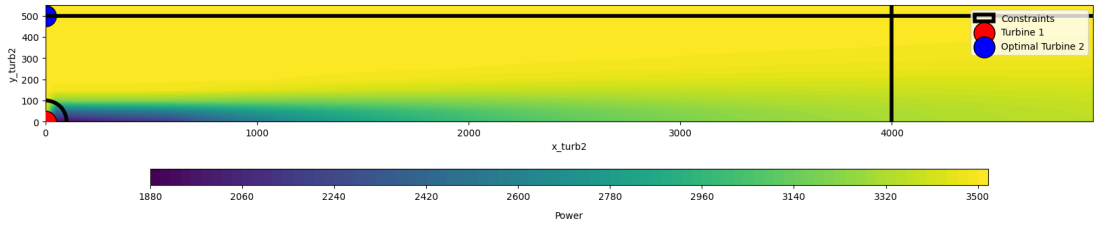

Figure 14: Deterministic optimization of the relative position of two wind turbines relative to each other with wind direction 270° and constant windspeed

4.3.2 *Stochastic Optimization: Wind distribution independent Neural Network*

As the deterministic optimization does not account for the distribution of wind condition parameters and has found that the deterministic approach leaves a range of infinite global optima outside of the wake of turbine 1, the next step is to proceed with the introduction of uncertainty of the wind condition parameters into the problem. To do so, the objective function is modified to represent the expected power generated by the farm, if a turbine were 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 condition parameters like direction and speed at a given location. The problem is formulated as follows:

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

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

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

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

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
- $p_{n,\text{wind conditions}}$ is the probability corresponding to each of the scenarios
- n is the index of the discretized possible combinations of wind conditions

The following section will first treat the univariate case of only wind direction being random, while wind speed and turbulence intensity remain constants.

The Notebook belonging to this formulation can be found in Stochastic optimization with farm power NN Formulation Notebook [41]

Modeling

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

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

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

We then proceed to the 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. As apparent in Figure 15, the performance appears to flatline for models with or more than 2 layers and with or more than 50 nodes, yielding similar performance. To minimize the number of nodes, we thus proceed with the initial model configuration $\text{NN}(5 - 50^{\times 2} - 1)$ for the following optimization.

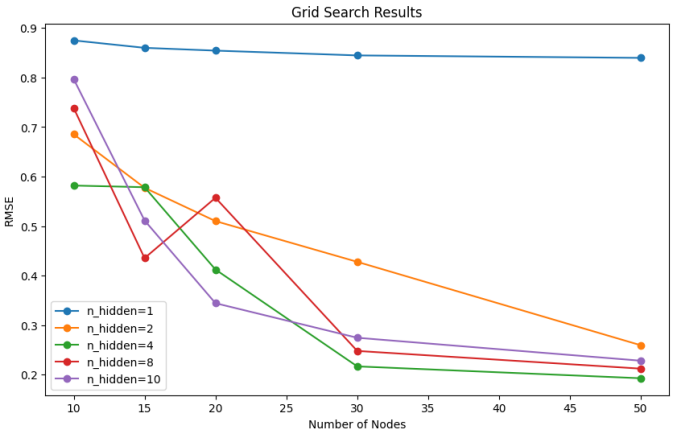

Figure 15: Grid Search results for a Neural Network configuration to predict for data with variable wind direction, showing similar performance for models of layers greater or equal than two, starting at 50 or more nodes

With a model chosen, we again investigate 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 greater deviation, especially at greater distances away from Turbine 1 for the case of wind direction 260°, as shown in Figure 16. The model appears to also show some artifacts, with the most notable being the highest values in parallel to the wake right at the borders of the wake flow itself.

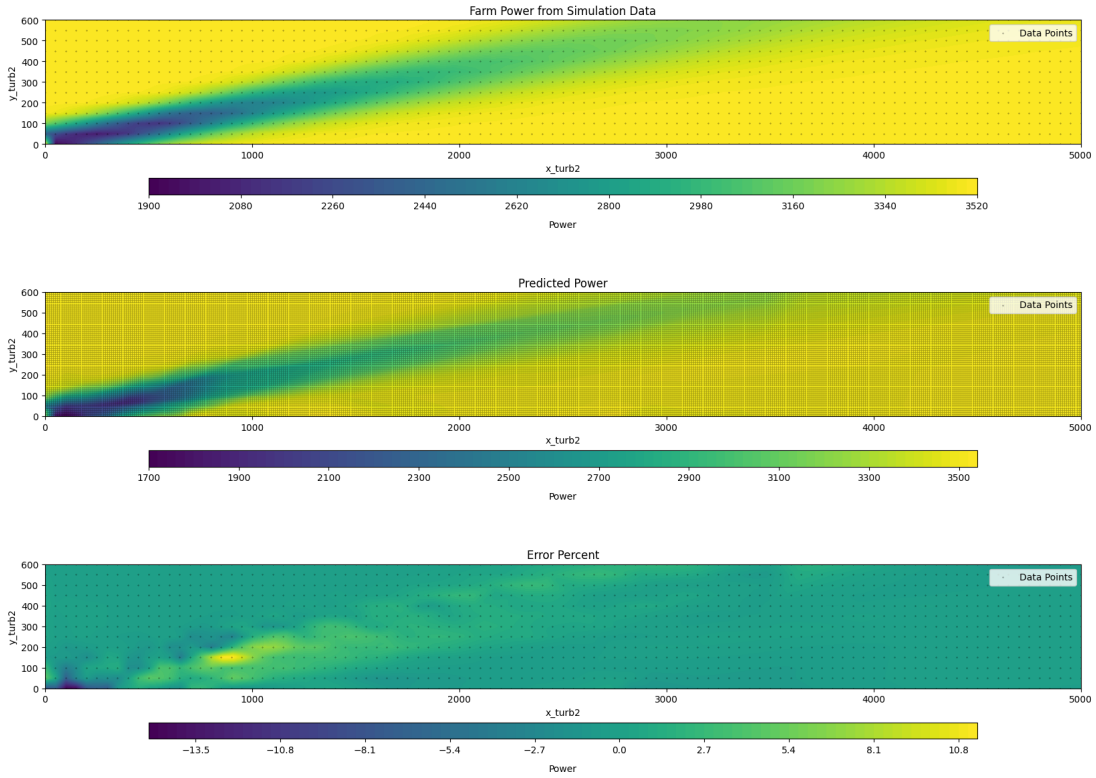

Figure 16: A heatmap of total generated farm power for a wind direction of 260° with datapoints and linear interpolation in between. Plot 1 shows the raw data, plot 2 the predictions of the NN$(5-50^{\times 2}-1)$, plot 3 the percentage difference between training points and predictions

Having observed the deviation described above, another layer is added to the model in an attempt to prevent the previous behaviour. The resulting configuration NN$(5-50^{\times 3}-1)$ improved the previous models weaknesses in that respect as can be seen in Figure 17, which is why *this is the final model chosen for the optimization*.

Since the parameter space has now become three-dimensional, conditioning on a single wind direction does not cover the full performance of the model. Figure 18 shows the development of RMSE across different wind directions, indicating that the border regions and wind directions with an increased total wake length (e.g. wind directions closer to 270° wind direction) experience higher RMSE, providing more space for errors to accumulate.

While increasing the size of the Neural Network might further reduce the deviations across wind directions and for the individual wind directions, it is restricted by the solvability of the optimization problem. While there is no exact threshold, the 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 model is deemed sufficient because of that trade-off.

For optimization purposes, the individual power per wind direction becomes less relevant, as the power outputs are weighted by their probability of occurring. To do so, first the probability distribution of the wind directions has to be defined, like the Normal distribution with mean 270° and standard deviation of 10° shown in Figure 19. To calculate the expectation for the discretized parameter space using the trained

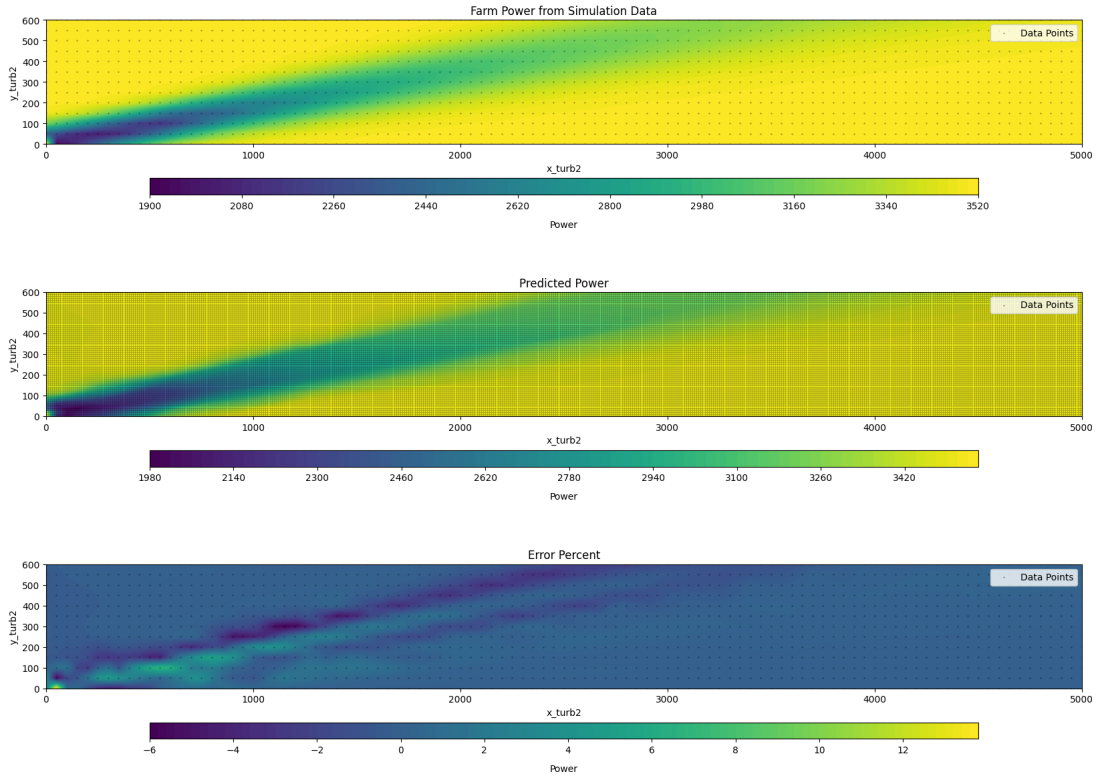

Figure 17: A heatmap of total generated farm power for a wind direction of 260° with datapoints and linear interpolation in between. Plot 1 shows the raw data, plot 2 the predictions of the NN$(5-50^{\times 3}-1)$, plot 3 the percentage difference between training points and predictions, overall showing improved

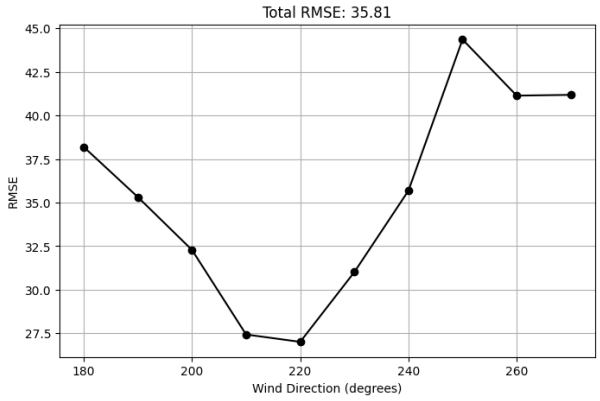

Figure 18: RMSE of NN$(5-25^{\times 10}-1)$ across the parameter space of wind direction

model, this distribution itself has to be discretized in line with the initial parameter space definition, which in this case is done in 10° steps.

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 20. The directions of the discretized wind direction probability distribution are shown as lines, and their impact on the distribution of expected farm power visible in the reduced expected power along them.

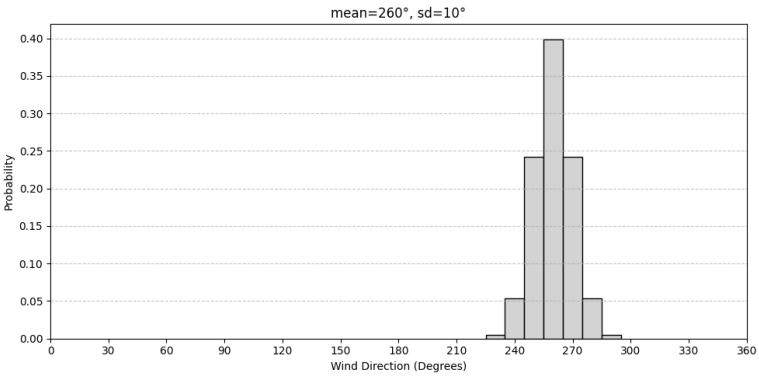

Figure 19: Discrete Wind Direction Probability Distribution with mean 270° and standard deviation 5°, discretized in intervals of 10°

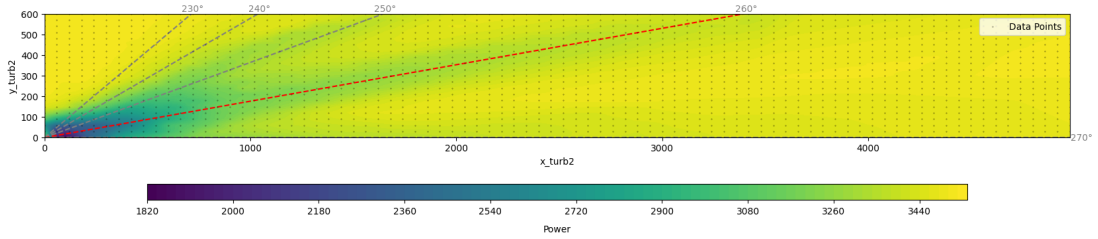

Figure 20: Distribution of Expectation of Farm power from a discretized Normal distribution with mean 260° and standard deviation 10° with step length 10°

In the same way the distribution of farm power was previously dependent on the deterministic wind direction, it is now dependent on the wind direction distribution, e.g. its mean and variance, assuming a Normal distribution. Changing the mean of the distribution corresponds 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 area affected 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 220° and standard deviation 20° shown in Figure 21.

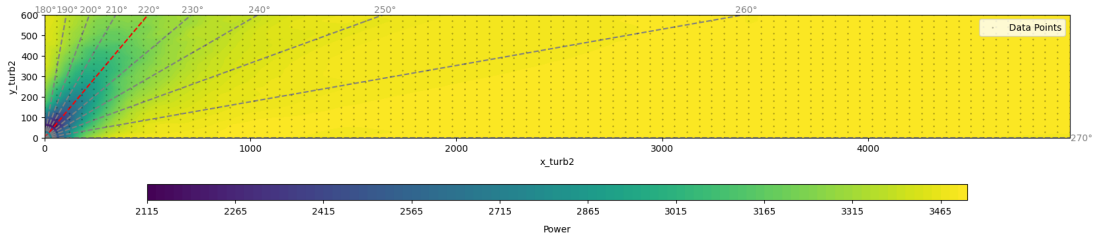

Figure 21: Distribution of Expectation of Farm power from a discretized Normal distribution with mean 220° and standard deviation 20° with steplength 10°

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

Optimization

With the model set up, solving the stochastic optimization problem for a maximum expected farm power output can be attempted. In the case of optimization, the problem is limited to three scenarios due to the computational constraints due to the size of the problem. A Normal distribution with a mean of 270° and a standard deviation of 5° was therefore used as the wind direction distribution, yielding three scenarios (260°,270°,280°).

The Neural Network and the wind direction distribution are introduced into pyomo to find a global maximum. As previously, a range of optima exists, with Pyomo choosing among this area, as seen in Figure 22. More specifically in this case, the area close to the wake had been shown to predict higher values than in the simulations by the model (see Figure 17), showing the optimization to find the global maximum within the provided Neural Network.

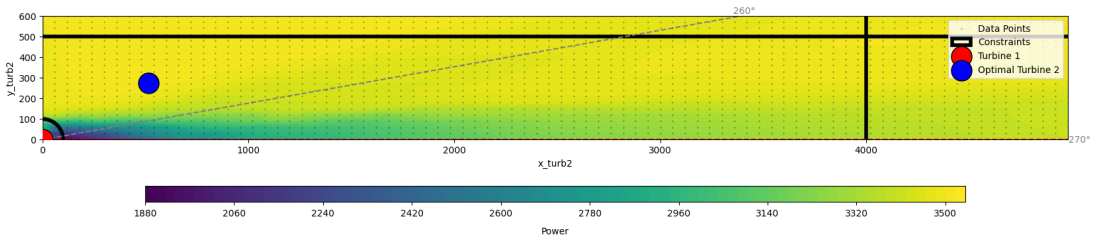

Figure 22: Optimization Result of the embedded wind distribution independent Neural Network, showing a optima at the border zone of the wake, where the model had shown to deliver high power values in its evaluation

Like before, it is possible to visually find that there is a large range of more or less equally optimal points, with the few scenarios not covering the space appropriately. This leads to seemingly optimal areas in between the scenario wind directions, where in reality, significant wake losses can be expected. In addition to that, some small deviations of the model due to its complexity affect the optimization outcome significantly, with the current model predicting a maximum farm power production for turbine 2 placement close to the wake of turbine 1. Both of those aspects are weaknesses in the current configuration of the stochastic optimization.

Stochastic Optimization: Quantile Discretization of Wind Direction Distribution

The first problem observed in this formulation as the gaps between the scenario wind directions as shown in Figure 22, can be partially solved by discretizing not with a constant step length but based on the quantiles of the distribution. The discretization can be instead performed not by a constant length of the discretization steps, but by 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. This approach, when used for the same Normal distribution with mean 270° and standard deviation 5°, fixing the scenario/quantile count to 7, yields the discretization shown in Figure 23.

Using this distribution, the expected power distribution changes for the corresponding scenarios/quantiles, with the space left between scenarios proportional to the probability density. When considering the previous approach to stochastic optimization, the number of scenarios that can be introduced into the problem has now

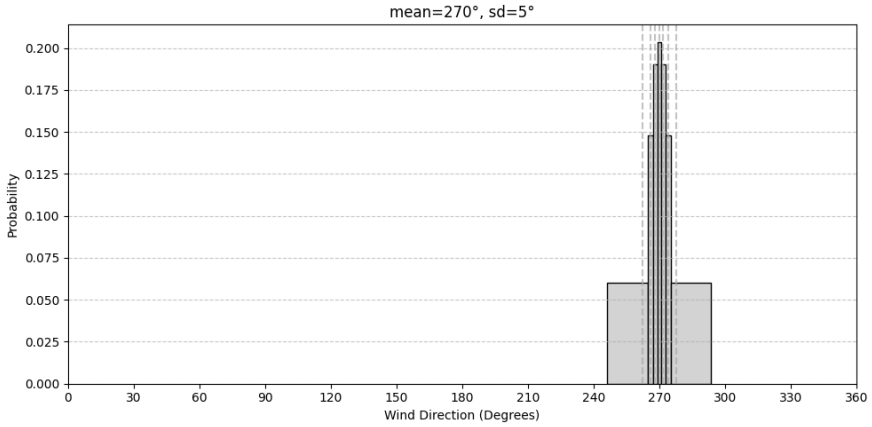

Figure 23: Quantile based discretization of Normal mean 270°, standard deviation 5°, disrcetized into 7 scenarios with equal probability for scenario and the scenario angle corresponding to the expected value for the quantile

become the main limitation. The following subsection provides an alternative set up that allows for a greater number of scenarios.

4.3.3 *Stochastic Optimization: Direct Expectation Neural Network*

While the expectation maximization has shown to be the most accurate depiction of the problem on a theoretical level, the previous method used for the resulting stochastic optimization has shown to be severely limited by computational power, as a large Neural Network is required to model the farm power output both across x and y ranges of the second turbine, as well as across the wind condition variables. To overcome these constraints, a alternative formulation is set up, where the Neural Network is designed to output the expected farm power at any x/y point, given a certain wind condition distribution.

$$\max_{\mathbf{x},\mathbf{y}} \mathbb{E}[f_{Power}(\Delta x, \Delta y) \mid \text{wind condition distribution}]_{\text{NN}} \tag{18}$$

$$\text{s.t.} \quad 0 \leq \Delta x \leq X_{\max} \tag{19}$$

$$0 \leq \Delta y \leq Y_{\max} \tag{20}$$

$$\sqrt{(\Delta x)^2 + (\Delta y)^2} \geq d_{\min} \tag{21}$$

Where:

- $(\Delta x, \Delta y)$ are the relative distances of the two turbines,
- $\mathbb{E}[f_{Power}(\Delta x, \Delta y) \mid \text{wind condition distribution}]_{\text{NN}}$ a Neural Network that predicts the Expected Farm Power at all $(\Delta x, \Delta y)$ positions in the parameter space, given a specific wind speed distribution
- $X_{\max}, Y_{\max}$ define the maximal distance the two turbines can be placed apart
- $d_{\min}$ is the minimum distance between the two turbines

In practice, the new Neural Network configuration is achieved by training the Network directly on the Expected Farm Power Values, with the only remaining variables

as the relative x/y position of the second wind turbine. To do so, the simulation values across the wind condition variables for a single location are used to calculate the expectation for that location as a probability weighted sum, with probabilities resulting from the distribution of the wind condition variables (wind direction, etc). To combine this approach with quantile discretization from Section 4.3.2, the resulting quantile scenario discretization is used to define the parameter space for the initial data set generating simulations. That way, a match between the scenario values and data set discretizations can be assured to allow for seamless calculation of the expectations.

The central benefit of this formulation is that the dimensionality of the Neural Network is significantly reduced, with only $\Delta x, \Delta y$ remaining free after calculating the expectations. The drawback is that the model is thus conditioned on a specific wind condition distribution and can not be used in a more flexible way.

The Notebook belonging to this formulation can be found in Stochastic optimization with farm power Expectation NN Formulation Notebook [42]

Modeling

To generate the model with the farm power expectation as output, more steps are required as initially described in Chapter 3.3. As detailed in the introduction, for the configuration used in this chapter, the Wind Condition distribution, which in this case will only be the distribution of wind directions, has to be defined. For this wind distribution, a discretization is generated the quantile based method described in Section 4.3.2. From this discretization, together with chosen limits for $\Delta x, \Delta y$, the parameter grid/space is defined and corresponding simulation performed to generate the data set. The probabilities from the wind condition distribution are then joined to the dataset and used to calculate the Expected Farm power for each $\Delta x, \Delta y$ combination in the parameter grid. This yields a significantly smaller data set with $\Delta x, \Delta y$ and Expected Farm power as features. This dataset is then used to train a model in the same way done previously, choosing a appropriat configuration of the model which then is used for embedding in the optimization problem. The described steps are shown in Figure 24.

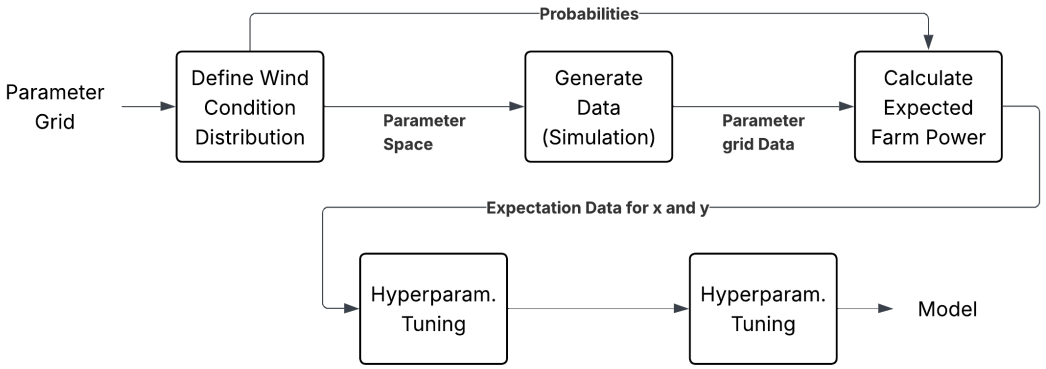

Figure 24: Flowchart displaying the steps used to set up a Neural Network model to model Expected Farm Power across Δx and Δy

As previously, we proceed to defining the parameter space, where the only difference compared to the previous stochastic formulation is that the wind direction how has variable steplength, using the method presented in Section 4.3.2.

Table 3: Value Ranges for Stochastic Expectation Neural Network Two Turbine Problem Data Set

Variable	**Const/Variable**	**Value**	*Steplength*
Δx_{turb2}	Variable	[0, 5000] m	50 m
Δy_{turb2}	Variable	[0, 500] m	50 m
wind_speed	Constant	8 m/s	-
wind_direction	Variable	[180°, 270°]	35 Quantiles
turbulence_intensity	Constant	0.06	-

As at this stage the parameter grid for wind direction is not yet well defined, the wind direction distribution and the number of discretization steps first has to be chosen. For this section, we proceed with the same Normal distribution with mean 260° and standard deviation 10° used in Section 4.3.2, but now discretized for 35 quantiles as shown in Figure 25.

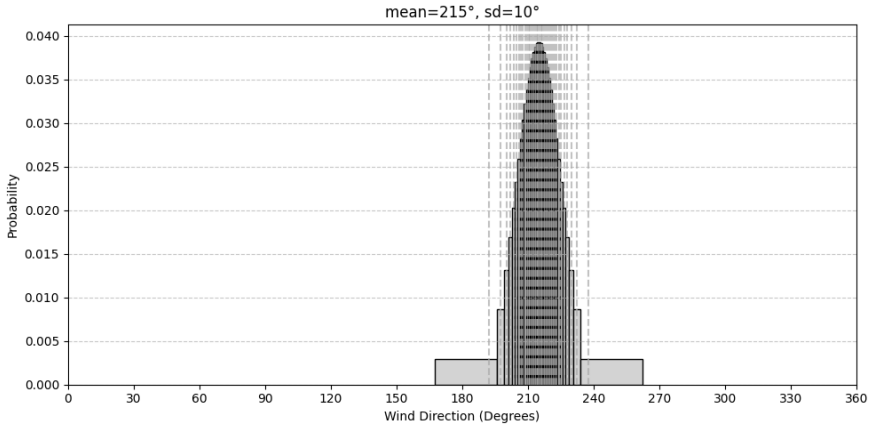

Figure 25: Quantile based discretized Normal with mean 260° and standard deviation 10°, discretized for 35 quantiles

With the parameter grid defined, the simulations can be performed, joint with the wind condition probabilities and expectations for the farm power calculated, yielding the dataset to be used in the training of the Neural Network. We thus proceed to the grid search optimization of the Neural Network configuration, as done in the previous chapters. For the distribution shown above, we find the 2-hidden layer and 20 neurons per hidden layer NN$(5 - 20^{\times 2} - 1)$ configuration to yield the best results for least numbers of neurons, as can be seen in Figure 26. *We thus proceed with the configuration* $NN(5 - 20^{\times 2} - 1)$.

The model of configuration NN$(5 - 20^{\times 2} - 1)$ is thus trained and a brief visual inspection performed to investigate how well the model fits the data. As shown in Figure 27, the model incorporates the shape of the Expected Power Distribution across x/y fairly well, with the only major deviations near the origin, e.g. the position of Turbine 1 and directly after Turbine 1 in the principal wind direction.

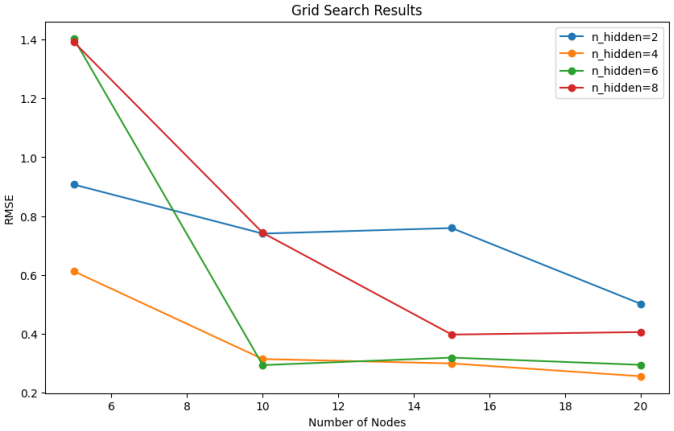

Figure 26: Grid Search Results for models trained on Farm Power Expectation data, with parameter grid as defined in Table 3 and Figure 25

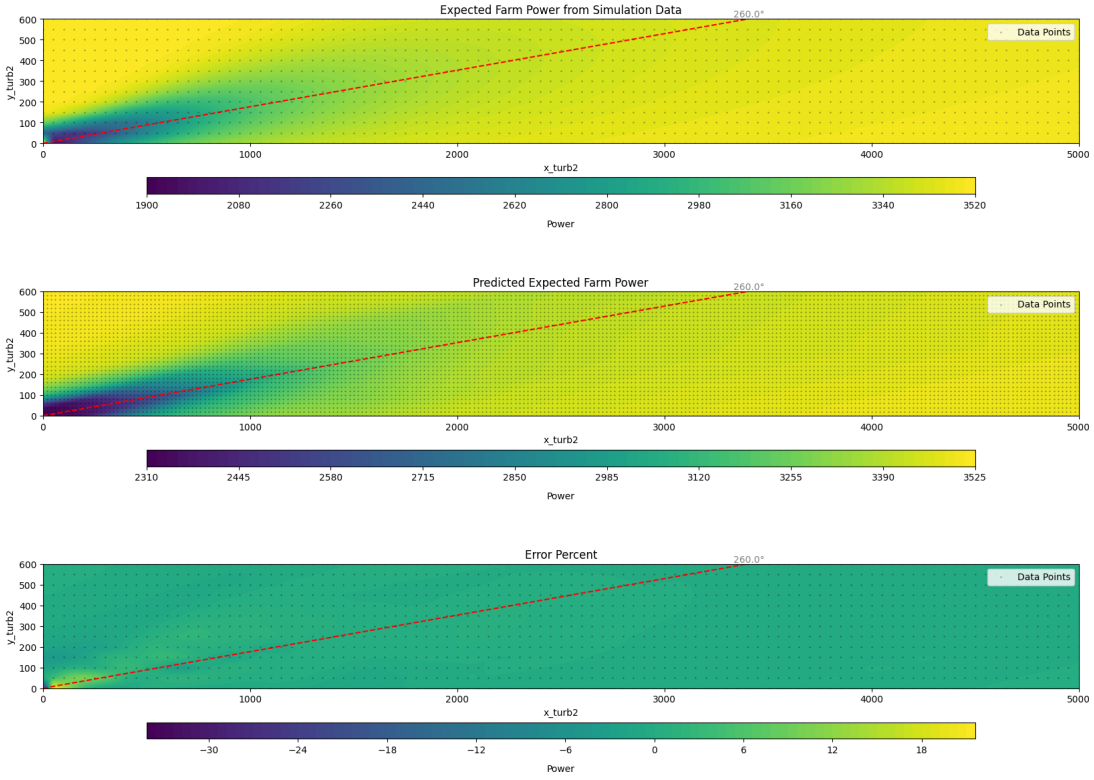

Figure 27: Visual comparison of training data vs predictions for Neural Network for Expectation of Farm Power, with linear interpolation in between data points to generate heatmap

The model thus appears to fit the physical behavior sufficiently well to be able to proceed to the optimization step.

Optimization

The set up model is thus again embedded into a now seemingly deterministic optimization problem, as the uncertainty has already been taken into account when setting up the Neural Network itself through the expectations. The Neural Network is thus embedded directly into the problem, with the constraints as initially defined.

The result shown in Figure 28 yields the expected result, with turbine 2 being placed as far upstream and as far away perpendicularly to the principal wind direction as possible from Wind Turbine 2.

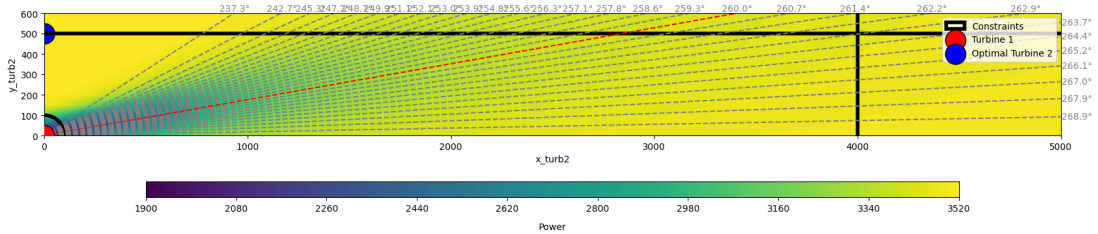

Figure 28: Optimization Result for maximization of Expected Farm Power, with wind direction Normally distributed with mean 260° and standard deviation 10°, discretized into 35 scenarios using quantile based discretization and Neural Network configuration $\mathrm{NN}(5 - 20^{\times 2} - 1)$

Notably, the solving of this variation optimization problem requires significantly less computational resources as the previous stochastic formulation, with a significantly larger quantity of scenarios considered for calculating the Expectation. Along the entire optimization pipeline, the main limitation is the number of simulations required for the individual discretization of different wind direction/wind condition distributions. Due to the efficiency of the FLORIS simulations, this does not seem to be a limitation for the two turbine configuration at this moment however. Therefore, this method appears to be better fit to scale up the complexity of the wind condition distribution, allowing for expanding to a multivariate distribution of the potentially dependent variables wind direction and wind speed for example.

5 CONCLUSION

The Thesis contains the training of a Neural Network to predict the total farm power or expected farm power generated by a wind farm consisting of two turbines. The inputs to this the model are the relative position of the second wind turbine to the first as well as the wind condition parameters like wind direction at the given location. The resulting Neural Networks were embedded into a mixed integer linear programming problem using an approach for decomposing a Neural Network into mixed-integer linear constraints. Multiple optimization problems were solved using this approach, beginning with a deterministic problem by only considering a dominant wind direction to find maxima for the total farm power. The formulation was then expanded to stochastic optimizations with a scenario based approach, to maximize the expected power generation, subject to a discretized distribution of the wind direction. Here two variants were developed, one via a large general neural network with the wind condition parameters as additional variables and one via a reduced Neural Network, trained on the already evaluated expected farm power for each x/y position of the second turbine.

While the deterministic approach could be quickly discarded as insufficient, both of the stochastic formulations yielded benefits and drawbacks. The first formulation, using a larger Neural Network independent of the wind condition distribution yielded a versatile variant, with a Neural Network that could be applied to any location and any wind condition distribution, while only being able to consider a low number of scenarios due to computational constraints. The second stochastic formulation evaded the computational limitations of solving a very large optimization problem by training its model directly on the farm power expectations, making both the Neural Network and the optimization model significantly more efficient, but losing generality due to the Neural Networks conditioning on the wind condition distribution.

Overall, the second stochastic formulation appears to be the only variant that sufficiently considers the randomness of the wind conditions like wind speed, while showing a path to scaling up computationally. In any case, results from both stochastic formulations were able to show how insufficient it would be to only consider a deterministic formulation.

With the second stochastic formulation having shown promising results in this thesis, the next steps would involve investigating how to generalize the formulation to a larger number of wind turbines. Here, a fundamental reformulation might be required, as the current approach using relative positions would cause the parameter space to grow exponentially. Nonetheless, this approach might be worthwhile to continue pursuing, as the underlying Constraint Learning Methodology has been shown to cope with the complexity of wind turbine wakes in this thesis.

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 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.

[2] 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.

[3] 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.

[4] 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.

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

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

[7] National Renewable Energy Laboratory (NREL). FLORIS: Flow redirection and induction in steady state. GitHub repository, 2025. Version 4.4.2, https://github.com/NREL/floris.

[8] 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, 2021.

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

[10] 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, 2024.

[11] 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, 2023.

[12] 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.

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

[14] 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.

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

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

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

[18] 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.

[19] 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.

[20] Antonio Alcantara and Carlos Ruiz. Distcl: A neural network-based distributional constraint learning tool for mixed-integer stochastic optimization. https://github.com/antonioalcantaramata/DistCL, 2022. Accessed: 2025-06-06.

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

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

[24] National Renewable Energy Laboratory (NREL). FLORIS: FLOw Redirection and Induction in Steady State. https://www.nrel.gov/wind/floris, Mar 2025. Open-source wind plant optimization tool.

[25] Mads M. Pedersen and Paul van der Laan and Mikkel Friis-Møller and Alexander Meyer Forsting and Riccardo Riva and Leonardo Andrés Alcayaga Román and Javier Criado Risco and Julian Quick and Jens Peter Schøler Christiansen and Bjarke Tobias Olsen and Rafael Valotta Rodrigues and Pierre-Elouan Réthoré. PyWake: an open-source Python wind farm simulation tool. https://topfarm.pages.windenergy.dtu.dk/PyWake/, 2025. Developed at DTU Wind Energy; computes flow fields, turbine-level power, and annual energy production for wind farms using various wake models.

[26] 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.

[27] 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.

[28] 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.

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

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

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

[32] 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.

[33] Simon Schmetz. generate_data_floris.ipynb - wind farm power modelling, 2025. Accessed: 2025-06-06.

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

[35] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[36] Michael L. Bynum, Gabriel A. Hackebeil, William E. Hart, Carl D. Laird, Bethany L. Nicholson, John D. Siirola, Jean-Paul Watson, and David L. Woodruff. *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, third edition, 2021.

[37] Antonio Alcántara-Mata. Distcl: A distributed contrastive learning framework. https://github.com/antonioalcantaramata/DistCL, 2024. Accessed: 2025-06-16.

[38] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.

[39] Simon Schmetz. Two-turbine wind farm optimization using constrained learning (deterministic). https://github.com/schmeti/uc3m_TFM_wind_farm_optimization_codebase/blob/main/Windfarm_power_modelling/0_two_turbine_problem_constrLearn_determin.ipynb, 2025. Accessed: 2025-06-06.

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

[41] Simon Schmetz. Two-turbine problem with constraint learning and probability-weighted objective. https://github.com/schmeti/uc3m_TFM_wind_farm_optimization_codebase/blob/main/Windfarm_power_modelling/0_two_turbine_problem_constrLearn_probweighted.ipynb, 2025. Accessed: 2025-06-06.

[42] Simon Schmetz. Two-turbine problem with constraint learning and probability-weighted objective (expectation neural network). https://github.com/schmeti/uc3m_TFM_wind_farm_optimization_codebase/blob/main/Windfarm_power_modelling/0_two_turbine_problem_constrLearn_probweightedt_expNN.ipynb, 2025. Accessed: 2025-06-06.