Offshore wind farm wake modelling using deep feed forward neural networks for active yaw control and layout optimisation

Sokratis Anagnostopoulos

Supervisor: Matthew Piggott

Github: https://github.com/acse-2019/irp-acse-sa1619

Submitted for the degree of

MSc Applied Computational Since and Engineering

Imperial College London

Abstract

Offshore wind farm modelling has been an area of increasing interest over the last two decades, with numerous remarkable analytical as well as computational approaches attempting to accurately describe the complex wind flows taking place. The ultimate aim is the development of tools that can be used to provide designs that increase the efficiency in power production. This work presents an open-source Machine Learning (ML) framework for the rapid modelling of wind farm flow field, using a Deep Feed Forward (DFF) neural network architecture, trained on approximate turbine wake fields, as calculated by the state-of-the-art wind farm modelling software FLORIS. The constructed neural model is capable of accurately reproducing the single wake deficits on the hub level of a 5MW wind turbine under yaw and a wide range of inlet hub speed and turbulence intensity conditions, at least an order of magnitude faster than the analytical wake based solution method, yielding results of 98.5% mean accuracy. A superposition algorithm is also developed to construct the wind farm domain of superimposed individual wakes. The ability of the trained neural network in providing reliable results is further demonstrated by performing wind farm yaw and layout optimisations, where the DFF produces optimal solutions that yield power gains of at least 70% while being an order of magnitude faster than the same optimisation carried out by FLORIS, across a wide range of wind conditions. A promising advantage of the present approach is that its performance and accuracy is expected to increase even further when trained on high-fidelity CFD data through transfer learning, while its computational cost for evaluation is kept low and constant.

Keywords: Fluid Dynamics, Machine Learning, Offshore Wind Turbines, Deep Neural Networks, Yaw Optimisation, Wind Farm Layout Optimisation

Acknowledgements

I would like to express my sincere gratitude for my supervisor Professor Mathew Piggott for his consistent input and support throughout this project and for proofreading the present report. I also want to thank Davor Dundovic for spending a lot of his time in discussing with me on the functionalities of FLORIS and on general software debugging.

Contents

List of abbreviations

ws	Wind speed at hub inlet
ti	Wind turbulence intensity
yw	Turbine yaw angle
ML	Machine Learning
DFF	Deep Feed Forward
CFD	Computational Fluid Mechanics
LES	Large Eddy Simulation
RANS	Reynolds-averaged Navier-Stokes
ANN	Artificial Neural Network
DNN	Deep Neural Network
CNN	Convolutional Neural Network
SGD	Stochastic Gradient Descent
Rprop	Resilient Backpropagation
MSE	Mean Squared Error
GS	Geometric Sum
LS	Linear Superposition
SOS	Sum of Squares
AEP	Annual energy production
WFLOP	Wind Farm Layout Optimization Problem
GAs	Genetic Algorithms
SLSQP	Sequential Least Squares method
CPU	Central Processing Unit
GPU	Graphics Processing Unit
RAM	Random Access Memory

1. Introduction

Wind power has maintained a significant share of the total worldwide electrical energy production of the 21st century (4.8% by the end of 2018 [1]) and is expected to grow up to 18% within the next 3 decades [2]. Wind energy production has increased over the past few years due to extensive efforts towards the modelling of turbine wakes and wind farm configurations that include a variety of analytical, experimental as well as high-fidelity numerical approaches.

Analytical wake modelling involves analytical or semi-analytical techniques used to perturb background wake flow information, providing an estimate for the flow field in the presence of turbines. Analytical modelling has been present from the earliest stages of modern wind turbines and is still playing an important role in producing low-cost wake predictions of the wake deficit profile. Some of the most commonly used models such as Larsen [3], Jensen [4], Curl [5] and Gaussian [6] are usually incorporated in packages available in the industry (FLORIS, WasP, WindPro etc) and depending on the application, they can significantly reduce the complexity of parametric studies or turbine array set-ups. However, since they usually involve the use of highly simplified physical assumptions and are mainly focusing on averaged velocity profiles and not transient turbulent wakes, they do not constitute a method that produces results of high-accuracy [7], [8]. Furthermore, the analytical models rely on empirical constants which require fine-tuning through computationally-expensive CFD simulations. Although these models are not able to capture a detailed representation of the turbine velocity deficit, the trade-off between accuracy and low-computational time is very often made in support to high-fidelity CFD studies in optimisation problems.

During the last two decades, the exponential advancements in CPU as well as GPU architectures have led to a significant progress towards wake modelling using computational methods [9], [10]. Coupled numerical models such as large-eddy simulation (LES) and Reynolds-averaged Navier-Stokes (RANS) are now capable of describing steady as well as transient wake flows very accurately and provide a realistic illustration of wind-turbine interaction physics [11], [12]. Nevertheless, accurately representing the wind flow profile, especially through the turbine blades which are moving at very high rotational speeds, requires very fine mesh settings in order to capture the boundary layer properties. Even then, some of the most widely used models for turbulence like k-ε, tend to overestimate the turbulence viscocity [13] leading to discrepancies between the numerical results and experimental measurements [14]. Thus, CFD is rarely used as a standalone method, even less when dealing with array optimisation problems (e.g. using Adjoint methods) which require multiple high-computational-cost runs.

Through the recent increase of computational power, the application of Artificial/Deep Neural Networks (ANN, DNN) as well as Convolutional Neural Networks (CNN), has been successfully tested across a wide variety of scientific fields, including fluid mechanics [15], [16], [17]. On wind turbines, there have been some recent efforts towards modelling of source terms with an ANN [18] and correlating Reynolds stress anisotropy with strain [19]. An ANN was constructed in order to assess the performance of wind turbines using the power vs torque curves [20], [21]. Very recently, a simple ANN architecture was trained on a large high-fidelity dataset and correlated two inputs (inlet wind speed and turbulence intensity) to produce 3D wake profiles of wind turbines in a single row [22]. So far, in the limited available research,

neural networks appear to be very efficient in the challenging task of building relationships between inflow conditions, rotor specifications and fluid properties. Traditional methods face difficulties in producing fast results that at the same time are accurate enough to support parametric studies. A well-constructed and well-trained neural network could be deployed and provide reliable results within seconds in order to predict flow properties of a wind farm that would otherwise require orders of magnitude higher computational times. At the moment, the use of powerful machine learning and regression tools shows promising results and could pave the way to even more sophisticated optimisation approaches in the future.

The present study aims at constructing a DNN, capable of handling wind properties and wind turbine yaw settings, in order to demonstrate for the first time in the available literature, that active yaw control, as well as layout optimisations using machine learning tools for turbine wake modelling are feasible, at considerable computational time gains. Using the presented neural network framework could enable further accuracy and computational cost improvements in wind farm optimisations that could easily be implemented with transfer learning (using gained ML knowledge from one problem to a similar one), with more advanced wake datasets, produced by iterative analytical models and high-fidelity numerical simulations (CFD) for further training the DNN.

2. Software Description & Methodology

A Deep Feed Forward (DFF) Neural Network is built in order to reproduce the velocity domain of a single wind turbine wake, when given up to a triplet of inlet conditions: downstream velocity at the hub (inlet speed), turbulence intensity and yaw angle. The network is trained on wake profiles produced by FLORIS using the Gaussian analytical wake model, where the 3D velocity deficit behind each turbine is derived from the following simplified form of Navier-Stokes:

$$\frac{u(x,y,z)}{U_\infty} = 1 - Ce^{(y-\delta)^2/2\sigma_y^2} e^{-(z-z_h)^2/2\sigma_z^2} \tag{1}$$

$$C = 1 - \sqrt{1 - \frac{(\sigma_{y0}\sigma_{z0})M_0}{\sigma_y\sigma_z}} \tag{2}$$

$$M_0 = C_0(2 - C_0) \tag{3}$$

$$C_0 = 1 - \sqrt{1 - C_T} \tag{4}$$

where C is the velocity deficit at the centre of the wake, δ is the wake deflection, z_h is the hub height, C_T is the thrust coefficient and σ_y, σ_z the widths of the wake in the y and z direction, respectively. The "0" subscript denotes the values at the start of the far wake and the widths σ_y, σ_z are given by:

$$\frac{\sigma_y}{D} = k_y \frac{x - x_0}{D} + \frac{\sigma_{y0}}{D} \tag{5}$$

$$\frac{\sigma_z}{D} = k_z \frac{x - x_0}{D} + \frac{\sigma_{z0}}{D} \tag{6}$$

where

$$\frac{\sigma_{y0}}{D} = k_y \frac{x - x_0}{D} + \frac{\sigma_{z0}}{D}\cos(\gamma) \tag{7}$$

$$\frac{\sigma_{z0}}{D} = \frac{1}{2}\sqrt{\frac{u_R}{u_\infty + u_0}} \tag{8}$$

where D is the rotor diameter, γ is the yaw angle and k_y, k_z define the wake expansion in the lateral and vertical directions, respectively.

2.1 Synthetic dataset method

The synthetic dataset was comprised of 4000 wake images (Fig. 1), with a batch size of 400 wakes, which was found to be sufficient in producing results of satisfactory accuracy (Fig. 4) for the purposes of the present study, and was applied on a 5 MW wind turbine of 126m in diameter. A batch size of 10% of the complete dataset lies within the commonly adopted range in the literature [23] and was proven to be effective in reducing overfitting of the model to specific wake inputs, while also reducing the training computational cost.

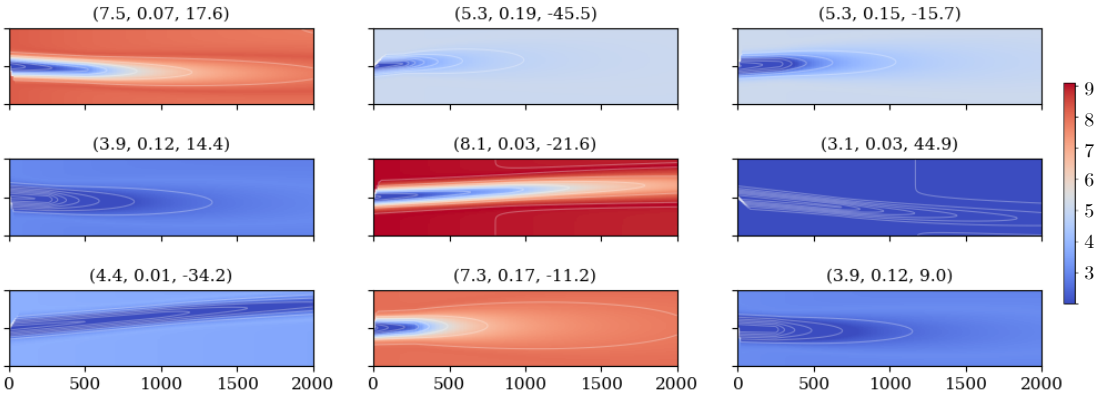


Figure 1. *Indicative wake dataset sample where each title denotes a wind speed (ws), turbulence intensity (ti), yaw angle (yw) triplet.*

The ranges of the inlet speed (ws), turbulence intensity (ti) and yaw angle (yw) are [3, 12] m/s, [0.01, 0.2] and [-50, 50] degrees, respectively, while their values are produced randomly following a uniform distribution. The range of the inlet speeds is selected based on the power curve of the specific wind turbine (shown in Fig. 2), where the lower operation limit is 3 m/s and the upper limit is 20 m/s. However, right after the point where the Cp value becomes constant (at 12 m/s), there is usually little potential gain from yawing the hub [24], thus the selected upper limit for the training was set at 12 m/s, which significantly reduced the size of the required synthesised dataset. The range of the turbulence intensity is selected based on commonly reported measurements [25]. It is worth mentioning that the range, as well as the distribution of the dataset, could easily be adjusted to follow a non-uniform distribution based on available weather data, in order to further improve the capabilities of the DNN on site-specific weather conditions.

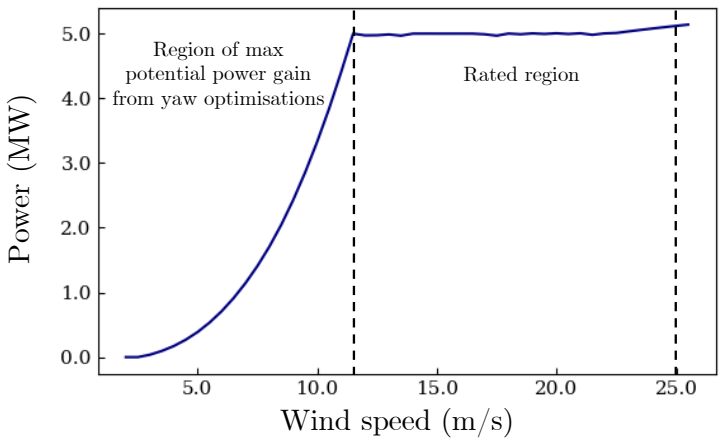


Figure 2. *Power curve of the 5MW wind turbine of the study. The rated region of the turbine is the "plateau" of potential power output and begins after about 12 m/s.*

2.2 DNN architecture

The architecture of the DNN is shown in Figure 3, which consists of two fully connected hidden layers of 100 neurons each, that output an m by n grid of points that represent the downstream velocity domain of the wake. It has been shown that re-normalising the training batch after each hidden layer of a DNN significantly increases the performance of the trained network [26]. Thus, two batch normalisations have been applied after the first two layers, which as expected, increased the accuracy of the resulting wake. Several activation functions were tested for their performance (sigmoid, tanh, tansig, purelin), as well as various normalisation methods (min-max, mean/std, -1/1) for the input parameters. The selected activation functions which performed best and thus is selected for this study are the tanh for the first two layers and a linear activation function for the output along with mean/std normalisation.

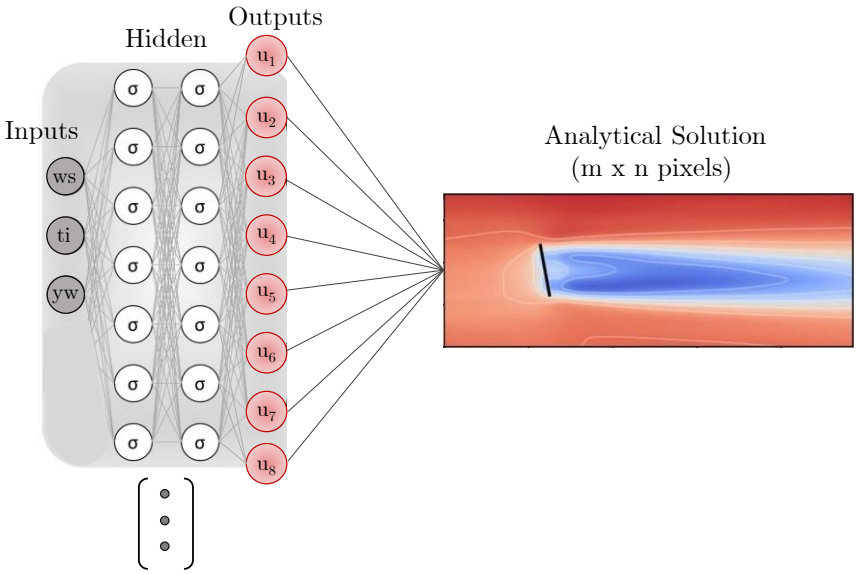


Figure 3. *DNN architecture. The dots represent the optional functionality of independent parallel sub-networks, each of which is trained on a partition of the domain.*

The default resolution of FLORIS is a 200 x 200 grid which is also the default adopted

resolution for the training of the DNN. However, for higher resolution demands, the user can specify the desired resolution before training the model which will also produce an analytical wake dataset of that same resolution. To tackle arising memory problems for models trained on higher resolution images, a parallel sub-network functionality has been implemented (Fig. 3). These sub-networks are deployed in parallel instead of the main network and are trained independently on a portion (piece) of the domain, specified by the user. Furthermore, this functionality allows the user to specify if each independent "piece" of the domain will be row, column or block-wise. The user can also select if these individual pieces are to be sent to the GPU, if Cuda is available on the system, or multiple CPU cores. It is worth mentioning that the resulting weights are saved as a single .pth file, which facilitates the usability of the code.

2.3 Hyper-parameters & training

The model hyperparameters (Table 1) were calibrated through iterative process by evaluating the metrics of the loss and absolute accuracy of the validation set (Figs 4b, 4d). A common practice when using large datasets is to create a validation test that constitutes 10-20% of the initially created dataset [27], which does not contribute in any backpropagation step during the training process and is only used to assess the current network's performance on previously unseen data. The training dataset, is the remaining proportion of the synthesised FLORIS images (90%), and is compared with the DNN's output images by applying the Root Mean Square Error (RMSE) between the wake velocities at each cell of the computational grid, which is given by:

$$RMSE_{fo} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(z_{fi} - z_{oi})^2} \tag{9}$$

where z_{fi}, z_{oi} are the DNN forecasts and analytical observed values, respectively and N is the sample size. Note that for the final training, the whole dataset (4000 wake images) is used as the training dataset.

Several optimisers were tested for their performance including Stochatic Gradient Descent (SGD), RMSprop, Adam and Resilient Back Propagation (Rprop). The latter, qualified as the fastest converging optimizer for this study and was one of the most significant improvements on early vanilla versions of the DNN's development, resulting in a ~97% accuracy on both training and validation datasets, as shown in Figure 4. Rprop [28] is a gradient based method which introduces an additional factor η, which multiplies the weight correction Δ_{ij}, depending on the proximity to a local minimum of the error plane. The main learning rule is given by:

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^+ \cdot \Delta_{ij}^{(t-1)} \ , \ if \ \ \frac{\partial E^{(t)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t-1)}}{\partial w_{ij}} > 0 \\ \eta^- \cdot \Delta_{ij}^{(t-1)} \ , \ if \ \ \frac{\partial E^{(t)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t-1)}}{\partial w_{ij}} < 0 \\ \Delta_{ij}^{(t-1)} \quad , \ else \end{cases} \tag{10}$$

where $0 < \eta^- < 1 < \eta^+$.

Whenever the sign of the partial derivative of the weight w_{ij} changes, this indicates that the

last update was too large, resulting in "jumping over" a local minimum. Thus, the weight correction is adjusted by a factor of η-, whereas if the sign of the weight partial derivative is positive it is multiplied by a factor of η+ in order to increase the convergence rate. The weights are then updated as:

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)} \ , \ if \ \ \frac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\ +\Delta_{ij}^{(t)} \ , \ if \ \ \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \\ 0 \quad\quad , \ else \end{cases} \tag{11}$$

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)} \tag{12}$$

where the adjusted correction $\eta \cdot \Delta_{ij}$ is added if the error derivative is positive or subtracted if the error derivative is negative. However, the update is reverted if the previous minimum was skipped:

$$\Delta w_{ij}^{(t)} = -\Delta w_{ij}^{(t-1)} \ , \ if \ \ \frac{\partial E^{(t)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t-1)}}{\partial w_{ij}} < 0 \tag{13}$$

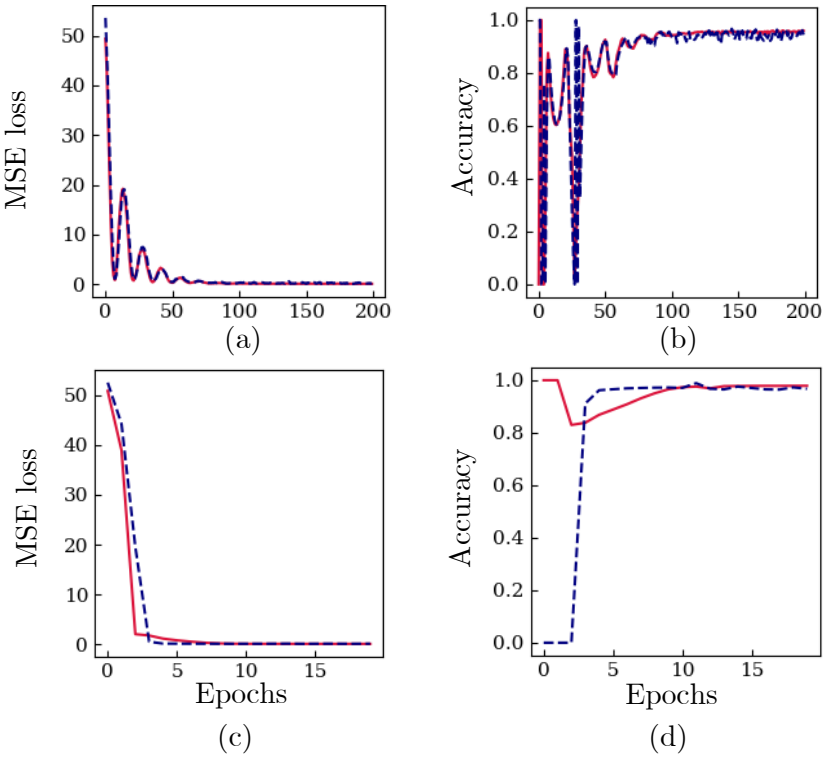


Figure 4. *Root mean square error (MSE) loss and accuracy convergence graphs of Stochastic Gradient Descent (SGD) (a, b) and Resilient Back Propagation (Rprop) (c, d), respectively.*

Table 1. *Model hyper-parameters, MSE loss, accuracy and timings for SGD and Rprop optimisers.*

	Hyper-parameters		**Loss**		**Accuracy**		**Time** (m)
	Learning rate	Momentum	Validation	Training	Validation	Training	
SGD	0.1	0.99	0.06	0.2	0.95	0.94	50
Rprop	0.01	-	0.035	0.05	0.98	0.97	15

2.4 Wake superposition

Most analytical wake models include an independent approach for the superposition of wakes in order to form an array of multiple wind turbines, as well as the interactions between them [29], [30]. Some of the most widely used models for the superposition are the following:

Geometric Sum (GS): $$\frac{u_i}{U_\infty} = \prod_{j=1}^{n} \frac{u_{ij}}{u_j} \quad (14)$$

Linear Superposition (LS): $$(1 - \frac{u_i}{U_\infty}) = \sum_{j=1}^{n} (1 - \frac{u_{ij}}{u_j}) \quad (15)$$

Sum of Squares (SOS): $$(1 - \frac{u_i}{U_\infty})^2 = \sum_{j=1}^{n} (1 - \frac{u_{ij}}{u_j})^2 \quad (16)$$

where u_i denotes the wind speed at turbine i, u_{ij} the velocity at turbine i influenced by the wake of turbine j and n is the number of turbines placed upstream.

For this study, a special superposition algorithm has been developed, which is deployed for the combination of multiple individual wakes produced by the DNN. A SOS model (also selected for the FLORIS simulations) has been adopted, which manipulates individual wakes of the neural network and is coupled with an approximate method of calculating a uniform velocity at the hub of each turbine. The final domain represents a collage of the individual wakes that comprise the examined offshore wind farm.

One of the advantages of the presented neural model is that it is trained on 2D slices of a 3D wake domain of FLORIS, which also implicitly take into account the vertical velocity boundary layer, as affected by the sea surface. This significantly contributes in faster computation times, while including information about the velocity boundary layer of the site. However, since only the velocities along the line of the hub are known, an approximate formula has to be applied in order to correct the average velocity taken at the hub of each turbine. As shown in Figure 5, the turbine hub can be divided in radial rings of equal thickness dr, which represents the length of one pixel of the computational grid.

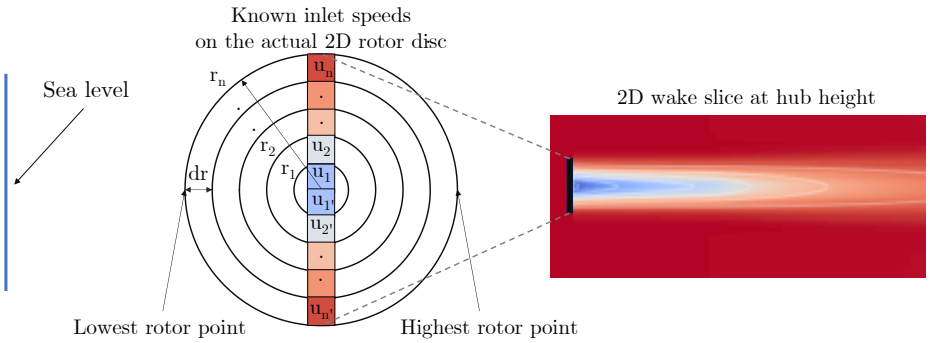


Figure 5. *2D representation of the 1D hub as calculated by the DNN.*

Assuming that the rotor lies within a linear region of the boundary layer, by integrating over the surface of the hub, the velocity on the 2D disc is given by:

$$u_{hub} = \int_{r_1}^{r_{n/2}} u_{mean} \cdot 2\pi r \cdot dr \tag{17}$$

where u_{mean} is the mean speed on each hub ring, which based on Figure 4 is given by:

$$u_{mean} = \frac{1}{2}(u_i + u_{i'}) = \frac{1}{2}(u_i + u_{n-i}) \tag{18}$$

The discretised form of equation (17) is then given by:

$$u_{hub} = \frac{1}{A}\sum_{i=1}^{n/2}(u_{mean} \cdot 2\pi r_n \cdot dr) \tag{19}$$

$$u_{hub} = \frac{dr}{{r_1}^2}\sum_{i=1}^{n/2}[(u_i + u_{n-i}) \cdot r_n] \tag{20}$$

where A is the rotor swept area.

However, for the calculation of power, the cubed mean velocity (kinetic energy) term is used:

$${u_{power}}^3 = \frac{1}{A}\sum_{i=1}^{n/2}({u_{mean}}^3 \cdot 2\pi r_n \cdot dr) \tag{21}$$

Thus, the power of each turbine is given by:

$$P = \frac{1}{2}Cp \cdot \rho \cdot \cos\theta \cdot {u_{power}}^3 \cdot A \tag{22}$$

where Cp is the turbine's power coefficient and ρ is the fluid density and θ is the yaw angle of the wind turbine. Therefore, for a farm of n turbines, the total generated power is given by:

$$P_{tot} = \sum_{i=1}^{n} P_i \tag{23}$$

2.5 Yaw and layout optimisations

Recent studies [24], [31] have shown that exploiting the active yaw control of wind farms using analytical models like FLORIS, can increase the annual energy production (AEP) of the wind plant and also reduce the thrust loads on the turbines. Fast active yaw optimisation using machine learning could be applied at a minimal additional cost, boosting the AEP even further since the reaction time of the turbine could be significantly decreased.

On the other hand, positioning the wind turbines such that the expected power production is maximised (Wind Farm Layout Optimization Problem, WFLOP [32]) is another complex problem that the existing available literature has very recently started attempting to solve. Analytical, as well as high-fidelity CFD simulations, have been previously used along with Genetic Algorithms (GAs) [33] and Adjoint methods [34], [35], [36] and although they represent a good starting point towards the WFLOP problem they usually constitute costly computational approaches. Moreover, they lack multi-objective optimisation aspects like the effect of turbulence intensity on the plant or other area-specific construction limitations [32].

On these grounds, two distinct optimisation modules are also developed in this work in order to assess the capability of the neural wake model in wind farm active yaw and array turbine placement optimisation problems. The optimisation functions are made from scratch using SciPy's SLSQP optimiser and attempt to optimise the total power output of the plant (Eq. 23) by handling the outputs produced by the DNN. The results of the optimisers are then compared with the corresponding optimised results of FLORIS, for both yaw and layout optimisation scenarios. The reliability and the accuracy of the modules is finally assessed for the whole operation range of the DNN.

2.6 Software description

One of the main objectives of this study is to develop a robust, open source ecosystem, the modules of which, can be further improved individually by contributors. The complete ecosystem of the standalone model is presented in Figure 6, where the inter-relations between each of the sub-modules are also shown. The individual components of the library achieve specific tasks in the process of wake modelling and are sub-divided as such, which facilitates their future expansion. For example, the NeuralWake model can be further trained using more high fidelity CFD wake data whereas different wake combination models could also be implemented in the Superposition module. In other words, the sustainability aim of the presented library is that any extensions or improvements to any of the sub-modules will also contribute directly to the final optimised output.

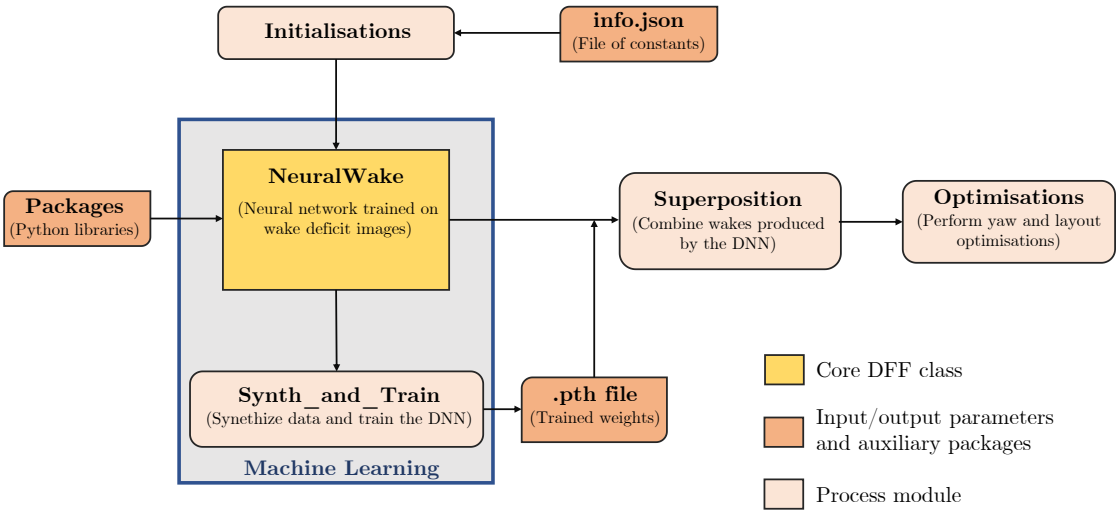


Figure 6. *Software architecture. A complete life cycle could be defined as follows: The parameters and boundaries of the model are initialised in the Initialisations module, which are passed into the core NeuralWake class. The training data is then synthesized in the Synth_and_Train module which also trains the weights of the NeuralWake model. The learned weights (.pth file) are then adopted in the forward propagation of the NeuralWake module which produces the individual wakes to be combined in the Superposition module. Finally, the Optimisations module attempts to optimise the total power output calculated in the Superposition module, for yaw and layout optimisation scenarios.*

2.7 Code metadata

All the modules of the software in this work were developed and tested within a Python3 environment. The machine was running on Ubuntu 20.04 with a 1.9 GHz 8th Gen i5 CPU and 20 GB of RAM. The suggested minimum RAM for the demonstrated tests and results is 8GB. The software can be run on either Linux or Windows. Although the presented software provides the functionality of parallel CPU cores or GPU computations, for the purposes of the study the network was trained with serial CPU computation, as it provided slightly faster results due to the relatively small scale of the computational grid of the presented cases (200 x 200). Some of the core libraries used in this study are listed below:

Numpy (v1.17.2): Very powerful for numerical array computations.

SciPy (v1.3.1): Includes a wide variety of optimisation modules.

Matplotlib (v3.1.1): Multi-platform data visualisation library.

Torch (v1.5.1): Powerful high-level ML library which handles n-dimensional tensors.

Dask (v2.9.0): Provides high-level parallel CPU computation functionalities.

FLORIS (v2.1.1): One the most widely used state-of-the-art software for wind farm modelling.

The complete library list can be found in the requirements.txt file on Github.

2.8 Novelties

More specifically, the novelties of the present work are listed below:

i. Adoption of a Deep Neural Network architecture (more than 1 hidden layers) for the modelling of yawed turbine wakes. While a simple ANN architecture was sufficient in reproducing the wakes of two inputs (inlet speed and turbulence intensity) [22], the addition of an extra hidden layer was proven to be necessary for the accurate representation of the wake envelope under yaw.

ii. Neural network trained on 2D instead of 3D computational domains. The output of the DNN is a 2D slice of the analytically computed grid, which however, includes information about the influence of the sea surface in the downstream velocity boundary layer. This results in significant further computational cost gains.

iii. Implementation of a special superposition method. For consistency, the method was partly based on the SOS model, which was the selected method in FLORIS, while also proposing the introduction of an approximate analytical method to obtain a representative average velocity on the hub of the turbine, which was found to satisfactorily compliment the 2D grid approach mentioned above.

iv. Flexibility of deployment depending on the application. The neural wake model comprises of a variety of options that can be adjusted including the dimensions and resolution of the wake domain, definition of up to 3 inputs (inlet wind speed, turbulence intensity, yaw angle), as well as the height of the domain (default is hub height). Furthermore, the network can be trained on a personal computer of limited available memory, using a parallel sub-network functionality, each of which specialises in a row/column/block-wise partition of the domain. In that case, a parallel CPU or GPU functionality can also be deployed.

v. Demonstration of the DNN's capabilities in optimisation problems. The presented model is capable of providing reliable optimised results, for both yaw and turbine positioning, at a significantly faster speed compared to the Gaussian analytical model used in training. The computational time gains achieved by the adopted methods, could pave the way to / enable the application of machine learning for fast active yaw optimisation of offshore wind farms, as well as compliment traditional methods in the optimal placement of wind turbines.

3. Implementation and Code results

To demonstrate the capabilities of the trained neural model, several indicative cases are examined, namely two cases (single and multiple wakes) for the assessment of the mean absolute error (%) between FLORIS and the DNN, two cases of multiple wakes to assess the correctness in the implementation of the SOS superposition model and three optimisation cases (A, B, C), the first two for the yaw and the latter for the optimisation of the farm layout. Note that for all optimizations, the power gain achieved by the DNN is the power gain as calculated by FLORIS but using the yaw (or layout) optimal solution obtained by the DNN. In other words, all power gain comparisons were performed using FLORIS, with the proposed optimal settings of either FLORIS or the DNN.

3.1 Single / multiple wake plots with errors and y-transects

Figures 7a-c and 8a-c show a single wake and a multiple wake scenario, respectively. For both cases, the resulting downstream velocity domain as calculated by FLORIS are shown in Figures 7a, 8a, whereas the domain produced by the DNN is shown in Figures 7b, 8b. The absolute relative error (%) between the analytical and neural results is shown in Figures 7c and 8c, and is given by:

$$|error_{\%}| = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{u_{fi}-u_{oi}}{u_{fmax}}\right| \cdot 100 \tag{24}$$

where u_{fi}, u_{oi} are the velocities calculated at each pixel i by FLORIS and by the DNN, respectively, u_{fmax} is the maximum velocity of the FLORIS domain and n is the total number of pixels (200x200 by default).

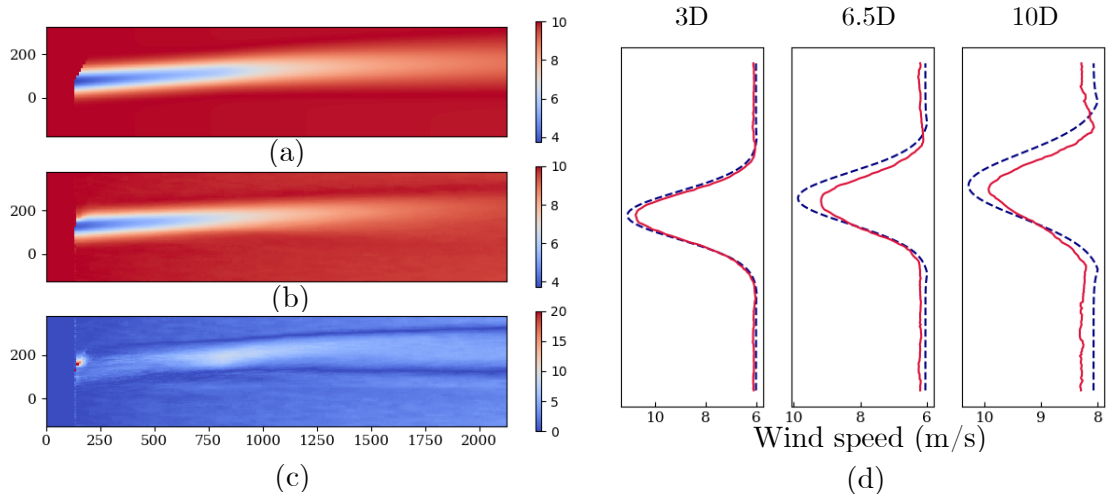


Figure 7. *Single wake produced by FLORIS (a), the DNN (b). Relative absolute error (%) between the two models (c) and y-transects at 3, 6.5 and 10D downstream (d). The inlet conditions are ws: 11 m/s, ti: 0.05, yw: 30º. DNN: red line, FLORIS: blue dashed line.*

For the single wake case (Fig. 7), the resulting DNN data appears to reproduce well the analytical wake deficit, with the highest error being at 10%, while the DNN velocity profiles at three y-transects along vertical cross-sections of the physical domain are in agreement with

the analytical ones in terms of the yawing angle, showing some minor deviation the further they are from the hub (Fig. 7d). In addition, in order to assess the accuracy in the predictions made by the DNN, the mean absolute error is calculated for a 400 different wakes of random inlet conditions as produced by the neural model and FLORIS, which was found to be at 1.67%, implying that the DNN has an accuracy of 98.3%.

The multiple wake case (Fig. 8) was selected in order to test the performance of the DNN, this time on a dense wind farm configuration with high turbine yaw settings. As expected, the absolute error increases (namely up to 20%) in certain regions of the wake velocity domain (Fig. 8c), since the superposition method is now also affecting the results. However, the mean absolute error is less than 10%, while the y-transects in Figure 7d agree well between the two models with only few exceptions, mainly in regions of multiple wake superposition.

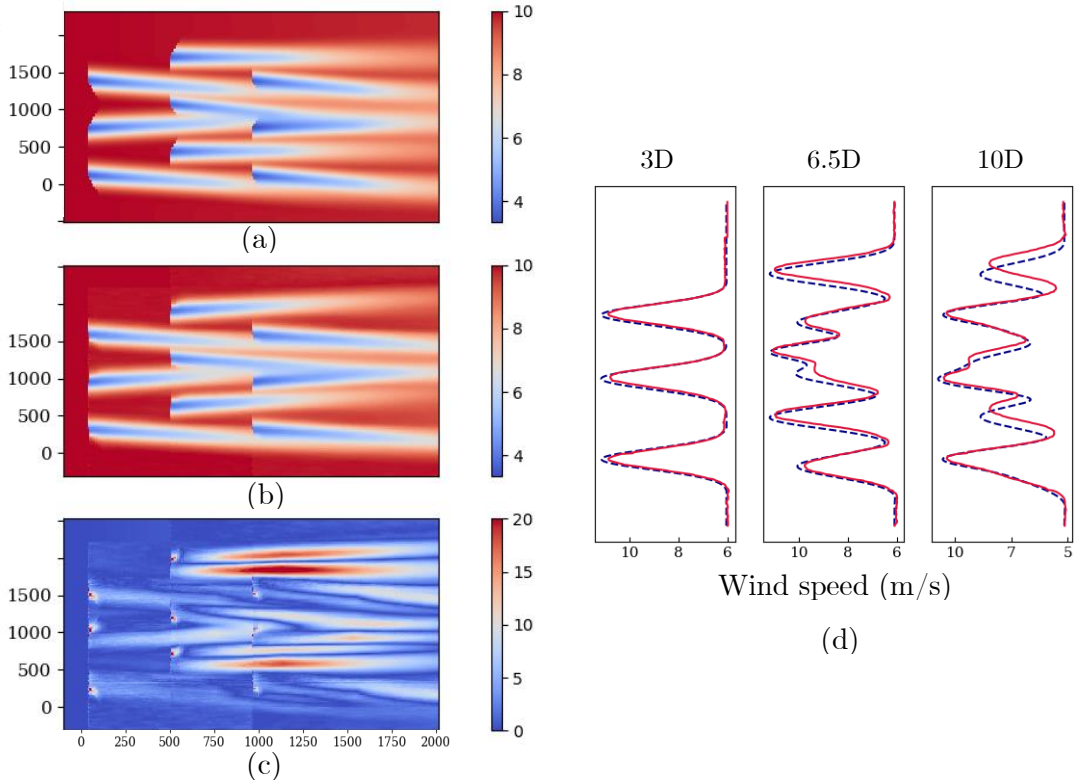


Figure 8. *Multiple interacting wakes produced by FLORIS (a), the DNN (b). Relative absolute error (%) between the two models (c) and y-transects at 3, 6.5 and 10D downstream (d). The inlet conditions are ws: 11 m/s, ti: 0.05, yw: [30, -30, 30, -30, 30, -30, 30, -30, 30]° (from left to right and bot to top). DNN: red line, FLORIS: blue dashed line.*

3.2 SOS superposition example-test cases

To demonstrate the effectiveness of the superposition model adopted in this work, two additional scenarios are tested, which show a 3x3 wind farm configuration with zero yaw (Fig. 9) and a theoretical 3-turbine scenario (Fig. 10). Although the latter constitutes an extreme layout setting, it provides useful information about the correctness in the application of the SOS model, which appears to produce a smoothened region between two or more turbine interactions.

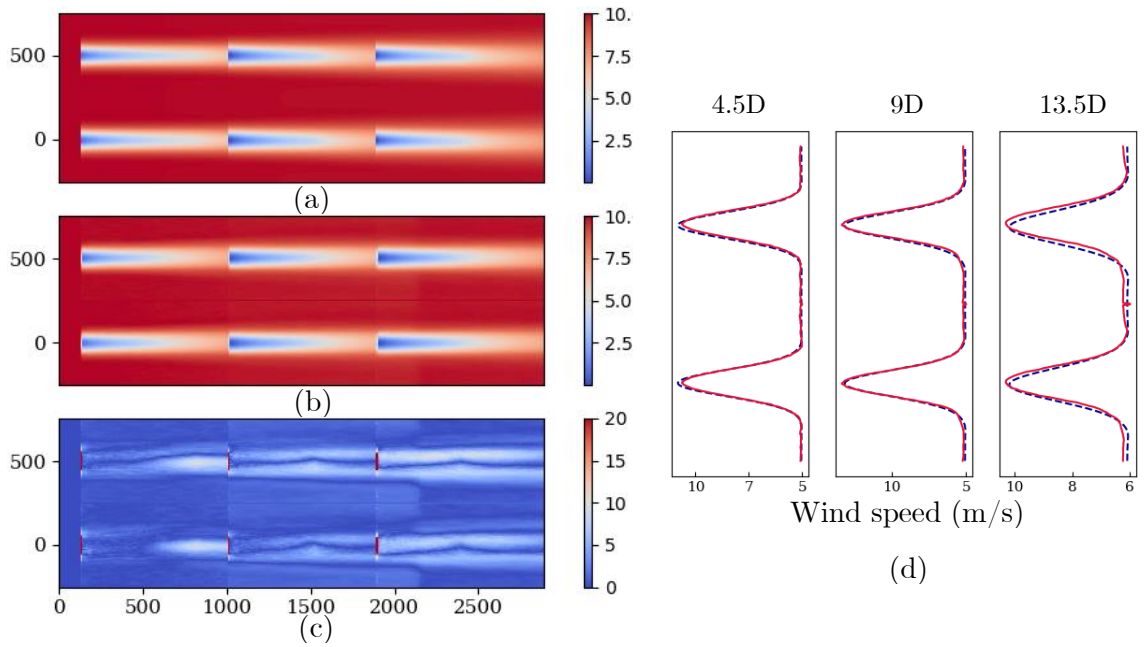


Figure 9. *Superposition test case: 2x3 wind farm array produced by FLORIS (a), the DNN (b). Relative absolute error (%) between the two (c) and y-transects at 4.5, 9 and 13.5D downstream (d). The inlet conditions are ws: 11 m/s, ti: 0.05, yw: 0°. DNN: red line, FLORIS: blue dashed line.*

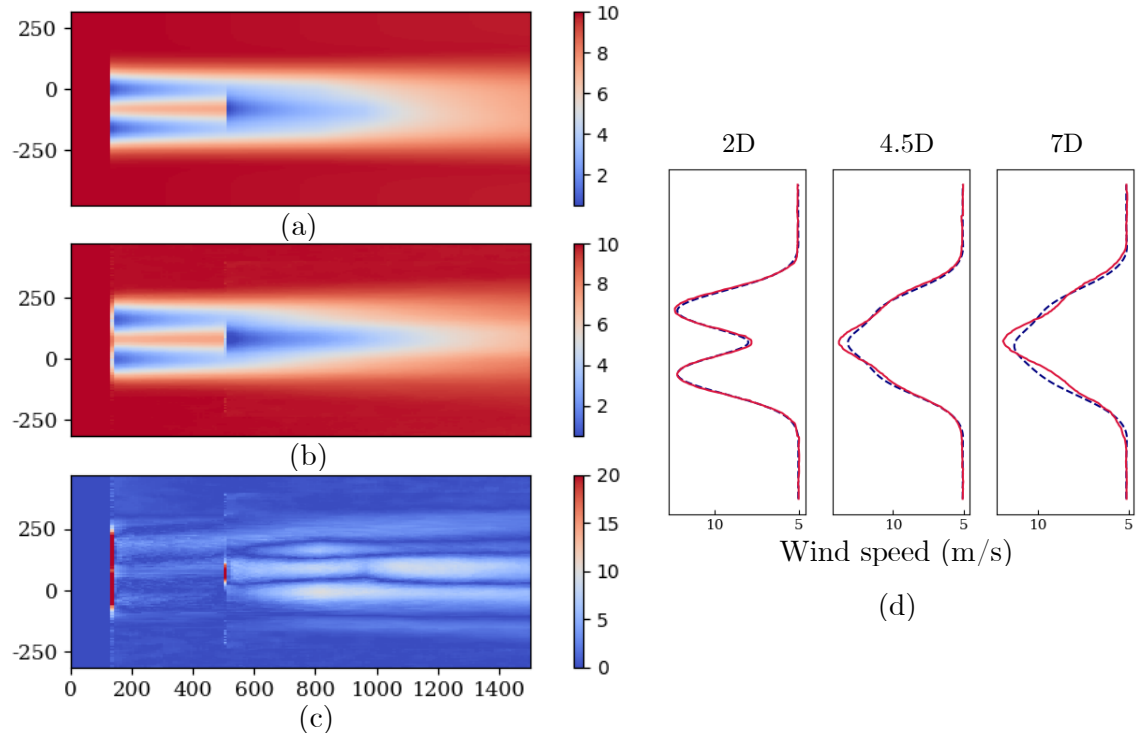


Figure 10. *Superposition test case: 2+1 wind farm array produced by FLORIS (a), the DNN (b), the absolute error (%) between the two (c) and y-transects at 2, 4.5 and 7D downstream. The inlet conditions are ws: 11 m/s, ti: 0.05, yw: 0°.*

In Figures 10a, 10b, the inlet velocities at the hub of the two middle and two last turbines of the DNN wake deficit appear to decline in a similar fashion with the FLORIS result, as also shown in the transect curves of Figure 10d. The absolute error between the two is about 4% (Fig. 10c).

3.3 Scalability for wind farm with absolute error

Regarding the computational cost, Figure 11a shows that the DNN outer-performs FLORIS after about 3 superimposed turbines reaching a difference of 1 order of magnitude in computational time at 20 turbines. Note that in Figure 11a, FLORIS CPU time has been divided by 100 because it calculates the full 3D domain which consists of a 200x200x100 grid. However, comparing the actual computation times (Fig. 11b) probably constitutes a more fair approach, since the proposed neural network method of this study results in very accurate optimisation results, comparable to those produced by FLORIS, as it will be discussed in the next section. In Figure 11b, the increase of computational cost with the number of turbines appears to be of 2nd order for Floris, and 1st order for the DNN, resulting in 3 orders of magnitude gain for an array of 20 turbines. Note that the computational time for each number of turbines is obtained by taking the average time of 5 iterations of the same superposition scenario.

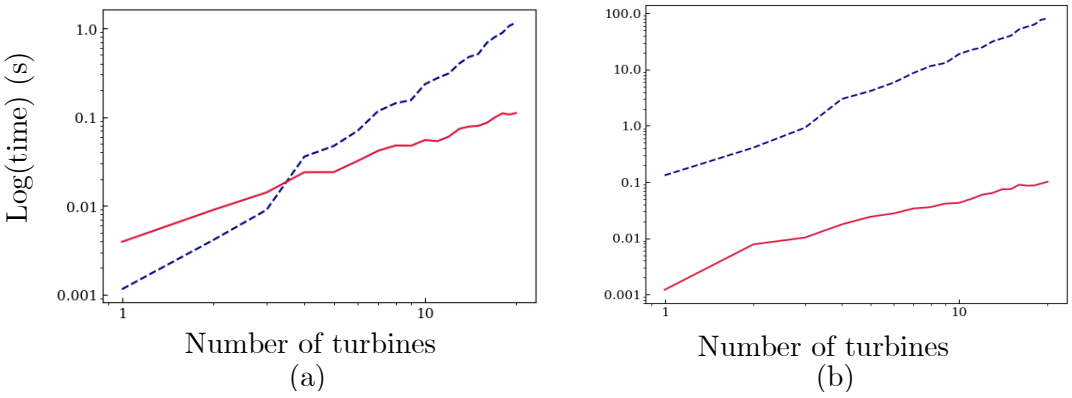


Figure 11. *Logarithmic plots of computational time scaling vs number of superimposed turbines for FLORIS (blue dashed), DNN (red). Actual times plot (a) and FLORIS time divided by 100 (b) for comparison of the time it takes for both models to calculate one 2D slice.*

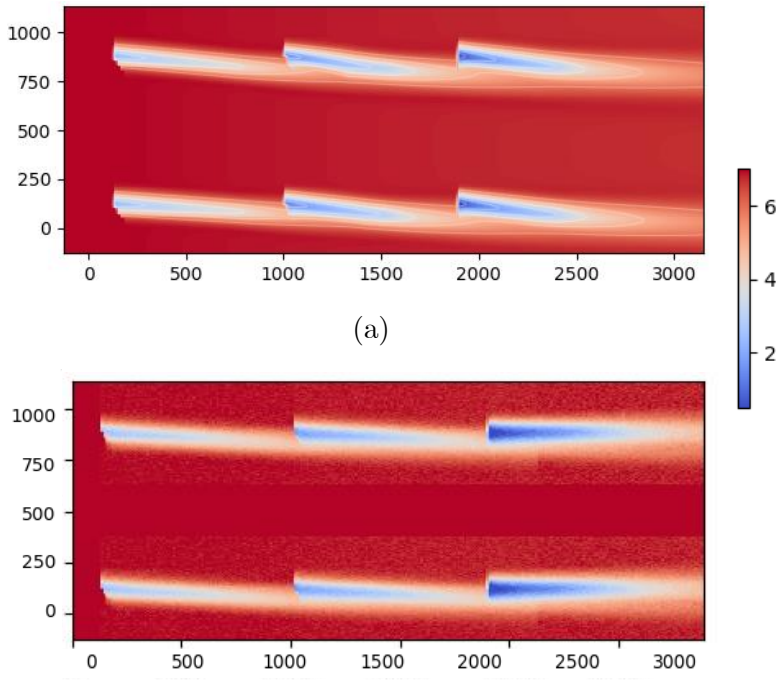
3.4 Yaw optimisation test cases

Yaw optimisation has recently become a key topic, especially in wind farm configurations where the total power output of the plant can be significantly boosted based on the yaw settings of the turbines [24]. Furthermore, active yaw optimisation is heavily dependent on fast predictions based on live weather forecasting [37], [38] as to take advantage of as much of the incoming wind energy as possible. This study attempts to demonstrate that even when compared to a relatively cheap wake model as the Gaussian, a DNN model can produce very good yaw setting predictions with significant computational cost gains, given that weather

conditions are known. Two indicative example cases are discussed in this section (A and B), involving a 6- and a 15-turbine wind farm configuration, respectively.

Case A

In general, compared to a greedy yaw setting where the turbines are facing the wind direction (zero yaw in the present case), deflecting the wakes of the front turbines can usually be regarded as a good yaw setting strategy for a set of turbines that cooperatively increase the overall performance of the farm [31]. In Figure 12a, FLORIS achieves a 25.8% power gain by setting a 29, 18, 0° of yaw on the three turbine pairs from front to back of the stream, respectively, after 15s. A very similar approach is adopted by the optimisation produced by the DNN, where a 22.8% power gain is achieved with a 26, 20, 0° corresponding setting. The optimisation required less than half the time (7s). Note that FLORIS has an additional functionality where the direction of the wind at the inlet of turbines further downstream, is affected by the yaw of the turbines in front, which is not yet implemented in the neural model. This could be the reason why in general, the wakes produced by FLORIS show an increased yaw angle compared to the DNN, even though both models have similar yaw settings. Note that for all yaw optimisations, the initial yaw angles are at 0 degrees.



(a)

(b)

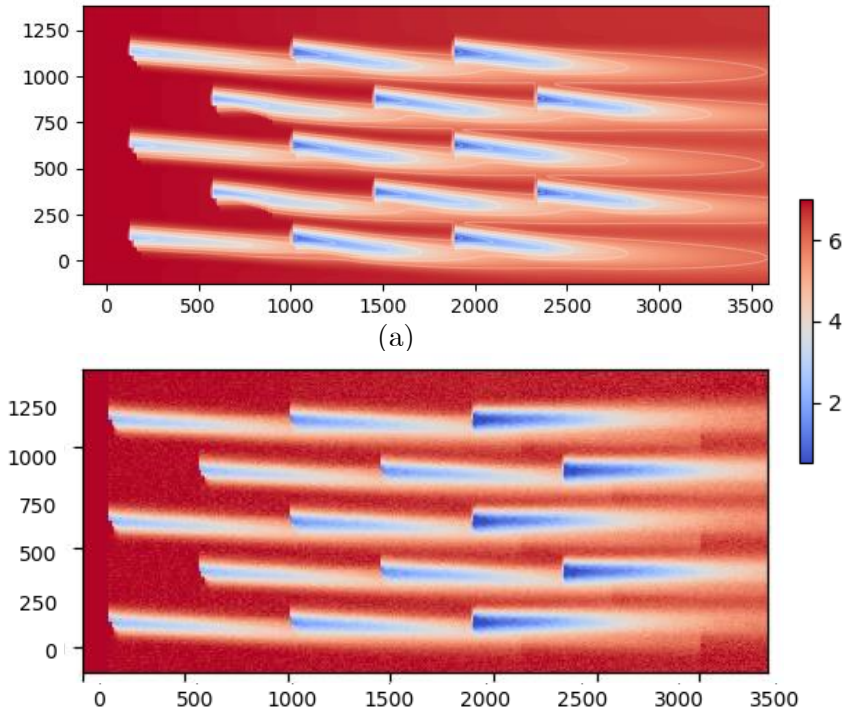
Figure 12. *Case A: Optimised yaw result for a 2x3 wind farm array deduced by FLORIS YawOptimisation module (a) and the DNN (b).*

Table 2. *Optimised results produced by FLORIS and the DNN for Case A. Yaws correspond to turbines from left to right and bottom to top.*

	Yaws (deg)	Initial power (MW)	Optimal power (MW)	Power gain (%)	Comp. time (s)
FLORIS	[29, 29/ 18, 18/ 0, 0]	3.77	4.75	25.8	15
DNN	[26, 26/ 20, 20/ 0, 0]	3.77	4.63	22.8	7

Case B

Case B examines the yaw optimisation of a higher density 15-turbine wind farm, where by using a similar yaw cooperative strategy the power gain achieved by the DNN is again very close to that of FLORIS (28.7% and 31.6%, respectively), requiring almost half the computational time (60s vs 110s).



(a)

(b)

Figure 13. *Case B: Optimised yaw result for a 15x wind farm array deduced by FLORIS YawOptimisation module (a) and the DNN (b).*

Table 3. *Optimised results produced by FLORIS and the DNN for Case B. Yaws correspond to turbines from left to right and bottom to top.*

	Yaws (deg, from left to right rows)	Initial power (MW)	Optimal power (MW)	Power gain (%)	Comp. time (s)
FLORIS	[30, 30, 30/ 23, 23/ 8, 8, 8/ 7.5, 7.5/ 0, 0, 0/ 0, 0]	9.41	12.39	31.6	110
DNN	[25, 24, 24.5/ 24.5, 24/ 19, 18, 18/ 18, 18/ 0, 1, 1/ 0, 0]	9.41	12.12	28.7	60

3.5 Yaw optimisation - Farm power heatmaps

For the above cases A and B, the corresponding total farm power heatmaps are presented in Figures 14 and 15, in order to assess the capability of the neural network in obtaining the optimal power gain. As evident from these heatmaps, the region with the highest potential power gain is around the bottom right corner, which is defined by low turbulence intensity and high inlet speeds. Although cases where high wind speeds are not usually accompanied by low turbulence intensity [24], they are examined for the purpose of creating a heatmap across the whole range of inlet conditions. For Case A, the DNN is capable of producing at least 70% of the total power gain of FLORIS (~80% for speeds below 8m/s), as shown in Figure 14b within 1/10 of the computational time (60s vs 5s). Similarly, in Case B, the DNN achieves a large proportion of the total power gain of FLORIS, requiring less than 1/10 of the computational time (180s vs 16s).

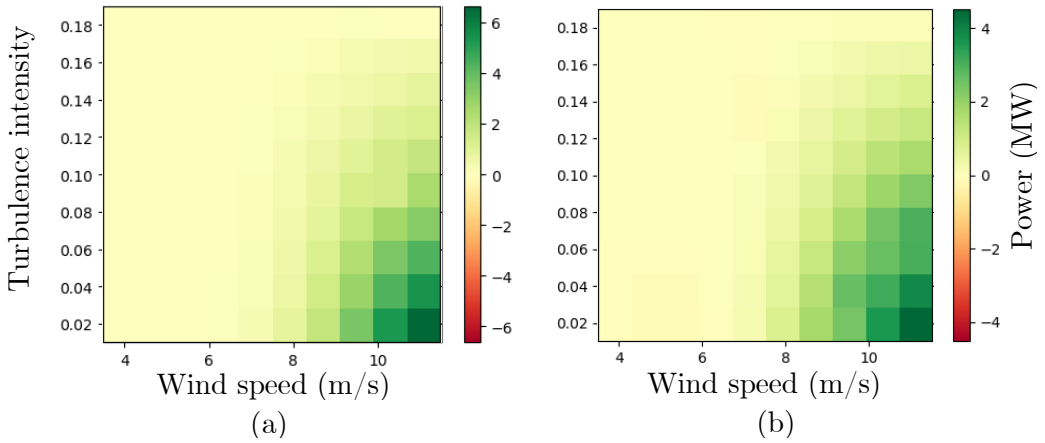


Figure 14. *Case A: Optimised yaw heatmaps for FLORIS (a) and the DNN (b) with ti and ws ranges [0.01-0.19] and [3.5-11.5], respectively. Average FLORIS time: 60s, average DNN time: 5s.*

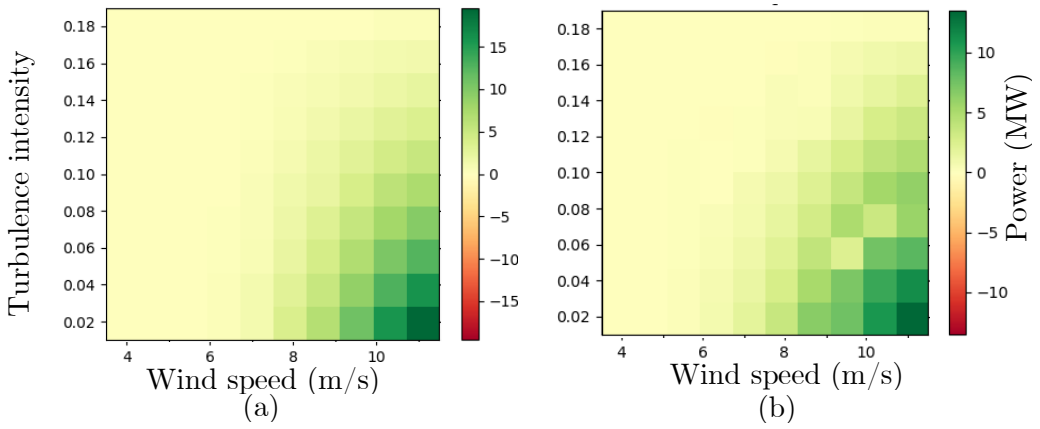


Figure 15. *Case B: Optimised yaw heatmaps for FLORIS (a) and the DNN (b) with ti and ws ranges [0.01-0.19] and [3.5-11.5], respectively. Average FLORIS time: 180s, average DNN time: 16s.*

3.6 Layout optimisation

Case C

An indicative test case for layout optimisation of the initial configuration (Fig. 16a) is presented below, where the wind direction is assumed to be constant at 11 m/s and the boundary constraints of the domain are 20D X 10D. A minimum lateral distance between each individual turbine of 2D has also been implemented as a constraint in the code. As shown in Figure 16b, the optimum configuration obtain with the DNN model achieves a 78.93% power gain (or AEP gain), whereas FLORIS achieves 79.41% (Fig. 16a). It is important to note that, once trained, the computational time required by the DNN to provide a forward solution is an order of magnitude less than that of FLORIS (5.5 sec vs 56 sec).

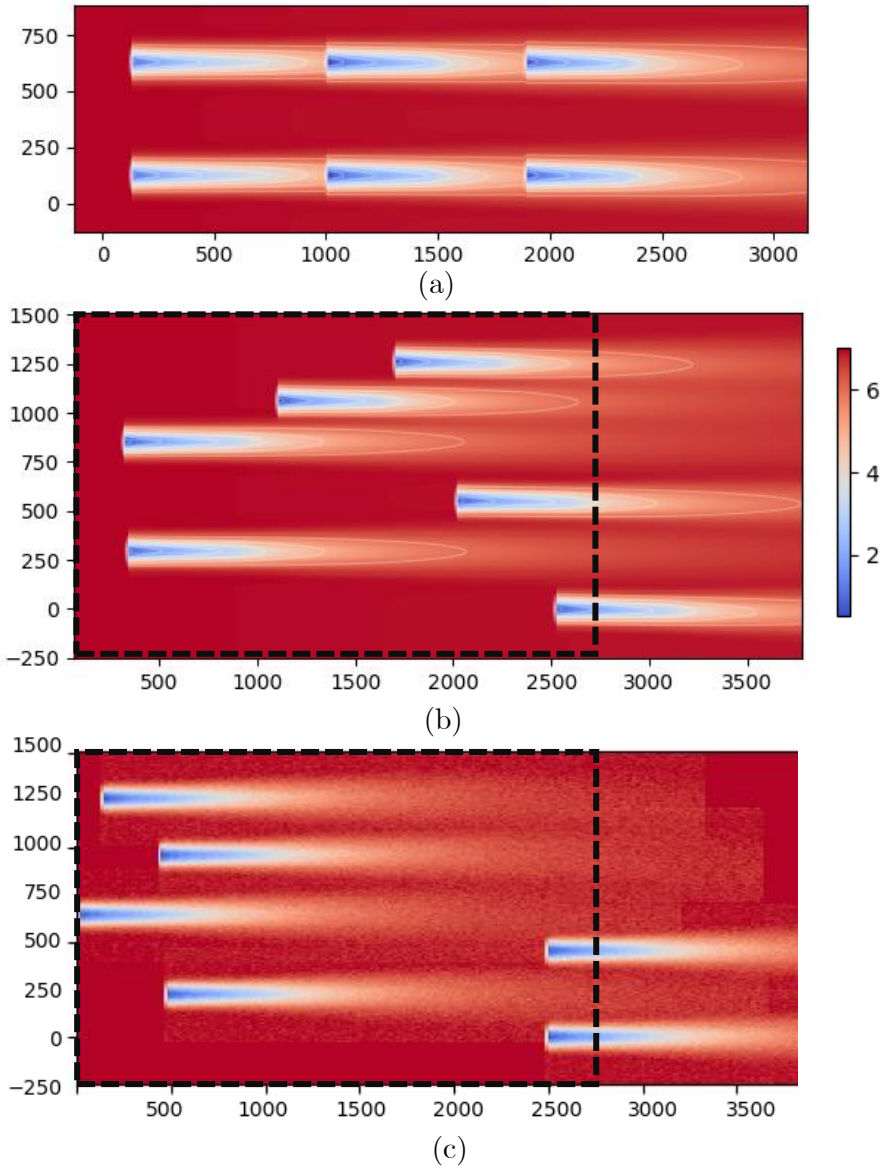


Figure 16. *Case C: Initial 2x3 layout configuration (a), optimised layout result produced by FLORIS LayoutOptimisation module (b) and the DNN model (c). The boundary constraints are defined by the dashed lines (2520 x 1260 m).*

Table 4. *Optimised results produced by FLORIS and the DNN for Case C.*

	Initial power (MW)	Optimal power (MW)	Power gain (%)	Comp. time (s)
FLORIS	3.78	6.78	79.41	56
DNN	3.78	6.76	78.93	5.5

3.7 Layout optimisation - Farm power heatmap

The corresponding heatmaps of Case C are also presented in Figure 17, where again, the DNN optimisation appears to perform well, requiring an order of magnitude less computational time on average (6 sec. vs 70 sec). FLORIS appears to produce some false optimisation results on the bands of 0.02 and 0.075 ti, which could be attributed to divergence of the solution due to the default settings of the optimiser.

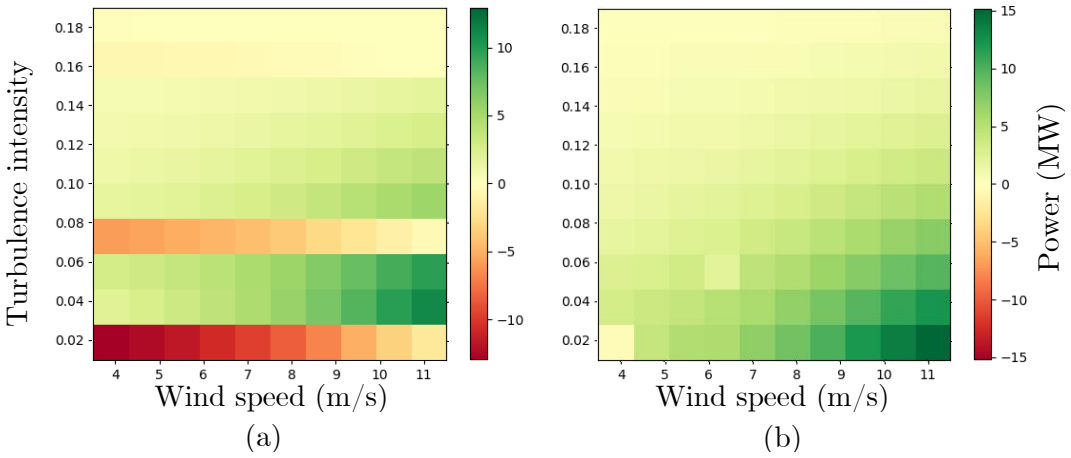


Figure 17. *Case B: Optimised yaw heatmaps for FLORIS (a) and the DNN (b) with ti and ws ranges [0.01-0.19] and [3.5-11.5], respectively. Average FLORIS time: 70s, average DNN time: 6s.*

4. Discussion and Conclusions

In this Thesis, the capabilities of machine learning on the modelling of offshore wind farm wakes is investigated. The main stages and corresponding novel accomplishments of this study were the following:

a. Development of a DNN for the accurate representation of 2D offshore turbine wakes under yaw, which produces the velocity domains of single wakes with an average accuracy of 98.3%. The 2D velocity deficits produced by the DNN at the hub level can also reproduce the sea surface effect on the boundary layer of the downstream wind.

b. Implementation of a special superposition method, which combines the SOS model along with an approximation which renders the 2D wake deficit produced by the DNN sufficient in predicting the power output of a given wind farm. The scaling of the

computational time for the superposition of up to 20 turbines was found to be up to 3 orders of magnitude faster than that of FLORIS.

c. Demonstration of the optimisation capabilities of the neural model against those of FLORIS. For the yaw optimisations, the optimal yaw settings produced by the DNN provided at least 70% of the farm power output of the corresponding FLORIS optimisation module, on average 10 times faster. For the layout optimisations the DNN was capable of producing a higher power output (~2MW more) over a wide range of wind speeds and turbulence intensities.

d. Development of an open-source machine learning wake model with multiple functionalities and settings, aiming to provide flexibility to the user, depending on the needs of application. Using the default model hyper-parameters and settings, a reliable neural wake model can be trained on a personal computer, requiring less than 15 minutes of training time.

The main conclusions gleaned from this study are:

1. Based on the obtained promising results, wake modelling with machine learning tools can be accurate enough to contribute in active yaw and layout optimisation applications.

2. Even using a relatively cheap analytical model like the Gaussian, the computational time gains of the DNN can be significant enough to produce rapid predictions, about one order of magnitude faster, for active yaw and layout optimisations. These findings could enable further wind farm power gains at a minimal installation cost.

3. By further training, using more sophisticated analytical models like Curl or high fidelity CFD results of single turbine wakes, the accuracy of the wake deficits produced by the developed DNN could further increase significantly. It is important to note that the computational time required by the DNN to provide a forward solution would remain constant, thus achieving results of accuracy comparable to those of a CFD model multiple orders of magnitude faster.

Planned future implementations include: (a) addition of the influence that upstream turbines under yaw have on the wind direction at the inlet of turbines downstream, (b) transfer learning using high fidelity CFD data, (c) additional tests and comparison with experimental results for further validation, (d) further optimisation of the DNN structure.

5. Bibliography

[1] Council GWE, Annual Wind report (2019).

[2] Council GWE, Global wind report annual market update (2013), GWEC, Brussels.

[3] G. C. Larsen (2009), A simple stationary semi-analytical wake model, Riso National Laboratory for Sustainable Energy, Technical University of Denmark.

[4] N. Jensen (1983), A note on wind generator interaction.

[5] L.A. Martinez-Tossas, J. Annoni, P.A. Fleming, M.J. Churchfield (2019), The aerodynamics of the curled wake: a simplified model in view of flow control, *Wind Energ. Sci.,* **4**, pp. 127-138.

[6] M. Bastankhah, F. Porté-Agel (2014), A new analytical model for wind-turbine wakes, *Renew Energy*, **70**, pp. 116-123.

[7] N. Sedaghatizadeh, M. Arjomandi, R. Kelso, B. Cazzolato, M.H. Ghayesh (2018), Modelling of wind turbine wake using large eddy simulation, *Renew Energy*, **115**, pp. 1166-1176.

[8] A. Niayifar, F. Porté-Agel (2016), Analytical modeling of wind farms: A new approach for power prediction, *Energies*, **9**, p. 741.

[9] Y.-T. Wu, F. Porté-Agel (2012), Atmospheric turbulence effects on wind-turbine wakes: An LES study, *Energies*, **5**, pp. 5340-5362.

[10] M.P. van der Laan, N.N. Sørensen, P.E. Réthoré, J. Mann, M.C. Kelly, N. Troldborg, *et al.* (2015), An improved k-ε model applied to a wind turbine wake in atmospheric turbulence, *Wind Energy*, **18**, pp. 889-907.

[11] Y.-T. Wu, F. Porté-Agel (2015), Modeling turbine wakes and power losses within a wind farm using LES: An application to the Horns Rev offshore wind farm, *Renew Energy*, **75**, pp. 945-955.

[12] Z. Ti, M. Zhang, Y. Li, K. Wei (2019), Numerical study on the stochastic response of a long-span sea-crossing bridge subjected to extreme nonlinear wave loads, *Eng Struct*, **196**, p. 109287.

[13] M.P. van der Laan, N.N. Sørensen, P.E. Réthoré, J. Mann, M.C. Kelly, N. Troldborg, *et al.* (2015), An improved k-ε model applied to a wind turbine wake in atmospheric turbulence, Wind Energy, **18**, pp. 889-907.

[14] A. El Kasmi, C. Masson (2008), An extended k–ε model for turbulent flow through horizontal-axis wind turbines, *J Wind Eng Ind Aerodyn*, **96**, pp. 103-122.

[15] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis (2018), Hidden Fluid Mechanics: A Navier-Stokes Informed Deep Learning Framework for Assimilating Flow Visualization Data, Division of Applied Mathematics, Brown University.

[16] M.Raissi, P.Perdikaris, G.E.Karniadakis (2019), Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational Physics*, **378**, pp. 686-707.

[17] L. Lu, X. Meng, Zhiping Mao, G. E. Karniadakis (2020), Deepxde: A deep learning library for solving differential equations, Division of Applied Mathematics, Brown University.

[18] B.D. Tracey, K. Duraisamy, J.J. Alonso (2015), A machine learning strategy to assist turbulence model development, *53rd AIAA aerospace sciences meeting*, p. 1287.

[19] J. Ling, A. Kurzawski, J. Templeton (2016), Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, *J Fluid Mech*, **807**, pp. 155-166.

[20] A. Biswas, R. Gupta (2009), An artificial neural network based methodology for the prediction of power & torque coefficients of a two bladed airfoil shaped H-rotor, *Open Renew Energy J*, **2**, pp. 43-51.

[21] A. Biswas, S. Sarkar, R. Gupta (2016), Application of artificial neural network for performance evaluation of vertical axis wind turbine rotor, *Int J Ambient Energy*, **37**, pp. 209-218.

[22] Z. Ti, X. W. Deng, H. Yang (2020), Wake modelling of wind turbines using machine learning, *Applied Energy*, **257**, 114025.

[23] Y. Bengio (2012), Practical recommendations for gradient-based training of deep architectures, Cornell University, arXiv:1206.5533.

[24] K.A. Kragh, M.H. Hansen (2015), Potential of power gain with improved yaw alignment, *Wind Energy*, **18**, pp. 979-989.

[25] T.J. Chung (2002), *Computational Fluid Dynamics*, Cambridge Univ. Press.

[26] S. Ioffe, Ch. Szegedy, (2015), Batch normalization: Accelerating Deep Network training by reducing internal covariate shift, arXiv:1502.03167v3.

[27] I. Guyon (1977), A scaling law for the validation-set training-set size ratio, *AT&T Bell Laboratories*, Berkeley, California.

[28] M. Riedmiller, H. Braun (1993), A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm, *International Conference on Neural Networks.*

[29] J. Kuo, D. Romero, C.H. Amon (2014), A novel wake interaction model for wind farm layout optimization, *ASME 2014 Intl. Mech. Engin. Congress,* Nov. 14-20, Monteral, Canada.

[30] T. Gocmen, P. Laan, P.E. Rethore, A.P. Diaz, G.Ch. Larsen, S. Ott (2016), Wind turbine wake models developed at the technical university of Denmark: A review, *Renewable & Sustainable Energy Reviews*, **60**, pp. 752-769.

[31] P.M. Gebraad, F.W. Teeuwisse, J.W. van Wingerden, P.A. Fleming, S.D. Ruden, J.R. Marden, L.Y. Pao (2016), Wind plant power optimization through yaw control using a parametric model for wake effects – A CFD simulation study, *Wind Energy,* **19**, pp. 95-114.

[32] M. Samorani (2010), The wind farm layout optimization problem, DOI: 10.1007 / 978-3-642-41080-2_2.

[33] D.E. Goldberg DE (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc., Boston.

[34] E.G.A. Antonini, D.A. Romero, C.H. Amon (2020), Optimal design of wind farms in complex terrains using computational fluid dynamics and adjoint methods, *Applied Energy*, **261**, 114426.

[35] R.N. King, K. Dykes, P. Graf, P.E. Hamlingon (2016), Adjoint optimization of wind plant layouts, *Wind Energy Science Discussions,* doi:10.5194/wes-2016-25, 2016.

[36] S.W. Funke, R.E. Farrell, MD. Piggott (2013), Tidal turbine array optimization using the adjoint approach, *Renewable Energy*, **63**, pp. 658-673.

[37] S. Dongran, J. Yang, Y. Liu, M. Su, A. Liu, Y.H., Joo (2017), Wind direction prediction for yaw control of wind turbines, Intl., J. of Control Automation and Systems, **15**, pp. 1-9.

[38] A.S. Dar, L. von Bremen (2019), Short-Term Forecasting of Wake-Induced Fluctuations in Offshore Wind Farms, *Energies*, **12**, 2833; doi:10.3390/en12142833.