Neural networks for predicting the output of wind flow simulations over complex topographies

Michael Mayo
Dept. of Computer Science
University of Waikato
Hamilton, New Zealand
michael.mayo@waikato.ac.nz

Sarah Wakes
Centre for Materials Science and Technology
University of Otago
Dunedin, New Zealand
sarah.wakes@otago.ac.nz

Chris Anderson
Dept. of Computer Science
University of Waikato
Hamilton, New Zealand
zsmyna@gmail.com

***Abstract*—We use deep learning techniques to model computational fluid dynamics (CFD) simulations of wind flow over a complex topography. Our motivation is to "speed up" the optimisation of CFD-based simulations (such as the 3D wind farm layout optimisation problem) by developing surrogate models capable of predicting the output of a simulation at any given point in 3D space, given output from a set of training simulations that have already been run. Our promising results using TensorFlow show that deep neural networks can be learned to model CFD outputs with an error of as low as 2.5 meters per second.**

***Index Terms*—wind farm layout optimisation, deep learning, computational fluid dynamics, wind flow modelling, complex topography**

I. Introduction

We consider the problem of predicting the output of computational fluid dynamics (CFD) simulations using deep learning techniques. CFD is a branch of fluid mechanics dealing with the modeling of fluid flows using the Navier-Stokes equations [1] and has many applications in engineering design (e.g. [2]–[6]). However, a well-known disadvantage of CFD is the computation time: high-fidelity simulations often take many hours or even days of CPU-time to complete. If a CFD simulation is part of a larger optimisation algorithm (e.g. a genetic algorithm, requiring thousands of function evaluations) then the compute time required to run all the required CFD simulations may be impractical.

Therefore, the use of machine learning and deep learning techniques for building approximate models of CFD simulations is pertinent. Ideally a neural network model of a CFD simulation would be faster than running a full CFD simulation while at the same time retaining as much of the accuracy (for the purposes of subsequent analysis) as possible.

An illustrative problem domain where this approach could be useful is the 3D wind farm layout optimisation problem with complex terrain. In this problem, the aim is to optimise the positions and properties of wind turbines on a wind farm site. The wind farm site has a non-flat topography and therefore the optimiser must take into account the wind flow dynamics occurring as the wind passes over hills, along valleys, and through and between turbines at different heights. A critical factor is the wake interference between wind turbines which occurs when one turbine stands downwind of another.

The 2D version of the problem with flat topography [7] has been widely studied for several years. For the interested reader, Samorani [7] provides an accessible introduction while [8], [9] and [10] are all comprehensive recent surveys. In the 2D variant of the problem, simple and fast mathematical models exist for approximating wind turbine wakes and their interactions. The classic and still practically useful example is the Jensen model [7], [10]–[12].

The 3D variant of problem with complex terrain, on the other hand, has been researched to a much lesser degree. Herbert-Acero et al., in their 2014 survey [8], state (pp. 6987): "An important aspect that has been largely neglected is the topography of the wind farm area."

Since then, only a small handful of papers concerning the 3D wind farm layout optimisation problem with complex terrain have been published. These include the work of M. Song [13]–[15] who developed the "virtual particle" model; approaches that combine turbine-free CFD wind flow simulations with 2D wake models [16], [17]; and one approach that models a single turbine in different positions using CFD and then combines the simulation outputs together so that an overall estimate of the performance of multiple turbines can be made [18].

In each approach, performing multiple CFD simulations is a key part of the method since CFD is the only known method of accurately modeling wind flows over non-2D terrains. However, due to the computational complexity of CFD, the number of simulations that can be run is limited.

Our long term goal behind this research is to solve the 3D wind farm layout optimisation. However, in this research we focus on modeling turbine-free wind flows over a complex topography only. There are two main reasons for excluding turbines from the present study: (i) some existing methods for 3D wind farm layout optimisation use only turbine-free flow models anyway (e.g. the work of Feng [16] and Zheng [17] mentioned above, so our techniques could speed up their approaches); and (ii) the turbine-free wind flow prediction problem is itself difficult to solve and if we can make progress on this simpler form of the problem, then we can add turbines and their wakes to the CFD simulations at a later date.

This paper continues our work first started in [19] in which we used several machine learning methods in additional to neural networks to model the CFD-generated wake of a wind turbine in the absence of terrain. The results in that work showed that such an idea was feasible and that neural networks performed best out of a range of approaches including random forests and linear regression.

To evaluate the effectiveness of deep learning for predicting CFD output, we use simulation-based cross validation and we record the mean absolute error (MAE) and the Pearson product-moment correlation coefficients (CC) of each model. Our results show that deep learning can be effective for predicting the output of CFD-based wind flow simulations over complex topographies.

II. Background

A. CFD models of wind flow over complex terrain

There is an increasing need to understand wind flows over complex terrain for land planning purposes, wind farm power predictions, prediction of sediment erosion and deposition among others [20]–[24]. Previous simplifications that ignored topography changes or reduced them to isolated features are often now not enough. For example, Lange et al [25] found minor changes in the terrain could have a significant effect on flow parameters for wind turbines and wind variables were sensitive to topography details.

Typical methods to do this may include scale wind tunnel testing, field experiments and numerical simulation [21], [25]–[27]. Each have associated issues such as scale, resources or limits to data being measured. CFD simulations of wind flows over complex terrain is therefore an ambitious prospect. New surveying techniques permit the development of digital terrain models, however a stumbling block is the ability of CFD to emulate the wind flow over such landscapes [26]–[30]. Issues such as domain size, noise in the terrain model, computational capacity and resource, limit what is achievable in a CFD simulation.

B. Neural networks

Neural networks are a type of machine learning model useful for supervised or unsupervised learning from data. The main power of a neural network lies in its ability to discover and represent highly non-linear patterns that may exist in a large dataset. Figure 1 depicts a neural network. As the figure shows, neural networks are directed graphs with nodes representing either inputs, outputs, or function of other nodes if the node is part of a "hidden" layer. Each arc is weighted and the learning problem is to find the set of weights that most closely model the patterns in the data without over- or under-fitting.

Traditional neural networks consist of one or a few hidden layers, because the training of such models is easier. In more recent times however with the advent of deep learning, effective algorithms have been discovered for training huge networks with multiple hidden layers. [31] is an excellent introduction to this field and the interested reader is referred to that source for more information.

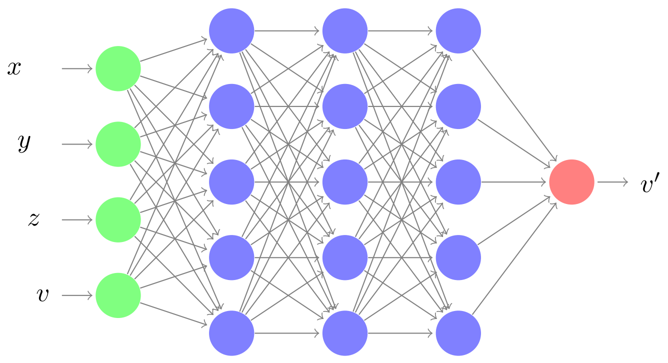


Fig. 1: An example of supervised neural network with four inputs, namely the (x, y, z) position of a point in 3D space and the uniform incoming wind velocity magnitude for the entire topography v. This network consists of three hidden layers and one output node v', that being the predicted wind velocity at point (x, y, z).

III. Approach

In this section, we describe our basic approach which involves (i) constructing suitable datasets for evaluating and comparing different models and (ii) the methodology we use for determining the optimal neural network configuration and hyper-parameters.

A. Dataset construction

The terrain used in our experiments is part of a coastal dune system, 66.2m along shore (x direction) and 105.4m in length (y direction). The computational domain was 78.5m in height (z direction). The terrain is illustrated in Figure 2. The datasets were generated with 10,000 streamlines on (y, z) planes at fixed x values along shore. Samples were concentrated in regions that had higher mesh near the terrain surface.

The complete output files from four CFD runs are in general cumbersome to deal with directly and we identified two main issues with the data. Firstly, each CFD run produces tens of millions of (x, y, z) samples; to make the data easier to handle, therefore, we decided to subsample it.

Secondly, the original data was highly imbalanced with a disproportionate number of samples coming from certain regions of the terrain near the surface. Imbalanced data is well known to cause problems when training and evaluating machine learning models and therefore it must be handled carefully [32]. Imbalanced training data can lead to poor predictive models, and imbalanced test data can give misleading results. Furthermore, while numerous methods exist for handling imbalanced classification problems, there are far fewer techniques available for dealing with imbalanced regression problems [32]. We therefore circumvented the problem by balancing the data at the same time as we sub-sampled it. In this way, the selected samples became more uniformly distributed across the 3D volume of each simulation.

The basic approach we took to subsample and balance the data was as follows: for each simulation's set of output

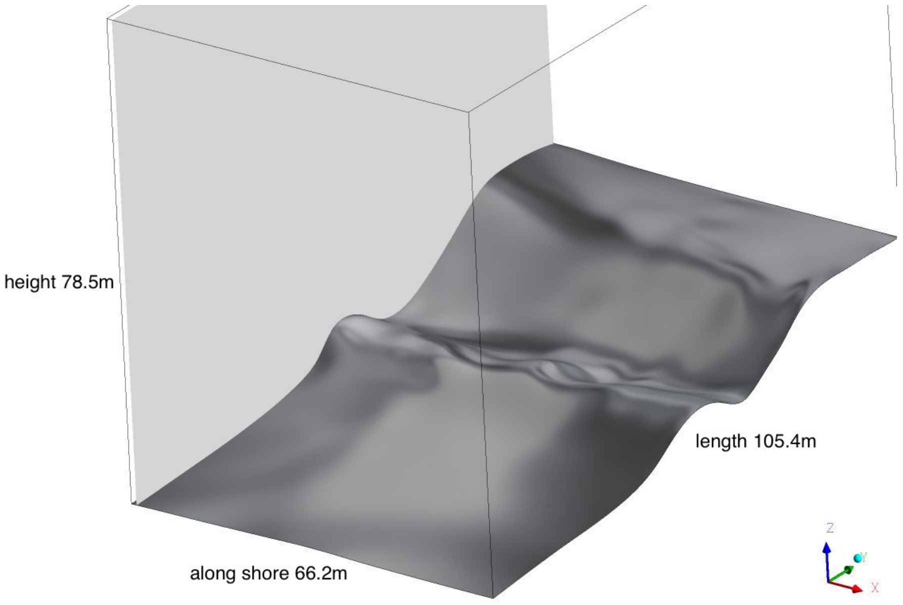


Fig. 2: Terrain used in our CFD experiments.

Simulation	Samples (before)	Samples (after)
10 m/s	84,228,589	846,121
15 m/s	67,023,368	924,088
20 m/s	90,276,605	794,307
25 m/s	89,941,611	782,423

TABLE I: Dataset sizes (i.e. number of 3D examples) before and after random sampling and balancing.

samples, we divided the 3D volume containing the samples into 200×200 sub-volumes of equal size. We then began randomly selecting samples from the original dataset, until at most 300 samples were selected per sub-volume. This meant that in theory, at most 200×200×300=12M samples could be selected, although in practice many fewer samples were selected because most sub-volumes had fewer than 300 samples while some sub-volumes had an excessive number of samples. Table I gives the dataset sizes before and after subsampling/balancing. As can be observed, the resulting datasets even after this process are still quite large, containing seven to nine hundred thousand examples each.

B. Hyper-parameter optimisation experiments

Important issues when designing neural networks are the selection of an appropriate configuration of hidden layers for the network and a decision about values for the hyper-parameters used by the learning algorithm. In general, the answer to these questions are problem dependent. Poor choices may severely impact on the ability of the model to learn and generalise.

In this research, we follow the approach for configuring neural network models outlined by [33]. The idea is to continuously test random network configurations of hyper-parameters until a near-optimal combination is found. If 240 such trials are performed, for example, then there is a $1-(\frac{19}{20})^{240} \approx 0.999$ probability of finding a configuration within the same 5% of the configuration space as the optimal configuration, and a $1-(\frac{99}{100})^{240} \approx 0.91$ probability of finding a configuration within the same 1% of the space. In general, this method outperforms grid-based search because grid-based search is restricted the configurations that lie on the grid lines, but in some cases optimal configurations may lie "between" the grid lines [33]. Furthermore, random search is simple to implement compared to other grid search methods.

To use random search we also require good bounds on the hyper-parameters in order to make the search feasible. To this end, a variety of heuristics that can be used as constraints have been proposed in the literature. [34], for example, demonstrates that setting the size of each consecutive hidden layer in a network to the same number of hidden nodes (e.g. a network with three hidden layers each of size five, as shown in Figure 1) works better than the "inverted pyramid" configuration in which each successive hidden layer is smaller than the previous one. Similarly, suitable ranges for hyper-parameters such $[0.01, 2]$ for the learning rate [34] have also been proposed. We utilise these guidelines wherever possible.

Table II gives the hyper-parameter ranges that we finally decided on. The first hyper-parameter (number of epochs) concerns the amount of time to train the network for; the second and third hyper-parameters concern the architecture of the network; and the final hyper-parameters are concerned with the learning algorithm which was the proximal adagrad optimizer [35] implemented in TensorFlow r1.4 [36]. We explored two types of regularisation (L1 and L2) but only allowed one type at a time. With probability $\frac{1}{3}$, L1 regularisation was chosen; with probability $\frac{1}{3}$, L2 regularisation was chosen; otherwise neither regularisation strategy was employed.

The hidden units utilised the rectified linear (RELU) function and the batch size for training was fixed at 50 examples. Unless otherwise stated by the table, all other hyper-parameters and settings are defaults as used in TensorFlow r1.4.

To evaluate the effectiveness of each random combination of hyper-parameter values, we performed repeated four-fold cross validation-by-simulation experiments. In order words, given the four simulations that we had data for, we trained the network on data from three of the simulations and then predicted the output samples of the remaining simulation. We repeated this procedure three times for each of the four folds of the cross validation. because neural networks can be sensitive to their initial random weights In summary, for each neural network configuration that we tested, we performed $3 \times 4 = 12$ train/test runs in order to assess the performance. The MAE and CC results for each of the twelve runs were then averaged, and this whole process was repeated 240 times for different random hyper-parameter combinations.

IV. Results

In this section we detail the main results from our experiments along with results of some additional analysis performed following the main experiment.

A. Main experimental results

Summary results of our experiments are shown in Figure 3. For each of the 240 sets of repeated cross-validations, we computed the mean MAE and mean CC. In the figure, a higher CC is preferred, with a maximum CC possible being 1.0. A CC of 0.0 indicates no correlation at all, and negative CC indicates a negative correlation. For MAE, lower is better with 0.0 indicating perfect predictions. We can observe from the figure that most of the models we tested have an MAE of less than 6.0, with a significant cluster of models with MAE of less than 4.0. CC, however, is much more distributed with wider range of approximately 0.5-0.8, even for the relatively low MAE models. Interestingly there are a few models with high CC but poor MAE, suggesting that these models may be useful but the scale of the predictions is clearly wrong, even though the direction is correct. Some networks have plainly failed to generalise at all as points on the plot appearing on the right-hand side of the diagram show.

How do these figures compare with a simple baseline approach that does not involve learning from the training data? To assess the answer to this question, we evaluated the simple mean predictor method. This method computes the mean $\bar{v}'$ of the target variable v' in the training data and then subsequently predicts $\bar{v}'$ for each test example. Since the predictions are constant, the CC is always zero, but the testing MAE depends on the variation of the target variable in the test data. Table III shows the results of this analysis. Overall, mean prediction achieves an average MAE of 7.05 m/s with the worst performance occurring on the 25 m/s data.

Next, we explored two of the models in more detail, specifically (a) the model with the lowest average MAE of 2.58 m/s and the (b) the model with the highest average CC of 0.82. The hyper-parameters for these models are shown in Table IV. Generally speaking, both sets of hyper-parameters are quite different. For example, the dropout rates vary by an order of magnitude, and one network uses L2 regularisation while the other does not. Neither model is small in terms of size. Model (a) has three hidden layers, each of which is relatively large with 47 hidden nodes; and (b) has four hidden layers of size 24.

Detailed performance metrics for both of these optimal configurations are provided in Table V. The table shows the specific test results for each of the twelve training/test runs performed using each configuration. It can be observed that different runs on the same training/test set often produce quite different results. For example, configuration (a) achieves a model with test MAE of 1.57 m/s on the 25 m/s data on one run, but 3.49 m/s on another run. More extremely, one of the runs of a model built with configuration (b) on the 10 m/s data produces a very high error of 12.73 m/s compared to the other two runs which are both below 0.85 m/s error.

Given that the hyper-parameters are constant for each model's runs, this suggests that the random initialisation of the network weights may be primarily responsible for the variation in performances when the training and test data is the same. This warrants further investigation.

As a further comparison, we also ran some non-neural network machine learning approaches. Standard linear regression achieved an average MAE across four folds of 7.85(1.33) m/s while random forests for regression (100 trees) achieved 7.69(1.36). Thus we can conclude that neural networks are a competitive approach outperforming other machine learning algorithms for this problem.

B. Hyper-parameter/performance correlation

Next, we explored which of the hyper-parameters in the search space were most responsible for the variation in MAE and CC. This can be achieved by computing the correlation between each individual hyper-parameter and the two performance metrics for all 240 sets of results. Figure 4 shows the very interesting results of this analysis. The most important hyper-parameter is the learning rate, which positively correlates with MAE and negatively correlates with CC. Therefore a smaller learning rate is preferred. The next most important parameter is the number of hidden layers, which is correlated in the same direction as the learning rate, but not to as great a degree. Therefore it can be concluded that shallower networks performed better for this problem. Interestingly, the two regularisation parameters (L1 and L2) have almost zero correlation with MAE and CC in our results.

To continue our analysis of the best configurations that the random sampling experiment found, we performed a smaller number of further experiments using just the two best configurations.

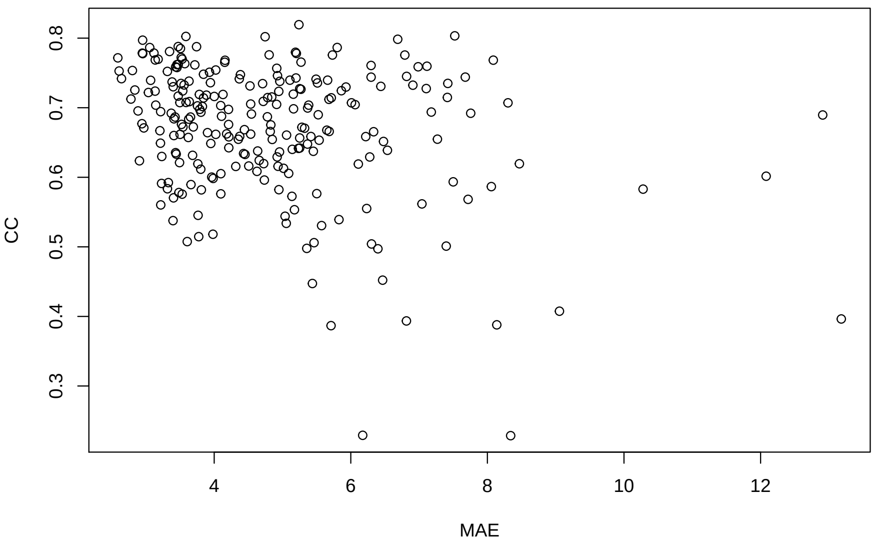
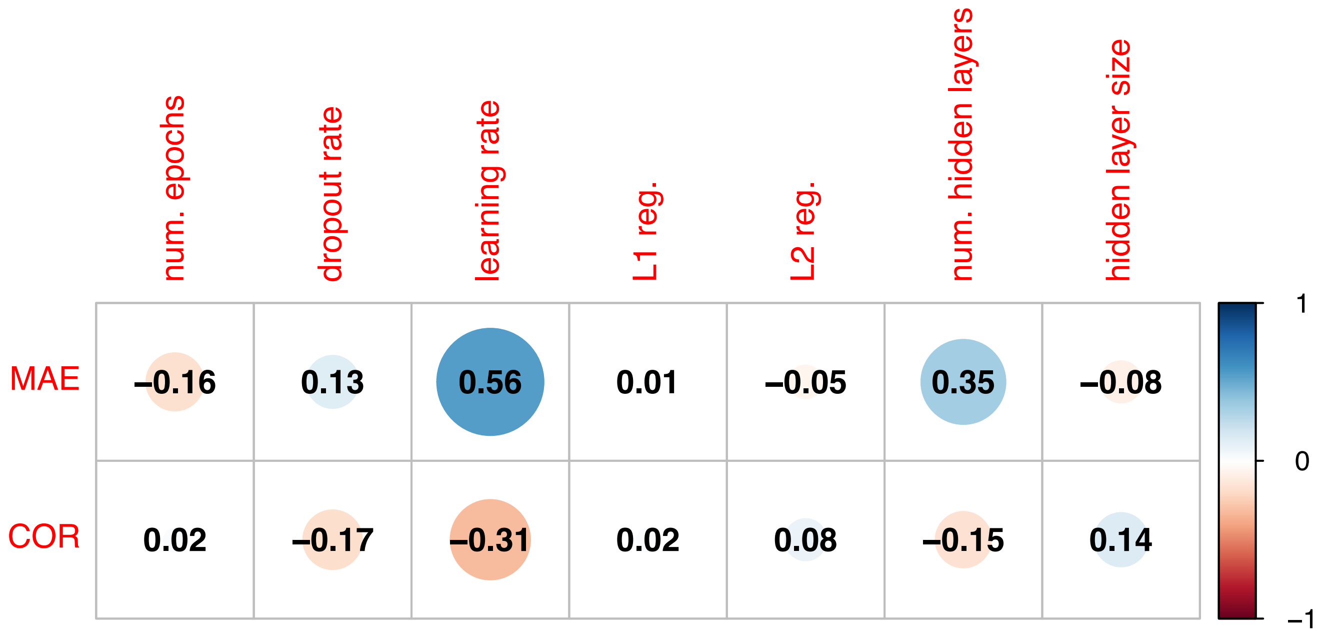


Fig. 3: Averaged MAE vs. averaged CC for 240 different sets of neural network hyper-parameters.



	num. epochs	dropout rate	learning rate	L1 reg.	L2 reg.	num. hidden layers	hidden layer size
MAE	−0.16	0.13	0.56	0.01	−0.05	0.35	−0.08
COR	0.02	−0.17	−0.31	0.02	0.08	−0.15	0.14

Fig. 4: Correlations between performance metrics and hyper-parameters.

Hyper-parameter	Range	Selection
num. epochs	$[1, 20]$	uniform
hidden layer size	$[4, 64]$	uniform
num. hidden layers	$[1, 4]$	uniform
dropout rate	$[0.0, 0.5]$	uniform
learning rate	$[0.001, 2.0]$	exponential
L1 reg. constant	$[3.1 \times 10^{-7}, 3.1 \times 10^{-4}]$	exponential
L2 reg. constant	$[3.1 \times 10^{-7}, 3.1 \times 10^{-4}]$	exponential

TABLE II: Hyper-parameter optimisation experiment ranges. The "uniform" selection scheme denotes random uniform selection from the range of possible values; the "exponential" scheme denotes random uniform selection from the logarithm of the range of values, a scheme suggested in [33].

Testing Sim.	Training Sim.	MAE	CC
25 m/s	{10 m/s, 15 m/s, 20 m/s}	9.82	0.00
20 m/s	{10 m/s, 15 m/s, 25 m/s}	5.22	0.00
15 m/s	{10 m/s, 20 m/s, 25 m/s}	7.24	0.00
10 m/s	{15 m/s, 20 m/s, 25 m/s}	5.92	0.00
mean		7.05	0.00

TABLE III: Individual MAE and CC for the mean predictor.

Hyper-parameter sets	config. (a)	config. (b)
num. epochs	17	9
hidden layer size	47	24
num. hidden layers	3	4
dropout rate	8.56×10^{-2}	2.99×10^{-1}
learning rate	2.30×10^{-3}	5.59×10^{-2}
L1 reg. constant	0.0	0.0
L2 reg. constant	6.54×10^{-5}	0.0

TABLE IV: Optimal hyper-parameter values for the network (a) with minimum MAE, and (b) with maximum CC.

The first of these concerned the amount of computational time spent training the models. The number of epochs used for training in our hyper-parameter optimisation experiments was $[1, 20]$, selected uniformly. We were curious as to whether or not enough training epochs were given, so we repeated the twelve train/test runs of both models built from configurations (a) and (b), but instead of a low number of epochs, we used 20, 40, 80 and 160 epochs for training respectively. This failed to yield any further improvement in either of our test metrics, suggesting that the optimised number of epochs is correct.

We were also concerned about the lack of impact of the regularisation parameters on performance, as it is commonly assumed that regularisation is essential to prevent the over-fitting of predictive models. Therefore we re-ran all twelve runs of model (a) with the L2 hyper-parameter set firstly to 3.1×10^{-4}, which was the top of our range in the random search optimisation experiment, and on a second run to 3.1×10^{-3}, an order of magnitude larger. Again, we observed no further improvements in model (a)'s performance. We therefore can conclude that the parameters selected by the original 240 repetition experiment are reasonable.

C. Visualisations

Finally, to conclude our results, Figure 5 presents visualisations of the predictions produced by models (a) and (b). The figures plot y (distance in wind direction) against z (height) and average the prediction errors over x. Clearly, model (a)'s predictions of the 25 m/s data are good, with low MAE across most of the volume except for the turbulent region between the dunes. Model (b)'s predictions, however, have an overall lower MAE (because the the test set is the 10 m/s data) but the prediction errors are more spread out.

V. Conclusion

To conclude, the best results we obtained suggest that an expected error of around 2.5 m/s is possible, but this figure likely depends on (i) the topography itself, which may be more or less complex; and (ii) the amount of simulations performed from which training data can be obtained. In our experiments, data from three simulations was used to predict the output of a fourth simulation. If data from more simulations is available, the error could likely be driven further down.

With respect to future research, (specifically related to steps 1 and 2 of the approach described in the introduction), exploring random weight initialisation strategies would be an interesting topic, as clearly our experiments have shown that TensorFlow's default strategy leads to high variance in test performance between runs when all hyper-parameters are the same. Similarly, it would be useful to perform a comprehensive comparison of different deep learning optimisation algorithms and assess the differences.

Next, the turbulent regions of higher error (shown in Figure 5) are interesting. Prediction error is not uniformly spread across the 3D volume of the test simulations. Possibly different configurations of network are optimal for different regions of the 3D space, and therefore multiple complementary networks could be used.

As mentioned, our long term aim is to provide a new technique for solving the 3D wind farm layout optimisation problem. However, more work needs to be done to ensure we have as accurate a model for turbine-free wind flow modeling first.

References

[1] P. Wesseling, *Principles of computational fluid dynamics*. Springer Science & Business Media, 2009, vol. 29.

[2] C. Wang, "Applying computational fluid dynamics to optimize adjustments on oral appliance used for the treatment of snoring and obstructive sleep apnea," *Sleep*, vol. 40, p. e136, 2017.

[3] M. Fahey, S. Wakes, and C. Shaw, "The use of CFD in oven design," *International Journal of Multiphysics*, vol. 2, no. 1, pp. 37–57, 2008.

[4] F. Johnson, E. Tinoco, and N. Yu, "Thirty years of development and application of CFD at Boeing Commercial Airplanes, Seattle," *Computers and Fluids*, vol. 34, pp. 1115–1151, 2005.

Testing Sim.	Training Sim.	MAE (a)	CC (a)	MAE (b)	CC (b)
25 m/s	{10 m/s, 15 m/s, 20 m/s}	2.15	0.89	4.35	0.89
25 m/s	{10 m/s, 15 m/s, 20 m/s}	3.49	0.80	3.88	0.89
25 m/s	{10 m/s, 15 m/s, 20 m/s}	1.57	0.89	4.41	0.88
20 m/s	{10 m/s, 15 m/s, 25 m/s}	2.69	0.85	7.42	0.89
20 m/s	{10 m/s, 15 m/s, 25 m/s}	3.88	0.79	3.02	0.87
20 m/s	{10 m/s, 15 m/s, 25 m/s}	3.35	0.86	7.80	0.90
15 m/s	{10 m/s, 20 m/s, 25 m/s}	4.38	0.72	1.79	0.69
15 m/s	{10 m/s, 20 m/s, 25 m/s}	4.01	0.67	7.58	0.63
15 m/s	{10 m/s, 20 m/s, 25 m/s}	1.91	0.70	8.25	0.72
10 m/s	{15 m/s, 20 m/s, 25 m/s}	1.18	0.65	0.84	0.76
10 m/s	{15 m/s, 20 m/s, 25 m/s}	0.99	0.79	12.73	0.84
10 m/s	{15 m/s, 20 m/s, 25 m/s}	1.46	0.64	0.82	0.85
mean		$\mathbf{2.58(1.19)}$	0.77(0.09)	5.24(3.58)	$\mathbf{0.82(0.09)}$

TABLE V: Individual MAE and CC for the best two networks, for each individual run. Network (a) is the model with minimum MAE, and network (b) has maximum CC. Each cross-validation was repeated three times, so there are three results for each train/test split.

[5] H. Avci, "Optimisation of the design parameters of a domestic refrigerator using CFD and artificial neural networks," *International Journal of Refrigeration*, vol. 67, pp. 227–238, 2016.

[6] M. Dharmawan, "Aerodynamic analysis of formula student car," in *AIP*. AIP, 2008.

[7] M. Samorani, *The Wind Farm Layout Optimization Problem*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 21–38. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-41080-2_2

[8] J. F. Herbert-Acero, O. Probst, P.-E. Réthoré, G. C. Larsen, and K. K. Castillo-Villar, "A review of methodological approaches for the design and optimization of wind farms," *Energies*, vol. 7, no. 11, p. 6930, 2014. [Online]. Available: http://www.mdpi.com/1996-1073/7/11/6930

[9] J. González, M. Payán, J. Santos, and F. González-Longatt, "A review and recent developments in the optimal wind-turbine micro-siting problem," *Renewable and Sustainable Energy Reviews*, vol. 30, pp. 133–144, 2014.

[10] R. Shakoor, M. Hassan, A. Raheem, and Y. Wu, "Wake effect modeling: A review of wind farm layout optimization using Jensens model," *Renewable and Sustainable Energy Reviews*, vol. 58, no. C, pp. 1048–1059, 2016.

[11] N. Jensen, "A note on wind generator interaction." Risø DTU National Laboratory for Sustainable Energy, Tech. Rep., 1983.

[12] I. Katic, J. Høstrup, and N. Jensen, "A simple model for cluster efficiency," in *Proc. Europe and Wind Energy Association Conference and Exhibition*, 1986.

[13] M. Song, K. Chen, Z. He, and X. Zhang, "Wake flow model of wind turbine using particle simulation," *Renewable Energy*, vol. 41, no. Supplement C, pp. 185 – 190, 2012.

[14] ——, "Bionic optimization for micro-siting of wind farm on complex terrain," *Renewable Energy*, vol. 50, pp. 551–557, 2013.

[15] ——, "Optimization of wind farm micro-siting for complex terrain using greedy algorithm," *Energy*, vol. 67, pp. 454–459, 2014.

[16] J. Feng and W. Shen, "Wind farm layout optimization in complex terrain: A preliminary study on a gaussian hill," *Journal of Physics: Conference Series*, vol. 524, no. 1, 2014.

[17] K. Zheng, W. Tian, J. Qin, and H. Hu, "Investigation of wind turbine wakes over complex terrain based on actuator disk method," *35th AIAA Applied Aerodynamics Conference*, 2017.

[18] J. Kuo, I. Wong, D. Romero, J. Beck, and C. Amon, "Wind farm layout optimization in complex terrains using computational fluid dynamics," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 2A, 2015.

[19] B. Wilson, M. Mayo, and S. Wakes, "Surrogate modeling a computational fluid dynamics-based wind turbine wake simulation using machine learning," in *Proc. IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017.

[20] S. Hong, "CFD modelling of livestock odour dispersion over complex terrain, part i: Topographical modelling," *Biosystems Engineering*, vol. 108, no. 3, pp. 253–264, 2011.

[21] H. Nedjari, G. O., and S. M., "CFD wind turbine wake assessment in complex topography," *Energy Conservation and Management*, vol. 138, pp. 224–236, 2017.

[22] S. Parker and R. Kinnersley, "A computational and wind tunnel study of particle dry deposition in complex topography," *Atmospheric Environment*, vol. 38, no. 23, pp. 3867–3878, 2004.

[23] G. Katul and D. Poggi, "The influence of hilly terrain on aerosol-sized particle deposition into forested canopies," *Boundary-Layer Meteorology*, vol. 135, pp. 67–88, 2010.

[24] E. Abd-Elaad, J. Mills, and M. X., "Numerical simulation of downburst wind flow over real topography," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 172, pp. 85–92, 2018.

[25] J. Lange, "For wind turbines in complex terrain, the devil is in the detail," *Environmental Research Letters*, vol. 12, 2017.

[26] S. Wakes, "Numerical modelling of wind flow over a complex geometry," *Environmental Modelling and Software*, vol. 25, no. 237-247, 2010.

[27] A. Hart, "The impact of ammophila foredune development on downwind aerodynamics and parabolic dune development," *Journal of Coastal Research*, vol. 28, no. 1, pp. 112–122, 2012.

[28] S. Wakes, M. Hilton, and T. Konlechner, "Topographic steering of oblique incident winds across a foredune-parabolic topography, mason bay, stewart island, new zealand," *Journal of Coastal Research*, vol. S175, pp. 243–247., 2016.

[29] S. Wakes, "Three-dimensional computational fluid dynamic experiments over a complex dune topography," *Journal of Coastal Research*, vol. 65, pp. 1337–1342, 2013.

[30] W. Pattanapol, S. Wakes, and M. Hilton, "Using computational fluid dynamics to determine suitable foredune morphologies in New Zealand," *Journal of Coastal Research*, vol. S164, no. 1, pp. 298–302, 2011.

[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

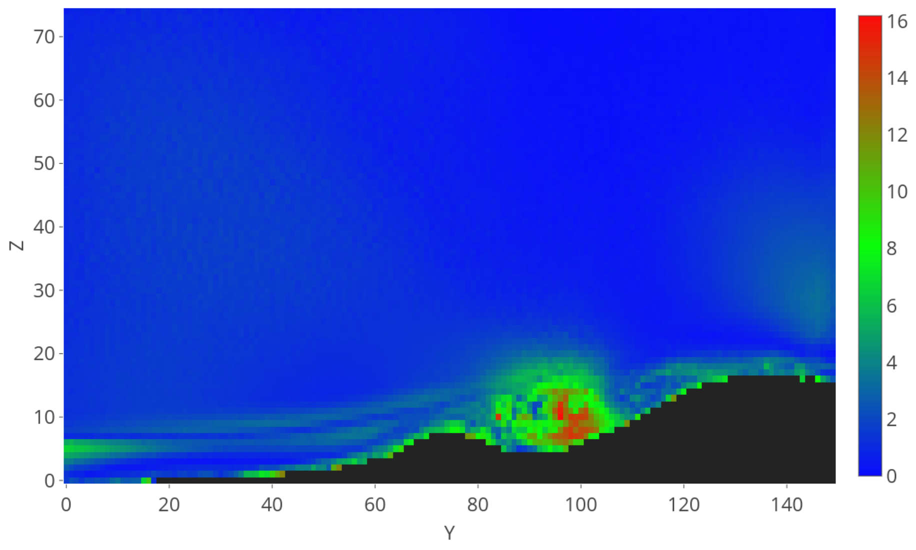
[32] P. Branco, L. Torgo, and R. P. Ribeiro, "A survey of predictive modeling on imbalanced domains," *ACM Comput. Surv.*, vol. 49, no. 2, pp. 31:1–31:50, Aug. 2016. [Online]. Available: http://doi.acm.org/10.1145/2907070

[33] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/jmlr/jmlr13.html#BergstraB12

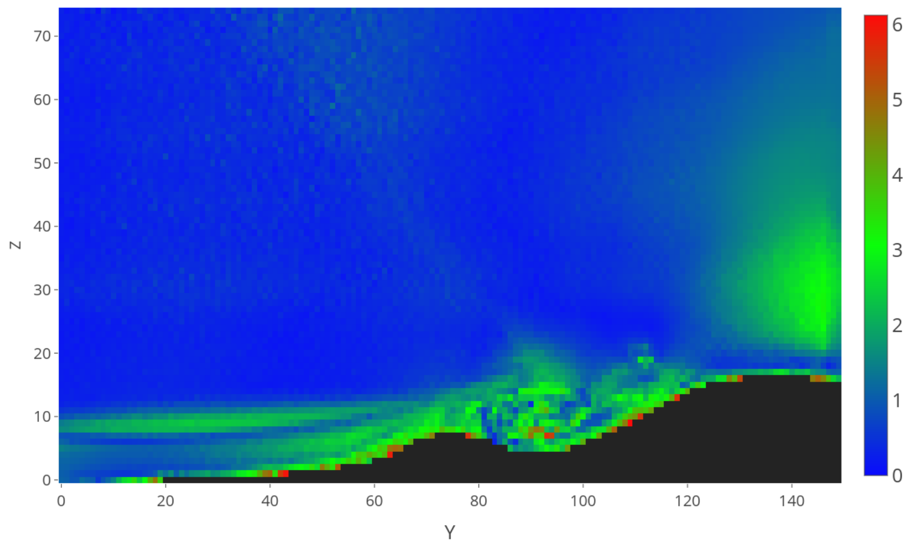
[34] Y. Bengio, *Practical Recommendations for Gradient-Based Training of Deep Architectures*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 437–478.

[35] Y. Singer and J. Duchi, "Efficient learning using forward-backward splitting," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 495–503.

[36] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf



(a) Aggregated prediction errors on the 25 m/s datasets after training on the {10 m/s, 15 m/s, 20 m/s} datasets, run 3



(b) Aggregated prediction errors on the 10 m/s datasets after training on the {15 m/s, 20 m/s, 25 m/s} datasets, run 3

Fig. 5: Predictions made by models (a) and (b). Please view the colour online version of this figure.