Imperial College London

Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Independent Research Project
Final Report

Wind farm wake control using convolutional neural networks

by

Jens Bauer

jtb20@imperial.ac.uk

GitHub login: acse-jtb20
https://github.com/acse-2020/acse2020-acse9-finalreport-acse-jtb20

Supervisors:

Prof. Matthew Piggott

Dr. Roy Brown (Frazer-Nash)
Zoe Goss (Frazer-Nash)

August 2021

Abstract

Wind turbines make up a substantial amount of all generated electricity but wake effects can degenerate the power produced by downstream turbines. Therefore, control systems that adjust the yaw angle of individual turbines to drive the wakes away from the downstream turbines can increase the overall power generated by the wind park. This work presents the first model-based wake steering algorithm that is fully reliant on Machine Learning (ML). The resultant model was trained by and validated against FLORIS, a controls-oriented wind farm modeling package.
A Convolutional Neural Network (CNN) is used to generate the wake of individual turbines which are combined to represent the flow field of a wind farm using a wake superposition model. This CNN can generate individual wakes with an average accuracy of 99.2% while being 18% faster than FLORIS for wind parks with fewer than 26 turbines. The local turbulent intensity (TI) and the power generated by the turbines is calculated by Fully Connected Neural Networks (FCNN) using the wind speeds along a horizontal line upstream of the turbines. The FCNNs can predict the power generation and local TI with a mean error of 3.7% and 4.6%, respectively. Using the NNs to perform wake steering has shown that the optimal yaw angles found, produced only 5% less power than a solution found using FLORIS while being 60% faster. This approach shows promising results for use in wind farm control systems to increase the total power generated by wind parks.

Keywords: Wind turbines, Machine Learning, Yaw angle optimisation, Fully Connected Neural Networks, Convolutional Neural Networks

Acknowledgements

I would first like to thank Professor Matthew Piggott for his guidance throughout the project and proofreading the report. Additionally, I want to thank Dr. Roy Brown and Zoe Goss of Frazer-Nash for the weekly catch-up calls, continuous support and proofreading the report. Lastly, I am deeply grateful to my family and Stephanie for their unwavering support.

Contents

Abbreviations

NN	Neural Network
CFD	Computational Fluid Dynamics
RANS	Reynolds-averaged Navier-Stokes
LES	Large Eddy Simulations
ML	Machine Learning
FCNN	Fully Connected Neural Network
DRL	Deep reinforcement learning
CNN	Convolutional Neural Networks
GPU	Graphics Processing Unit
SOS	Sum of squares
SLSQP	Sequential Least-Squares Programming
CPU	Central Processing Unit
RAM	Random-access memory
TI	Turbulent kinetic energy
MSE	Mean squared error
APWP	Average pixel wise percentage

1 Introduction

Climate change is one of the essential challenges that is threatening our current way of life. To prevent the worst consequences, the member states of the United Nations have agreed to limit the rise of global temperatures to two degrees Celsius (United Nations/Framework Convention on Climate Change 2015).

One key industry which needs to substantially decarbonise to reach this target is the energy sector. Currently, renewable energy sources make up 28% of the global electricity produced (IEA 2020). However, to reach the goals set out by the Paris Climate Agreement, the globally installed capacity for renewable power generation must grow to at least 42% by 2030 (IRENA 2020). Wind energy is the largest producer of renewable energy in the UK (Department for Business 2021) and it accounted for 19.8% of all UK produced electricity in 2019 (Goodman & Martin 2020).
To further increase the total energy produced by wind turbines, new wind parks need to be built and existing parks need to be more efficient. One way to increase the efficiency of already existing wind farms and newly built ones is by using wake steering. This work explains the use of Neural Networks (NN) to replace traditional, fluid models for the control of wind farm yaw angles to substantially improve the efficiency of wind parks.

2 Description of problem and existing work

Wakes of upstream wind turbines can lead to a significant decrease in power production in downstream turbines due to reduced wind speeds and higher turbulent intensities. This can lead to an approximate reduction of 20% in the annual energy production of a wind park (Barthelmie et al. 2009) and Lundquist et al. (2018) found that wake effects can result in roughly £500.000 in lost sales annually for a single wind farm. Wake effects have the most impact on the wind farm yield with increasing number of wind turbines (Jacobson & Delucchi 2009) and when wind speeds are below the rated values (Abdullah et al. 2012). To reduce the impact of the wake generated by upstream turbines, the upstream turbines can be misaligned to the wind (yaw angle) which will steer the turbulent wake away from the downstream turbines. This results in less energy generated by the yawed turbines but increases the overall amount of electricity produced. Yaw angle optimization can lead to power gains of up to 7% for offshore wind farms (Dou et al. 2020) and should ideally be done in real time.

The flow field calculations and yaw angle optimization are usually done using Computational Fluid Dynamics (CFD). A tool often used in industry is FLORIS (NREL 2020) which is an open-source code to compute steady-state wakes in wind farms. FLORIS uses analytical models to compute the turbine wakes, these models are computationally inexpensive and are based on a highly simplified physical description of the flow physics. These models can compute the power generated by wind farms with a minimum mean absolute error of 10-16% compared to Reynolds-averaged Navier-Stokes (RANS) (Martínez-Tossas et al. 2021) and Large Eddy Simulations (LES) (Bastankhah & Porté-Agel 2014). FLORIS is widely used in wake steering and wind farm optimization (Bensason et al. 2021, Cioffi et al. 2020, Gao et al. 2020), and the calculated power increase is within 2% error to the theoretical maximum for a simple wake steering case (Simley et al. 2020).

Other methods which solve the full Naiver-Stokes equations such as RANS and LES are more accurate in modelling the transient turbulent flows but they are multiple orders of magnitude slower than FLORIS. RANS calculations are done in the order of hours (Tabib et al. 2015) while LES might need days (Bay et al. 2019), which makes them unsuitable for use in real time control systems.

There has always been a trade-off between speed and accuracy, however, machine learning (ML) might be able to solve this problem. ML has already been used to generate wind turbine wakes. Ti et al. (2020) used a fully connected neural network (FCNN) to predict the three-dimensional velocity field of a wind turbine with an error of less than 5% when compared to RANS data (Ti et al. 2020). The

flow field of a wind farm can also be modelled using FCNNs and the results are in good agreement with CFD data (Feng & Den 2020). Other work has shown that flow fields generated by a FCNN can be used for wake steering with good results (Van Dijk 2020, Fischetti & Fraccaro 2019).
Another option for using ML in wake steering are model-free, deep reinforcement learning models (DRL). DRL uses a neural agent which interacts with an environment and learns to maximise its rewards. It has been shown to work well for wake steering (Dong et al. 2021, Tomin et al. 2020) but the training takes significantly longer than other methods due to their low sampling efficiency and tendency to unlearn already learned policies.

Model-based methods rely on the velocity field to calculate the power generated by the individual turbines, see Inoue et al. (2005) for a detailed explanation. However, NN can be used to approximate this function, Świątek & Dutka (2015) used FCNN to predict the power output of a wind farm from the inlet wind speed and TI with less than 15% error.

All existing studies that model the flow around wind turbines with ML use FCNN rather than Convolutional Neural Networks (CNN). CNN is a type of neural network which uses filters rather than neurons. This makes CNNs more efficient, and they are often used in image generation due to their ability to reproduce spatial features very accurately. CNNs have successfully been used to generate flow fields in various applications, such as the flow around airfoils (Donglin et al. 2020, Wu et al. 2020), or smoke simulations (Kim et al. 2019).

Using trained NNs instead of CFD models reduces the amount of time it takes to calculate the power generated by a wind farm since an optimiser needs to evaluate many different yaw angles to find the best solution. This will improve the speed at which an active control system can respond to changes in wind speed, TI or wind direction. Ideally, a NN based algorithm can find the optimal yaw angles with LES accuracy in real-time, resulting in a large increase in the power generated by wind parks.

3 Objectives of project

This project uses a CNN to generate the wake behind wind turbines for different inflow conditions. Multiple turbine wakes can be stitched together to represent the wake of a wind park using a wake superposition model. This results in a surrogate model which can calculate the total power output of the wind farm for different inlet conditions and yaw angles. The power and local TI predictions are done by FCNNs. Figure 1 shows a flow chart of the surrogate model, which can be used by a global optimization method to find the yaw angles maximising the total power generation.

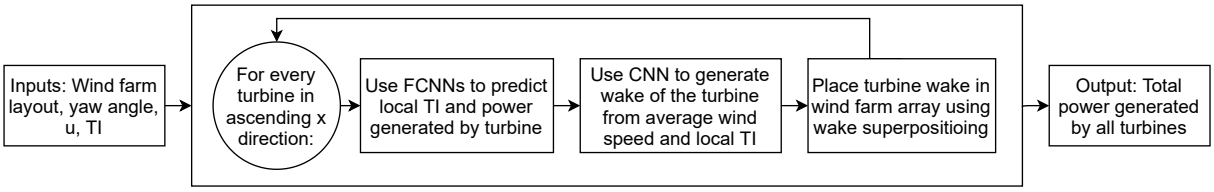


Figure 1: Schematic of main algorithm to calculate wind park power generation.

The aim is to further reduce the time taken to find optimal yaw angle in an active control system. To summarise, the objectives of the project are:

- Train CNN to predict flow-field around wind turbine under different inflow conditions.
- Create a surrogate model which calculates the total power output of the wind farm from the flow-field using FCNN to predict the power generated by every wind turbine and local TI conditions.
- Use surrogate model to optimize yaw angles for maximum power generation.

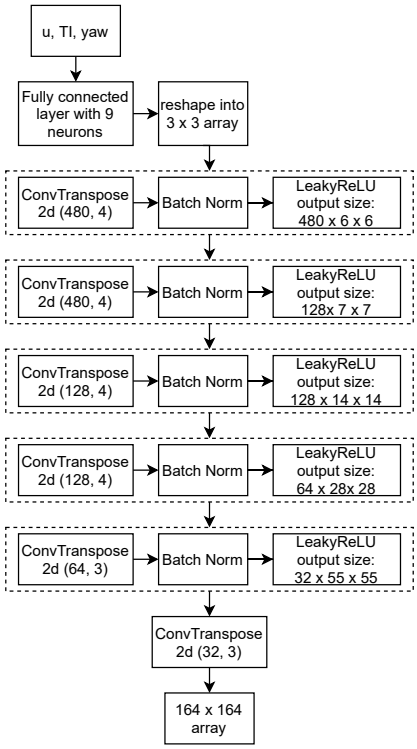
This work showcases the first model-based wake steering algorithm, that consists of only NNs while previous methods used mathematical operations to calculate the power output and local TI. Other novelties are:

- The use of a novel CNN & FCNN architecture to predict the flow field around wind turbines.
- The use of a FCNN to calculate the power generated by turbines from the wind speed along a line segment upstream of the turbines.
- The use of FCNN to predict the local TI at turbines from the wind speed along a line segment upstream of the turbines.

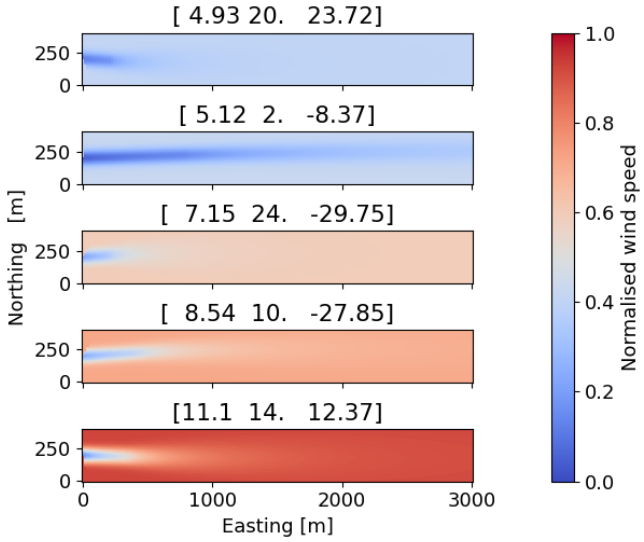
4 Methodology and software description

4.1 Flow field prediction

A combination of fully connected and deconvolutional layers are used to build the network which reconstructs the flow field around a single wind turbine. The first layer is a fully connected layer that takes in the wind speed, TI and yaw angle of the turbine at hub height and outputs 9 neurons. The output is reshaped into a 3 x 3 array which is passed to the deconvolutional layers. This fully connected layer was added because it improved the generalization capability compared networks consisting of only deconvolutional layers. There are 6 deconvolutional layers, each layer consists of ConvTranspose2d (PyTorch 2019), 2D batch normalisation (Riva 2021) and the leaky ReLU activation function (Glorot et al. 2011). The output of the model is a two-dimensional array with 163 x 163 pixels, which is a reconstruction of the flow field of size 3000 m x 400 m. The number of pixels is set by the network architecture while the physical dimensions of the flow fields are determined by the training set. This network architecture was the best performing architecture in the hyperparameter search where different models were tested. Figure 2a shows the network architecture and 2b shows example training wakes and the corresponding flow conditions.



(a) Representation of NN used to generate flow field with number of filters and stride used in every layer in brackets.



(b) Example normalised wakes used for training and their corresponding labels (u [m/s], TI[%], yaw [°]).

Figure 2: Fully connected and deconvolutional layers and training set examples.

The training set consists of 5000 two-dimensional flow fields generated by FLORIS using the Gauss model (Bastankhah & Porté-Agel 2014) which is described in more detail in Appendix A. FLORIS generates a three-dimensional wake of a NREL 5-MW wind turbine with a 126 meter rotor diameter. A horizontal two-dimensional slice at hub height is extracted from the generated wake. These wake images determine the physical size of the flow field and can be easily changed if a larger domain is needed. The shape of the array needs to be the same as the CNN output shape. The training set is generated by randomly sampling a range of inlet wind speeds of [3, 12] m/s, yaw angles [-30, 30] ° and turbulent intensities [1, 25] % which is based on reported TI values in wind farms and offshore

measurements (Argyle et al. 2018, Marek et al. 2016). This wind speed range is based on the fact that no further power gain can be achieved when the wind speed is above rated power (12 m/s) or below cut-in speed (3 m/s). The yaw angle range is based on (Bensason et al. 2021) which suggest that the optimal yaw angle range is [-25, 25]° but an additional 5° were added to extend the range. The training set was normalised such that all wind speeds are between 0 and 1 by dividing the whole data set by 12. This has significantly improved network learning and no other normalisation method showed further improvements. The network will learn to generate the wind velocity in the wake of the turbine for a given inlet condition and therefore it will implicitly learn the boundary conditions such as surface roughness or boundary layer effects imposed by FLORIS.

The model was trained on a Nvidia K80 graphics processing unit (GPU) for 100 epochs which took 42 minutes. The learning rate was initially set to 0.009 but a learning rate scheduler was used which reduces the learning rate by a factor of 0.6 if the loss did not drop in the past 6 epochs which has helped to further reduce the error. The Adam optimizer (Kingma & Ba 2017) has proven to be faster converging than other algorithms. As a cost function, the mean square of the L_2 norm (MSE loss) was used. In comparison to the L_1 norm or cGAN architecture (Mirza & Osindero 2014), the MSE loss was faster converging and generated images with a smaller absolute error.

4.2 Wake superposition

The network described in 4.1 generates flow fields around individual turbines. To describe wind speeds in a wind park, the individual turbine wakes are stitched together using a wake superposition model (Gunn et al. 2016, Vogel & Willden 2020) which preserves the interactions between different turbine wakes.
The wake of the individual turbines are being calculated in order of their x location, meaning in ascending streamwise order. If turbines have the same x location, it is being calculated in ascending y order. This ensures the wake effects of all upstream turbines are accounted for. The local wind speed for every turbine is found by averaging the wind along a horizontal line segment about 50 meters upstream of the blades. The local conditions (mean u, local TI) are used to generate the flow field of every turbine individually. Every wake is placed in the wind farm array using a wake superposition model. There are three superposition models available:

$$\text{Root sum of square (SOS): } u = \left(1 - \sqrt{\sum_{i=0}^{n}\left(1 - \frac{u_i}{u_{hub,i}}\right)^2}\right) u_{park} \tag{1}$$

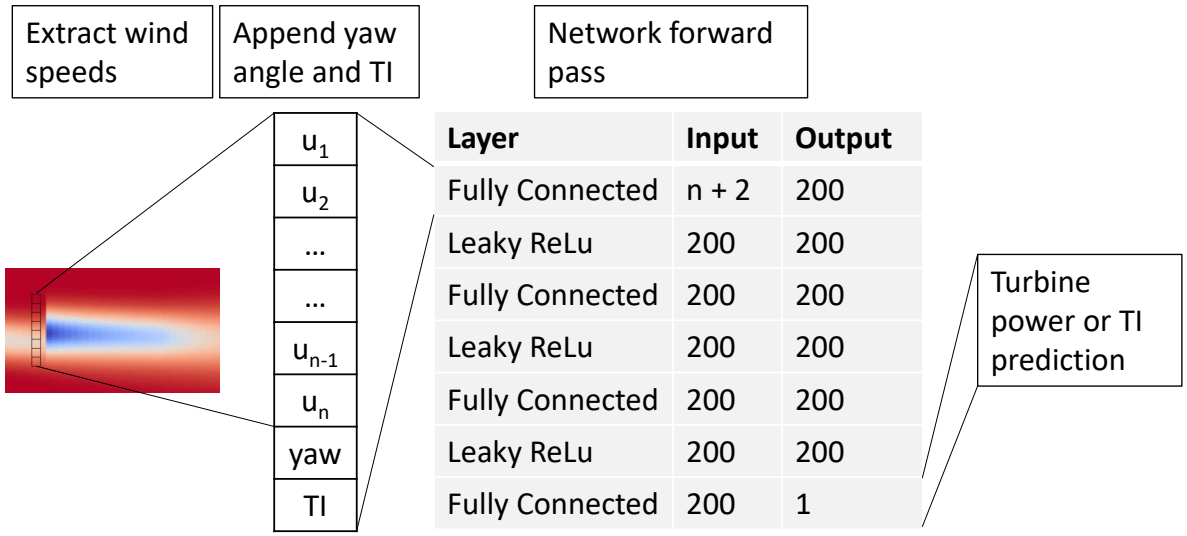
$$\text{Linear: } u = \left(1 - \sum_{i=0}^{n}\left(1 - \frac{u_i}{u_{hub,i}}\right)\right) u_{park} \tag{2}$$

$$\text{Largest deficit: } u = \min\left(\frac{u_1}{u_{hub,1}}, \frac{u_2}{u_{hub,2}}, ..., \frac{u_n}{u_{hub,n}}\right) u_{park} \tag{3}$$

where u is the wind speed of the combined wakes, u_i is the i^{th} wakefield which is normalised by the corresponding turbine hub wind speed $u_{hub,i}$, n is the number of wakes impacting the location and u_{park} is the wind velocity at the park inlet. The best superposition model depends on the turbine locations (Vogel & Willden 2020) but for this project, the SOS model is the proposed first choice since it performs well for most cases and it is used in the rest of the report.

4.3 Power and TI prediction

The power generated by a wind turbine is usually calculated from the three-dimensional flow field. However, the CNN only generates a two-dimensional slice which means a lot of information about the wind speed in the domain is not known, leading to the problem that the power generated by a three-dimensional turbine needs to be predicted using two-dimensional data. To solve this problem, a FCNN is trained to predict the power output of wind turbines from the wind speed along a horizontal line 50 meters upstream the turbine. This line is parallel to the turbine and stretches along the whole diameter of the blades.
The TI of the flow varies within the wind park due to the turbulent wakes. Therefore, a second network is used to predict the local TI at the turbine using the same flow data along the line. Both networks also require the yaw angle and inlet TI to make the predictions. The network should learn to infer the third-dimensional power output from the two-dimensional data it is provided with, which should reduce the error in the power output prediction. Figure 3 shows how the predictions are made and the network architecture.



Layer	Input	Output
Fully Connected	n + 2	200
Leaky ReLu	200	200
Fully Connected	200	200
Leaky ReLu	200	200
Fully Connected	200	200
Leaky ReLu	200	200
Fully Connected	200	1

Figure 3: Schematic of power and TI prediction using the fully connected network and the network architecture of both networks.

The networks were trained on 4000 example wind speeds along the line upstream of the turbine and the corresponding FLORIS power output or TI. The training data is generated by four example wind farms, consisting of six turbines, which are shown in Appendix B. To generate the training data, the wind speed along the line, yaw angle and inlet TI for every turbine is saved with the corresponding power generation or local TI. The power and local TI are calculated from the third-dimensional flow field while the extracted flow data is from a two-dimensional slice. The inlet wind speed, TI and yaw angle are varied with the same limits as described in 4.1.

Both networks share the same architecture because it simplifies the code and it required less time to perform a hyperparameter search while still achieving good results. The network architecture was chosen to favour the power prediction because it was deemed more important to accurately predict the power output for yaw angle optimisation. However, the local TI predictions have an implicit impact on power prediction but in this case, both errors are small enough that the power prediction could be prioritised. The power network was trained with a learning rate of 0.003 for 150 epochs while the TI network was trained with a learning rate of 0.0065 for 80 epochs. Both used the Adam optimizer and a batch size of 250.

4.4 Wake steering

Combining all the steps described above results in a surrogate model that can predict the power generated by a wind farm for different inflow conditions and yaw angles. To maximise the power output of the wind farm, the optimal yaw angles need to be found which can be stated as a minimization problem:

$$\arg\min_{\mathbf{y}} -E(\mathbf{y}), \text{ subject to: } y_i \in [y_{min}, y_{max}] \tag{4}$$

where $\mathbf{y}$ is the yaw angles of all turbines in the wind farm, E is the loss function which is the power output of the wind farm for given inflow conditions calculated using the surrogate model or FLORIS, $y_i \in \mathbf{y}$ and y_{min} and y_{max} are the yaw angle bounds. This work uses the SciPy Sequential Least-Squares Programming (SLSQP) algorithm (SciPy v1.7.0 Manual n.d.) which is based on Kraft (1988), Boggs & Tolle (1995) to find the yaw angle that maximises the power output.

4.5 Code description and metadata

The code developed for this project is called CNNwake and it is a robust, standalone, fully tested, open-source package. The software is split into two parts, the first part defines the NNs and their training. The second part is the wind park surrogate model and the yaw angle optimization. Figure 4 shows an overview of the code structure and which user and FLORIS inputs are required for the model training and optimisation.

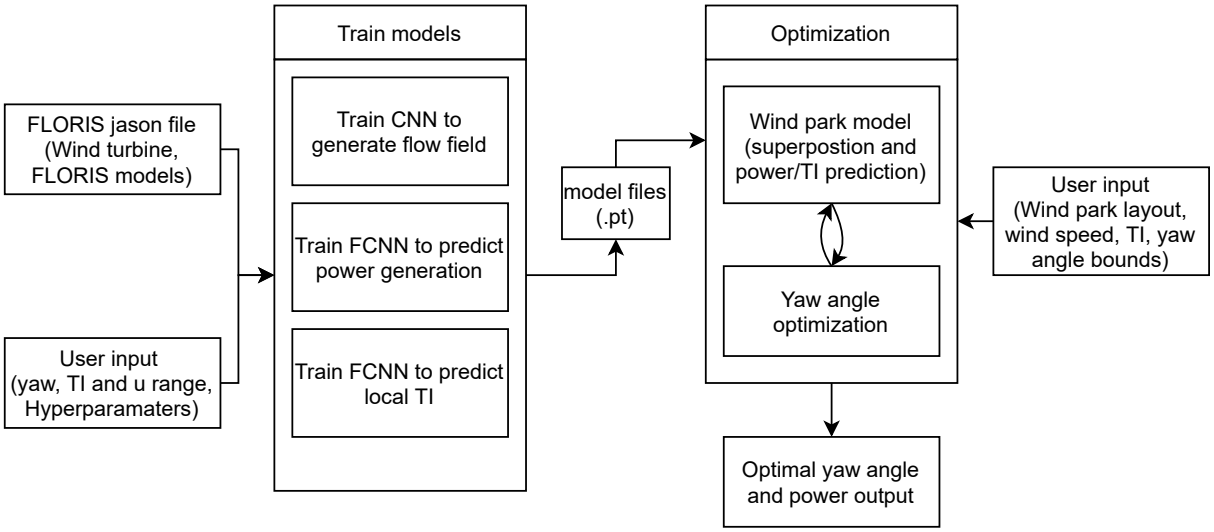


Figure 4: Schematic of the software package.

The CNN and FCNN are separate, self-contained classes. These classes include methods for training, training set generation, model validation and processing which simplify the training/usage of the models. All NNs can easily be retrained with different input ranges or domain sizes. The wind park model is a function that takes in the trained models and the wind park characteristics and outputs the power produced by the wind park.
The code is written in Python 3.8 and was developed and tested on Windows 10 19041.1110 64 bit with an Intel i5-6300HQ central processing unit (CPU), a Nvidia GeForce GTX 960M and 8 GB of RAM. All NNs were trained on a separate system, using a Nvidia K80 GPU and Intel Xeon CPU. CNNwake relies on multiple open-source libraries, the main ones are listed below:

- FLORIS 2.4
- Numpy 1.21.0 (Harris et al. 2020)

- SciPy 1.7.0 (Virtanen et al. 2020)
- Torch 1.9.0 (Paszke et al. 2017)
- Matplotlib 3.4.2 (Hunter 2007)

For a list of all external libraries used and a user guide, please go to CNNwake GitHub: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-jtb20.

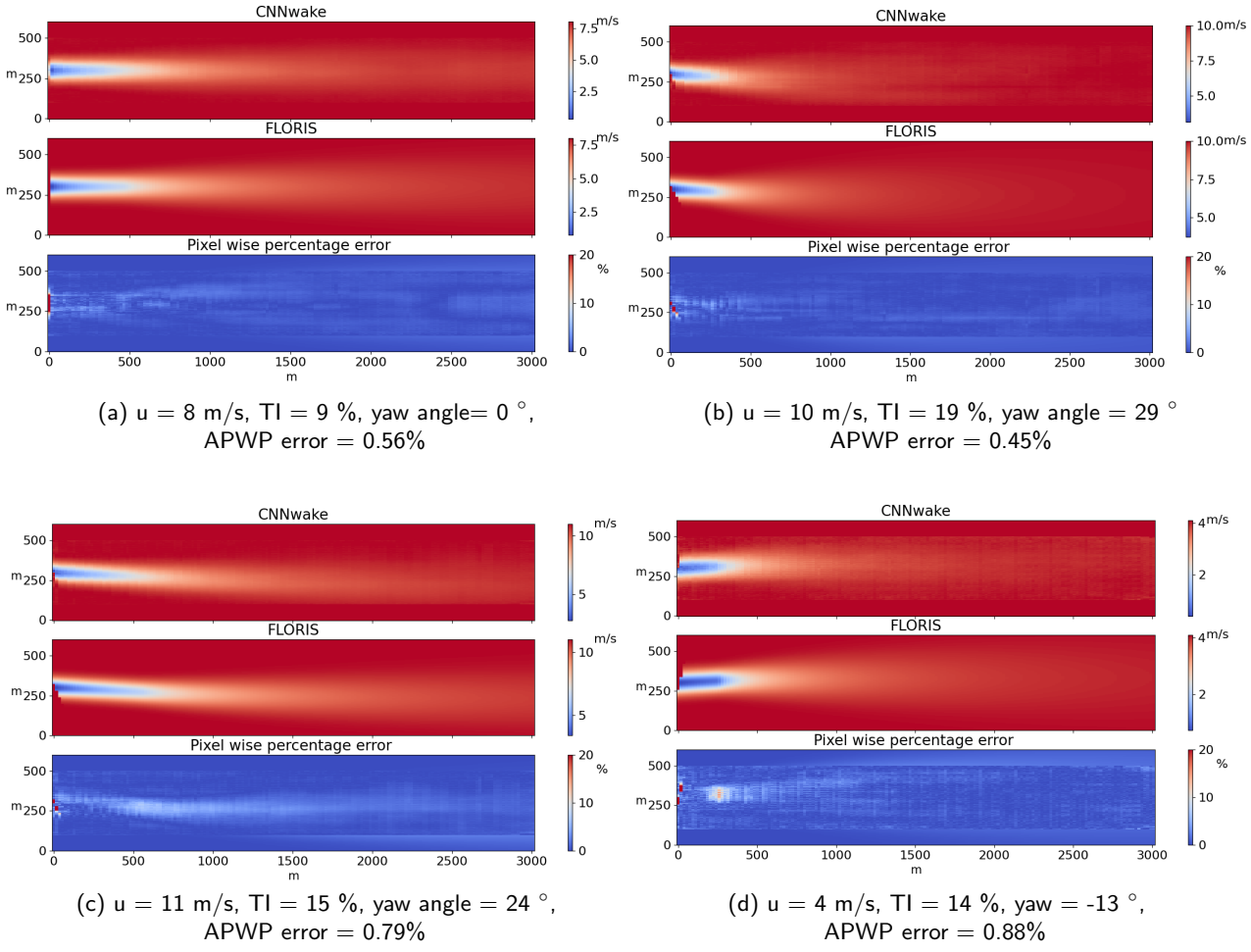
5 Results

5.1 Turbine wake prediction

CNNwake can predict the wake of individual turbines for a given inlet condition. The resulting array can be compared to the wake generated by FLORIS and an average pixel wise percentage (APWP) error can be calculated using the following equation:

$$\mathrm{APWP} = \frac{100}{n}\sum_{j=1}^{n}\frac{1}{p}\sum_{i=1}^{p}\frac{|\ \mathrm{FLORIS}_{j,i} - \mathrm{CNNwake}_{j,i}\ |}{\max \mathrm{FLORIS}_j} \tag{5}$$

where n is the number of wakes used, p is the number of pixels, $FLORIS_{j,i}$ and $CNN_{j,i}$ are the i$^{\text{th}}$ pixel of the j$^{\text{th}}$ wake for FLORIS and CNNwake respectively. CNNwake's APWP error on a set of 600 unseen inflow conditions is 0.8%, which means the CNN wake prediction is 99.2% accurate. Figure 5 compares four example flow fields generated by the CNN to FLORIS.



(a) u = 8 m/s, TI = 9 %, yaw angle= 0 °, APWP error = 0.56%

(b) u = 10 m/s, TI = 19 %, yaw angle = 29 °, APWP error = 0.45%

(c) u = 11 m/s, TI = 15 %, yaw angle = 24 °, APWP error = 0.79%

(d) u = 4 m/s, TI = 14 %, yaw = -13 °, APWP error = 0.88%

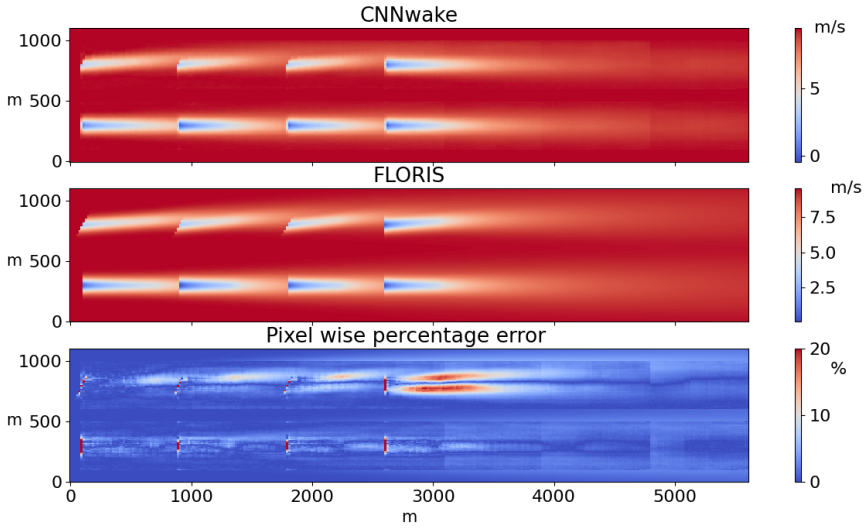
Figure 5: Comparison of four velocity fields generated by CNN and FLORIS. The pixel wise percentage error is the percentage error of every pixel of the CNN output compared to FLORIS.

As seen in Figure 5 and from the low APWP error, the CNN can generate wind turbine wakes that match wakes generated by FLORIS very well. In the area around the rotor, the error is above 20%, because the network is not able to represent the sharp changes in wind speed at the rotor. Similarly, the error in the near wake, up to 500 meters downstream, can be up to 15%. However, this does not

impact the use for wake steering since most wind turbines are placed at least 6-15 rotor diameters (Meyers & Meneveau 2012) downstream where the error of the CNN drops mostly below 5%.

5.2 Farm wake prediction

The individual turbine wakes are stitched together using the SOS wake superposition model. Figure 6 compares the CNNwake generated flow field for two wind farms with the output created FLORIS.



(a) 2x4 wind farm, u = 8.7 m/s, TI = 6%, yaw = [0, -30, 0, -30, 0, -30, 0, 0], APWP error = 1.9%.

(b) offset 3x3 wind farm, u = 11 m/s, TI = 5 % , yaw = [30, -30, 30, -30, 30, -30, 30, -30, 30], APWP error = 4.8%.

Figure 6: Comparison of wind farm wake generated by CNNwake to FLORIS.

CNNwake demonstrates that it can accurately superposition the individual turbine wakes to generate the flow field of different wind farms. This results in a mean APWP error, from the two cases shown above, of 3.4%.

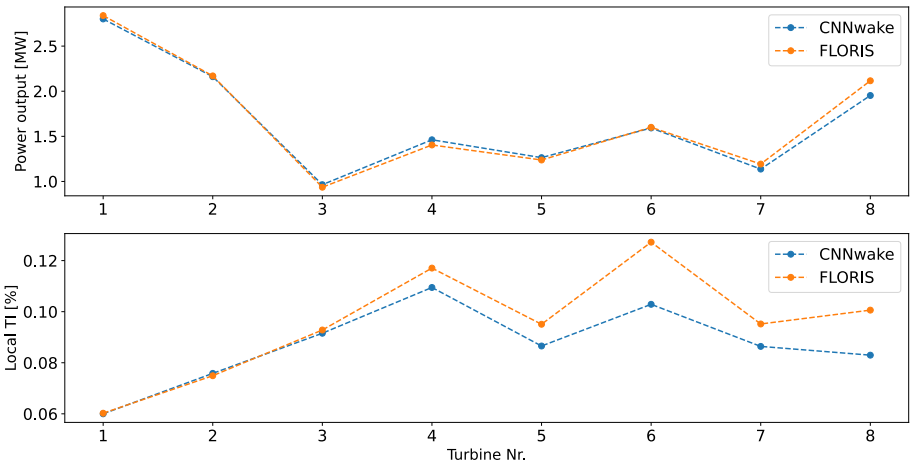
However, there are a few limitations to consider with this method. Firstly, the error in the top row of Figure 6a is above 15% and especially large for the last turbine which is the only turbine of that row with no yaw angle. The reason for the large error in the top row is that the incoming flow at the turbines is already at an angle from the upstream wakes, which will result in an additional sideways draft for the wake. This inflow effect is part of the FLORIS model used and it can be seen having a slightly upward direction in the last turbine's wake as if it had a yaw angle.
CNNwake only considers the average wind speed upstream of the rotor and can therefore not take into account any additional effect the incoming flow angle will have on the wake. The example flow field was selected to showcase a worst-case scenario; for the yaw angle optimization this effect can lead to a slight overestimation of optimal angles.

Secondly, for wind speeds and TI conditions that produce wakes larger than 3000 meters, as shown in Figure 6b, the error in the far wake of the wind farm is about 10%. The reason is that the wakes generated by CNNwake extend 3000 meters downstream. This means that any location three kilometres downstream from the first turbine will not incorporate the wake effect of that turbine. If larger domains are required, the CNN model can be retrained on data with a larger domain size.

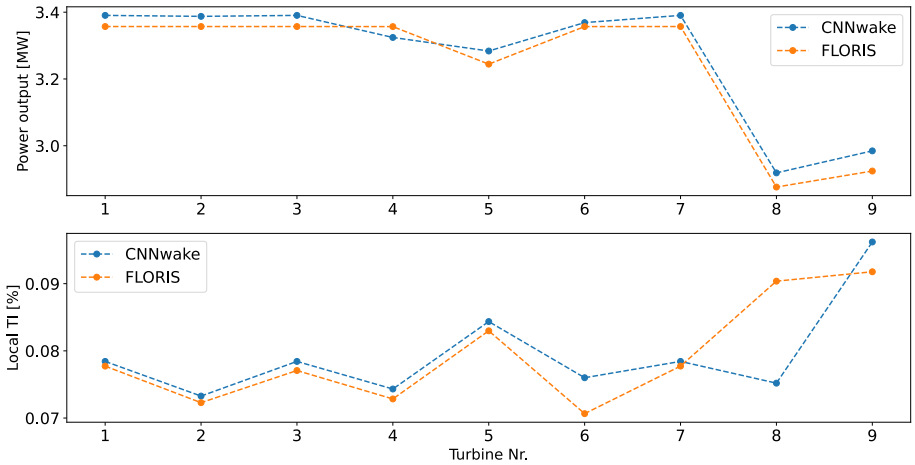
Overall, CNNwake can generate the wakes of wind parks with APWP errors of less than 5%. Even for dense wind parks with inflow conditions that promote large wakes, the APWP error is only 4.8% with the maximum error in the near wake of the turbines.

5.3 Power and TI prediction

The power generated by every turbine and the local TI at the turbine hub is calculated using FCNNs as described in section 4.3. Figure 7 compares the power and local TI predictions of CNNwake with FLORIS for the wind parks shown in Figure 6.



(a) Power and TI predictions of FCNNs for wind farm in Figure 6a.



(b) Power and TI predictions of FCNNs for wind farm in Figure 6b.

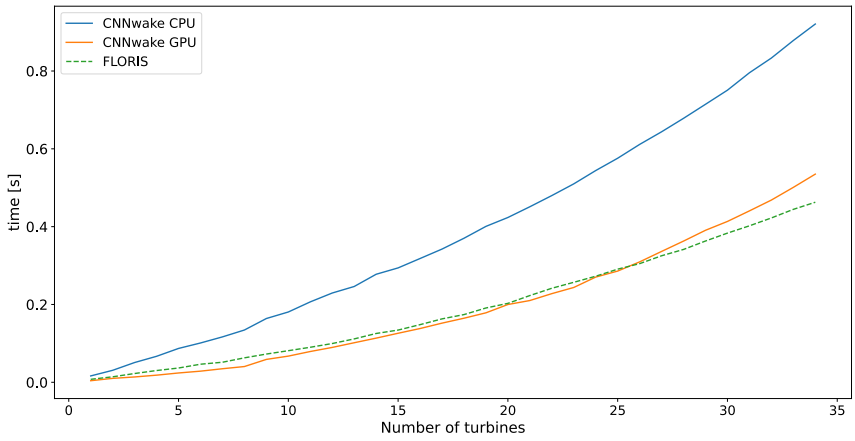
Figure 7: Comparison of power and TI predictions by CNNwake to true values calculated by FLORIS. The turbines are numbered by ascending x location, if turbines have the same x location, they are ordered by ascending y location.

CNNwake is able to accurately predict the power generated by the individual turbines. The largest percentage error in the power prediction is 3.5% while the average percentage error for both wind farms is less than 2.3% for the conditions given in Figure 6. The local TI predictions by the FCNN have a larger error when compared to FLORIS. The maximum percentage error is 16.8% at turbine 8 in Figure 7b while the overall TI error for both wind parks is about 8.6%.
For 200 randomly generated velocity and TI inflow conditions, the average percentage error for the power and TI prediction of both wind parks combined, was 3.7% and 4.6%, respectively.
When selecting the network architecture for both FCNN, accurate power prediction was prioritised rather than ensuring that both have a similar, but larger error. Prioritising the power network was done because the power output is more important for wake steering than local TI conditions even though the local TI has an implicit impact on accurate power predictions.

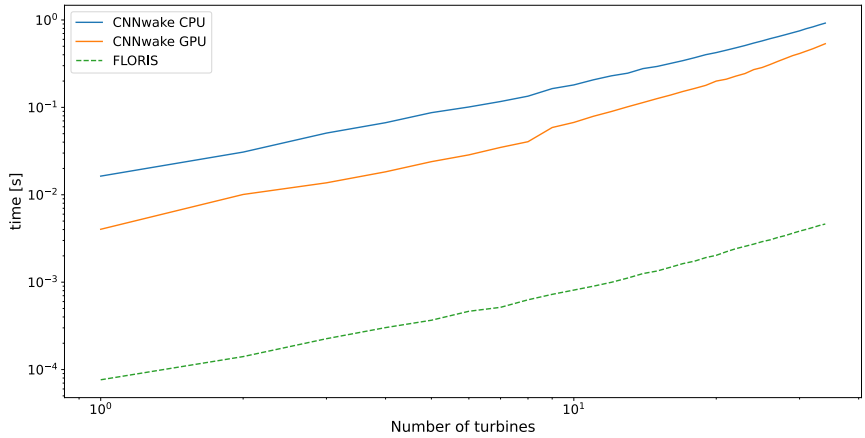
5.4 Computational time

Reducing the time taken to calculate the power output of a single wind farm is important since a global optimisation method will use it thousands of times. Figure 8a shows the time taken to compute the wake field and total power output for an increasing number of wind turbines. The timings for every wind park size were averaged over 5 runs.

CNNwake is 18% faster than FLORIS when run on a GPU for wind parks with less than 26 turbines, while CNNwake run on a CPU is significantly slower. If a wind park has more than 26 turbines, FLORIS is faster in computing the wake field, which might be due to a sub-optimal implementation of the CNNwake superposition algorithm which is significantly slower for large arrays. The superposition approach is the same as in FLORIS but CNNWakes's implantation is different.
Figure 8b normalises the FLORIS time to account for the fact that FLORIS three dimensional output is compared with CNNwake two dimensional output. FLORIS's output is of size 100 x 164 x 164, therefore dividing its' time taken by 100 will normalise the timings. Considering the normalised time, FLORIS is about two orders of magnitude faster than CNNwake.



(a) Time taken to compute wake fields for different number of turbines.



(b) Normalised FLORIS time on a logarithmic plot.

Figure 8: Time taken in seconds to compute wake field and power output for different number of turbines, turbines were placed in a row 300 meters apart. Plot b shows the normalised FLORIS time (FLORIS time divided by 100) in a logarithmic plot.

The logarithmic plot shows different gradients for the three methods used, which determines how the three models scale with an increasing number of turbines. CNNwake CPU is of order $N^{1.24}$ where N is the number of turbines, CNNwake GPU is $N^{1.45}$ and FLORIS scales in order of $N^{1.19}$. This means that CNNwake scales worse with array size than FLORIS. Surprisingly, the GPU scales worse than the CPU. This might be because device conversion (moving data between GPU and CPU) is slow and doing it many times for large wind parks might eventually outweigh the speedup from ruining the model on a GPU.

5.5 Yaw angle optimization

CNNwake can be used for wake steering and yaw angle optimization since it can accurately predict the power output of wind farms. Using the SLSQP algorithm has proven to be substantially faster for CNNwake and FLORIS than other SciPy algorithms such as COBYLA (Powell 1994, 1998), trust-region method (Lalee et al. 1998) or genetic algorithm (Storn & Price 1997).
CNNwake wake steering capabilities were tested on two cases, a 3 x 3 wind farm (case A) and a dense wind farm of 12 turbines (case B). Figure 9 shows the wind farm layout of the two cases.

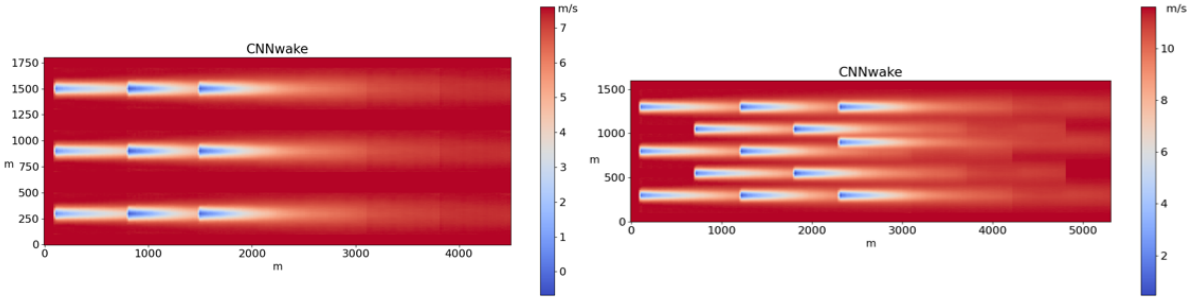


Figure 9: Layout of the two test cases, left side: Case A, u = 7.6 m/s, TI = 6%, FLORIS power output = 6.8 MW, right side: Case B, u = 11.6 m/s, TI = 7%, FLORIS power output = 47.1 MW.

Table 1 shows an overview of the optimization results. CNNwake was able to perform yaw angle optimization and increased the power output by a minimum of 21% for case A and 7% for case B. However, optimising the yaw angles using FLORIS reached a slightly better maximum power increase of 23% and 8% respectively. CNNwake is unable to learn all the three-dimensional flow interactions and power predictions done by FLORIS which results in a slightly worse yaw angle optimization. Regardless of this, CNNwake is able to reach a local maximum that produces only 5% less power than the maximum found by FLORIS while being approximately 60% faster. The speed is due to considerably fewer CNNwake function evaluations done by the solver. This might be because CNNwake is a lower fidelity model which means that additional small changes to the yaw angles, close to the optimal solution, do not lead to changes in the power prediction, as they do in FLORIS, which would make the optimiser reach the tolerance faster. On 70 randomly selected TI and wind speed inflow conditions CNNwakes's optimised power output, calculated using FLORIS with CNNwake's yaw angle prediction, for both cases was on average 4.6% lower than FLORIS's optimised output.

	Case A			Case B		
Model	optimized yaw angle [°]	optimized power output [MW]	Time taken [s]	optimized yaw angle [°]	optimized power output [MW]	Time taken [s]
FLORIS	28, 30, 30 23.5, 23, 23, 0, 0, 0	8.4	21.3 (244 function evaluations)	24, 24, 24 20.5, 20.5 17, 5, 17.5 0,-1, 0, 0, 0	51.0	113.6 (833 function evaluations)
CNNwake GPU	29, 30, 30 28, 29, 28 0, -1, -0.5	8.6 (FLORIS Power for same yaw settings: 8.3)	14.4 (191 function evaluations)	22, 21, 26 20, -25 15, 6, 15 1,-4, 0, 1, 0	50.4 (FLORIS Power for same yaw settings: 50.8)	62.1 (621 function evaluations)
CNNwake CPU	30, 30, 30 30, 30, 30 -2, -2, -2	8.45 (FLORIS Power for same yaw setting: 8.24)	19.8 (116 function evaluations)	20, 21, 21 20, -23 15, 6, -15 0.5, -4, 0, 0, 0	50.2 (FLORIS Power for same yaw settings: 50.7)	79.6 (345 function evaluations)

Table 1: Results of optimization for the two test cases. The optimization was run with a relative tolerance of 10^{-7}, a starting yaw angle of 0° for all turbines and timings were averaged over 3 runs. One function evaluation means a single wind park power output calculation for a specific yaw angle.

CNNwake yaw angle optimization can either run on a CPU or GPU for the NN calculations. Section 5.4 has shown that running CNNwake on a CPU is slower than running it on a GPU but the flow field, power and TI predictions have been the same regardless of the device it is run on. Surprisingly, the optimiser found different optimal yaw angles and used different number of function evaluations for running CNNwake on a GPU or CPU. This might be due to different rounding errors when using the CPU or GPU which can lead to inconsistent results, however, more work needs to be done to investigate this.

6 Conclusion and future work

In this work, the first model-based wake steering algorithm fully reliant on Neural Networks was developed and tested. The algorithm has three main components, the first one being a novel NN with fully connected and deconvolutional layers to generate the two-dimensional wake of individual turbines with a pixel-wise accuracy of 99.2%. It was shown that the network learned to reproduce the effect of the average inflow velocity, turbulent intensity and yaw angle on the wake with a maximum error of 5% in the far wake field. While a hyperparameter search was done, further investigation into the effect of different model parameters on the accuracy might deliver even better results. Additionally, instead of providing the network with the average wind speed upstream, the wind speed along a horizontal line would provide more information about the inflow condition which could lead to a reduction of error produced by uneven inflow conditions as described in section 5.2. Furthermore, the NNs can be trained on high fidelity CFD data generated using the FLORIS Curl model, RANS or LES which would further increase the accuracy of the NN's prediction, with little additional computational cost. Using a wake superposition model, the individual turbine wakes were combined to represent the flow field in a wind park with a downstream maximum error of 19% and average errors of less than 5%. CNNwake was about 18% faster than FLORIS for parks with less than 25 turbines. However, CNNwake scales worse than FLORIS due to a slower implementation of the wake superposition algorithm.

Secondly, the power and local TI predictions are made by two NNs trained on FLORIS data. This is the first time that NNs were used to predict the power output and local TI conditions of wind turbines from the velocity line upstream of the rotor. Due to time constraints, the network architecture for both networks was chosen to be the same. Two separate network architectures will give better results and further reduce the error in flow field and power prediction. The two NNs were trained on four example wind farms but they were able to accurately predict the power and local TI in unseen wind farm configurations. Using these pre-trained networks and fine-tuning them on specific wind park layouts via transfer learning might drastically improve their accuracy. This work has shown that NN can be used to approximate complex functions such as power and TI calculations from two-dimensional flow data. This idea can be extended to other functions such as turbine force/momentum calculations which can then be used for multi-objective optimization.

Lastly, the yaw angle optimization was validated on two test cases which have shown CNNwake's ability to improve the power output of wind farms while being significantly faster than FLORIS. This speedup is mostly due to fewer function evaluations done by the optimizer which might indicate that CNNwake's power output estimate is less sensitive to small changes in yaw angle. However, the results have reached yaw angles that produce about 5% less power than the optimal solution found by FLORIS. This demonstrates that CNNwake can find a good solution in less time.

CNNwake is able to predict flow fields, turbine power output, local TI and optimal yaw angles to a low error when compared to FLORIS. Even when compared to a cheap analytical model like the Gaussian model, CNNwake is still faster in most cases but it is of lower fidelity compared to FLORIS. This is to be expected as NNs will always be of lower fidelity than the model it was trained on. Overall, the CNNwake package can deliver good results at low computational cost.

References

Abdullah, M., Yatim, A., Tan, C. & Saidur, R. (2012), 'A review of maximum power point tracking algorithms for wind energy systems', *Renewable and Sustainable Energy Reviews* **16**(5), 3220–3227.

Argyle, P., Watson, S., Montavon, C., Jones, I. & Smith, M. (2018), 'Modelling turbulence intensity within a large offshore wind farm', *Wind Energy* **21**(12), 1329–1343.

Barthelmie, R. J., Hansen, K., Frandsen, S. T., Rathmann, O., Schepers, J. G., Schlez, W., Phillips, J., Rados, K., Zervos, A. & Politis, E. S. e. a. (2009), 'Modelling and measuring flow and wind turbine wakes in large wind farms offshore', *Wind Energy* **12**(5), 431–444.

Bastankhah, M. & Porté-Agel, F. (2014), 'A new analytical model for wind-turbine wakes', *Renewable Energy* **70**, 116–123.

Bay, C., King, J. & Fleming, P. (2019), *5th Wind Energy Systems Engineering Workshop*, NRLE.

Bensason, D., Simley, E., Roberts, O., Fleming, P., Debnath, M., King, J., Bay, C. & Mudafort, R. (2021), 'Evaluation of the potential for wake steering for u.s. land-based wind power plants', *Journal of Renewable and Sustainable Energy* **13**(3), 033303.

Boggs, P. T. & Tolle, J. W. (1995), 'Sequential quadratic programming', *Acta Numerica* **4**, 1–51.

Cioffi, A., Muscari, C., Schito, P. & Zasso, A. (2020), 'A steady-state wind farm wake model implemented in openfast', *Energies* **13**(23), 6158.

Department for Business, E. . I. S. (2021), *Energy Trends*.

Dong, H., Zhang, J. & Zhao, X. (2021), 'Intelligent wind farm control via deep reinforcement learning and high-fidelity simulations', *Applied Energy* **292**, 116928.

Donglin, C., Gao, X., Xu, C., Chen, S., Fang, J. & Wang, Z. (2020), Flowgan: A conditional generative adversarial network for flow prediction in various conditions.

Dou, B., Qu, T., Lei, L. & Zeng, P. (2020), 'Optimization of wind turbine yaw angles in a wind farm using a three-dimensional yawed wake model', *Energy* **209**, 118415.

Feng, X. & Den, X. (2020), 'Prediction of wake effect in wind farm using machine learning', *Advances in Civil, Environmental, Materials Research* .

Fischetti, M. & Fraccaro, M. (2019), 'Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks', *Computers Operations Research* **106**, 289–297.

Gao, X., Li, Y., Zhao, F. & Sun, H. (2020), 'Comparisons of the accuracy of different wake models in wind farm layout optimization', *Energy Exploration Exploitation* **38**(5), 1725–1741.

Glorot, X., Bordes, A. & Bengio, Y. (2011), Deep sparse rectifier neural networks, *in* 'Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics', pp. 315–323.

Goodman, G. & Martin, V. (2020), *Wind powered electricity in the UK*, Department for Business, Energy Industrial Strategy.

Gunn, K., Stock-Williams, C., Burke, M., Willden, R., Vogel, C., Hunter, W., Stallard, T., Robinson, N. & Schmidt, S. R. (2016), 'Limitations to the validity of single wake superposition in wind farm yield assessment', *Journal of Physics: Conference Series* **749**, 012003.
URL: *https://doi.org/10.1088/1742-6596/749/1/012003*

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. & Oliphant, T. E. (2020), 'Array programming with NumPy', *Nature* **585**(7825), 357–362.
URL: *https://doi.org/10.1038/s41586-020-2649-2*

Hunter, J. D. (2007), 'Matplotlib: A 2d graphics environment', *Computing in Science & Engineering* **9**(3), 90–95.

IEA (2020), *Global Energy CO2 Status Report 2019, IEA*.
URL: *https://www.iea.org/reports/global-energy-co2-status-report-2020*

Inoue, A., Hasan Ali, M., Takahashi, R., Murata, T., Tamura, J., Kimura, M., Futami, M., Ichinose, M. & Ide, K. (2005), A calculation method of the total efficiency of wind generator, *in* '2005 International Conference on Power Electronics and Drives Systems', Vol. 2, pp. 1595–1600.

IRENA (2020), *Renewable energy and climate pledges: Five years after the Paris Agreement*.
URL: *https://www.irena.org/-/media/Files/IRENA/Agency/Publication/2020/Dec/IRENA*$_N DC_u pdate_2 020.pdf$

Jacobson, M. Z. & Delucchi, M. A. (2009), 'A path to sustainable energy by 2030', *Scientific American* **301**(5), 58–65.

Kim, B., Azevedo, V. C., Thuerey, N., Kim, T., Gross, M. & Solenthaler, B. (2019), 'Deep fluids: A generative network for parameterized fluid simulations', *Computer Graphics Forum* **38**(2), 59–70.
URL: *http://dx.doi.org/10.1111/cgf.13619*

Kingma, D. P. & Ba, J. (2017), 'Adam: A method for stochastic optimization'.

Kraft, D. (1988), *A software package for sequential quadratic programming*, Tech. Rep. DFVLR-FB 88-28, DLR German Aerospace Center – Institute for Flight Mechanics.

Lalee, M., Nocedal, J. & Plantenga, T. (1998), 'On the implementation of an algorithm for large-scale equality constrained optimization', *SIAM Journal on Optimization* **8**(3), 682–706.

Lundquist, J. K., DuVivier, K. K., Kaffine, D. & Tomaszewski, J. M. (2018), 'Costs and consequences of wind turbine wake effects arising from uncoordinated wind energy development', *Nature Energy* **4**(1), 26–34.

Marek, P., Grey, T. & Hay, A. (2016), *A study of varaition in offshore turbulence intensity around the British Isles*.

Martínez-Tossas, L. A., King, J., Quon, E., Bay, C. J., Mudafort, R., Hamilton, N., Howland, M. F. & Fleming, P. A. (2021), 'The curled wake model: a three-dimensional and extremely fast steady-state wake solver for wind plant flows', *Wind Energy Science* **6**(2), 555–570.

Meyers, J. & Meneveau, C. (2012), 'Optimal turbine spacing in fully developed wind farm boundary layers', *Wind Energy* **15**, 305 – 317.

Mirza, M. & Osindero, S. (2014), 'Conditional generative adversarial nets'.

NREL (2020), 'FLORIS. Version 2.2.0'.
URL: *https://github.com/NREL/floris*

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. & Lerer, A. (2017), Automatic differentiation in pytorch, *in* 'NIPS-W'.

Powell, M. J. D. (1994), 'A direct search optimization method that models the objective and constraint functions by linear interpolation', *Advances in Optimization and Numerical Analysis* pp. 51–67.

Powell, M. J. D. (1998), 'Direct search algorithms for optimization calculations', *Acta Numerica* **7**, 287–336.

PyTorch (2019), 'Convtranspose2d — pytorch 1.9.0 documentation'.
URL: *https://pytorch.org/docs/stable/generated/torch.nn.ConvTranspose2d.html*

Riva, M. (2021), 'Batch normalization in convolutional neural networks'.
URL: *https://www.baeldung.com/cs/batch-normalization-cnn*

SciPy v1.7.0 Manual (n.d.), 'minimize(method='slsqp') — scipy v1.7.0 manual'.
URL: *https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html*

Simley, E., Fleming, P. & King, J. (2020), 'Design and analysis of a wake steering controller with wind direction variability', *Wind Energy Science* **5**(2), 451–468.

Storn, R. & Price, K. (1997), *Journal of Global Optimization* **11**(4), 341–359.

Tabib, M., Rasheed, A. & Kvamsdal, T. (2015), 'Les and rans simulation of onshore bessaker wind farm: analysing terrain and wake effects on wind farm performance', *Journal of Physics: Conference Series* **625**, 012032.

Ti, Z., Deng, X. W. & Yang, H. (2020), 'Wake modeling of wind turbines using machine learning', *Applied Energy* **257**, 114025.

Tomin, N., Kurbatsky, V. & Gulyvec, H. (2020), 'Intelligent control of a wind turbine based on reinforcement learning', *2019 16th Conference on Electrical Machines, Drives and Power Systems (ELMA)* .

United Nations/Framework Convention on Climate Change (2015), *Adoption of the Paris Agreement*.

Van Dijk, M. T. (2020), A Study on Yaw-Misaligment:Combined Optimization ofWind Farm Power Productionand Structural Loading, PhD thesis, Delfi University.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P. & SciPy 1.0 Contributors (2020), 'SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python', *Nature Methods* **17**, 261–272.

Vogel, C. & Willden, R. (2020), 'Investigation of wind turbine wake superposition models using reynolds-averaged navier-stokes simulations', *Wind Energy* **23**.

Wu, H., Liu, X., An, W., Chen, S. & Lyu, H. (2020), 'A deep learning approach for efficiently and accurately evaluating the flow field of supercritical airfoils', *Computers Fluids* **198**, 104393.

Świątek, B. & Dutka, M. (2015), 'Wind power prediction for onshore wind farms using neural networks', *Renewable Energy and Power Quality Journal* pp. 333–338.

A Gaussian model

The Gaussian analytical wake model is derived from "applying conservation of mass and momentum and assuming a Gaussian distribution for the velocity deficit in the wake" (Bastankhah & Porté-Agel 2014). The three dimensional normalized velocity deficit of a turbine wake is calculated using the following equation (Bastankhah & Porté-Agel 2014):

$$\frac{\Delta u(x,y,z)}{u_0} = \left(1 - \sqrt{1 - \frac{C_t}{8K}}\right) \exp\left(-\frac{1}{2K}\left(P^2 + L^2\right)\right) \tag{6}$$

where $\Delta u(x,y,z)$ is the wind speed wake deficit and x, y, z are the horizontal, span-wise and vertical coordinates, u_0 is the free stream velocity and C_T is the thrust coefficient.

$$K = (k^* \frac{x}{d_0} + \epsilon)^2 \tag{7}$$

where k^* defined the growth of the wake which is determined from experimental and LES data and varies with surface roughness and TI, d_0 is the rotor diameter and ϵ is the mass flow deficit rate at the rotor.

$$L = \frac{y}{d_0} \tag{8}$$

$$P = \frac{z - z_h}{d_0} \tag{9}$$

where z_h is the hub height.

B Wind park layouts for FCNN training

The FCNNs were trained on data that was extracted from four example wind parks. Figure 10 shows the layout of these wind parks with all yaw angle set to zero.

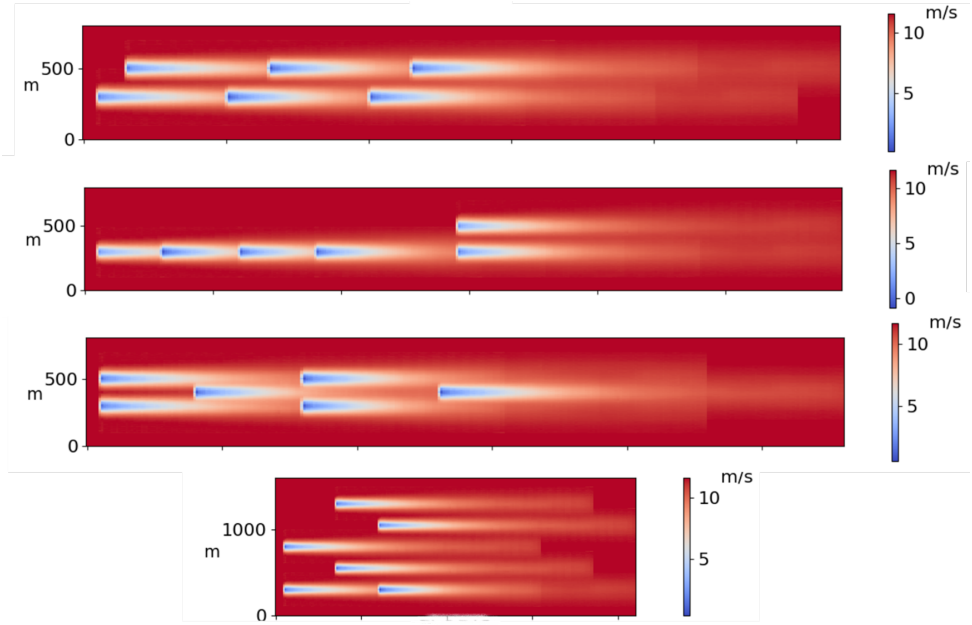


Figure 10: The four wind parks used to generate training data for the FCNN training