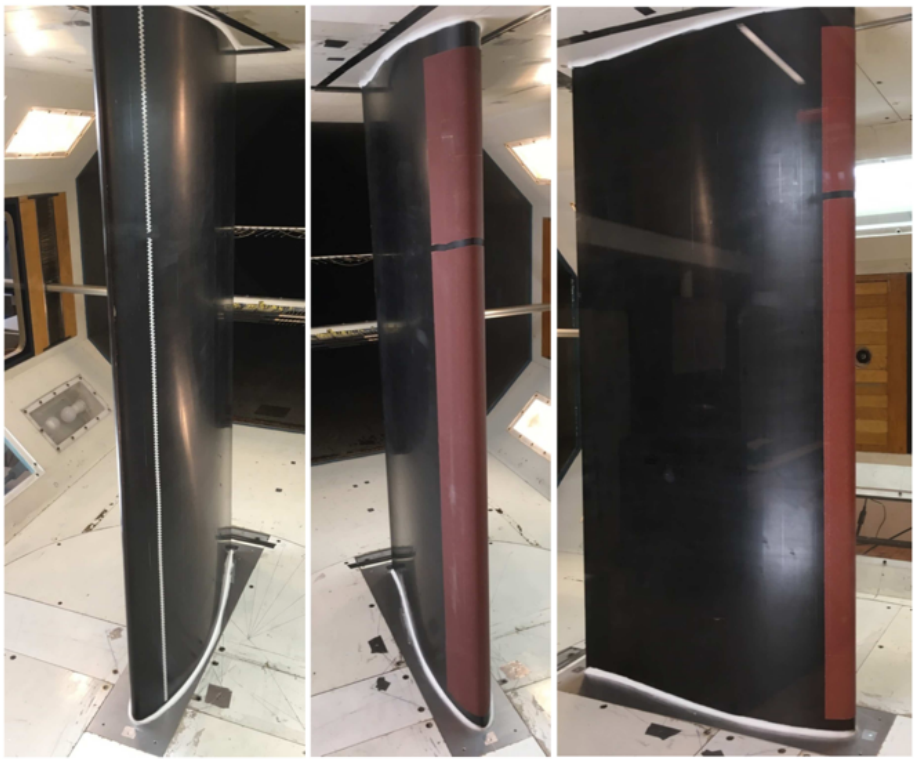
Aerodynamic Load Prediction via Machine Learning Regression of Wind Tunnel Data

John Wylie

21 April 2023



Select leading edge roughness test cases of the NACA $63_3 - 418$ airfoil in the TU Delft LST wind tunnel ([1]).

0.1 Executive Summary

The present study implemented a series of fully-connected artificial neural networks (ANN) to predict the lift and drag forces and the pitching moment experienced by a NACA $63_3 - 418$ wind turbine airfoil section in the TU Delft low-speed, low-turbulence wind tunnel. The wind tunnel data was gathered as part of the second call for experiments by the European Union's Integrated Research Program on Wind Energy. Individuals from research groups in the Netherlands (ECN and TU Delft), Denmark (DTU Wind Energy), and Spain (CENER). They conducted a rigorous series of experimental campaigns in-situ at a wind farm in Wieringermeer, NL and ex situ in the wind tunnel at TU Delft. This project aims to demonstrate a method for cutting down on costly wind tunnel experiment time by providing a framework for interpolating known test parameters and predicting the load and moment measurements on the airfoil for those cases. This project primarily utilized pressure port data and test parameters from the wind tunnel testing to conduct the regression of the target feature vector ($[C_l, C_d, C_m]$). Using an ANN with nearly 700,000 trainable weights, a trivial regression of the target vector using the pressure data was conducted with mean squared error (MSE) losses O(1e-2) achieved. This was not particularly valuable since it requires data collection, and a numerical integration readily yields the forces and moments from the airfoil when the pressures and geometry are known. However, a similar ANN with a larger input feature vector (8 test parameter features versus the 64 pressure port features previously used) achieved a comparable result. This new ANN trained over 10 million weight parameters to achieve MSE losses of roughly 0.005 in the validation set. Several hyper parameters for the test parameter ANN were tuned including the number of neurons per layer, the number of hidden layers, the number of epochs, activation functions, optimizers, kernel regularization options, and dropout layers. The most critical hyper parameters proved to be the learning rate, the number of neurons/hidden layers, and the activation functions, all of which critically affected the fidelity of the predictions to a significant degree. The ANNs were implemented in a Python code using the keras sequential class from Google's tensorflow package. Before implementing the neural networks, the data was first downloaded from the open-source database online and rearranged. New features regarding the roughness elements attached to the airfoil leading edge were generated based on the test case information provided, and data were rescaled and reformatted into several Python Pandas data frames and NumPy arrays for easier manipulation. The data was then visualized using the Seaborn pairplot method and fed into the ANNs for regression. The entire process revealed that careful tuning of the hyper parameters is absolutely necessary for a credible machine learning outcome. The trivial task of using the pressure data for the regression was much less difficult to achieve with the machine learning strategies implemented. On the other hand, the more noteworthy task of predicting the target vector loads and moment based on the test parameter information alone was much more involved of a task. A large ANN with orders of magnitude longer run time achieved a comparable result.

The results were validated using the MSE loss function and qualitatively by comparing them to the input data loads and moment they were intended to predict and replicate. The main difficulty of the ANNs was in predicting the more nonlinear interactions happening with complex flow interactions at or near stall conditions. The loading at high angles of attack proved more elusive for these models compared to the low angle of attack trends to which both ANNs managed to adhere. Overall, the project proves that machine learning provides great potential for expediting research efforts on the experimental end.

0.2 Introduction

Wind turbine blade fouling or icing leads to worsened performance and decreased power output. Dirt, dust, insects, ice, and surface wear are common culprits. As such it is paramount for scientists to understand the implications of these factors, particularly as they affect the thrust generated by wind turbine blades. Numerous papers in the literature have covered these effects. For instance, Sagol et al. reviewed wind tunnel tests and numerical simulations from many studies on roughness-induced transition on airfoils and wind turbine blades. In general, contamination above a critical size caused the premature transition to turbulence on the blade surfaces increasing skin friction drag ([2]). Several research groups have simulated the contaminants using sand and other grit adhered to the region near the blade's leading edge. Icing itself is an involved coupled problem between heat transfer and aerodynamics. Wallenius & Lehtomaki present a review of the challenges and issues with icing on airfoils, particularly cold weather wind energy ([3]). The detriments to energy production alone merit a better understanding of the problem of airfoil surface roughness and contamination. Models or techniques of predicting the adverse effects on aerodynamic coefficients like lift and drag are then of significant value to stakeholders in the wind energy sector.

Historically, machine learning has been used to predict regression-type problems in aerodynamics. One classic example is the handling of the airfoil self-noise dataset. This was generated and documented by Brooks, Pope, and Marcolini in 1989 ([4]). Brooks et al. collected the data in the NASA Langley wind tunnel and created a semi-empirical computer model for predicting the noise level of a given configuration. The data was later regressed by Lau et al. using an artificial neural network to great success ([5]). Such instances of machine learning success in a complicated field such as fluid mechanics/experimental aerodynamics proves that machine learning is fit to solve tangential problems such as load and moment regression.

This project focuses on a particular dataset of measurements from a hybrid experimental campaign conducted in-situ on a wind turbine in the field as well as the TU Delft low-speed, low-turbulence wind tunnel. Wind tunnel tests, the focus of this work, were conducted over a plethora of angles of attack, free stream velocities, and roughness sizes covering various extents of the leading edge of a NACA $63_3 - 418$ wind turbine airfoil ([6]). The wind tunnel experiments are detailed more in Pires et al. ([1]). They used static pressure ports and a pitot static probe rake in the model wake to determine lift and drag, respectively. Kulite dynamic pressure sensors are also employed for greater characterization of the fluid flow. This machine learning project applies an artificial neural network (ANN) machine learning model to the data to tackle the regression problem of predicting the loading and pitching moment based on the feature variables (e.g., free stream velocity, angle of attack, roughness type, etc.). The approach used in this effort focuses mostly on careful construction of an effective fully connected ANN to tackle the regression task at hand. The several hyper parameters for the ANN are considered, and each are tuned with multiple values

and configurations chosen for each.

0.3 Problem Definition

In addition to understanding wind turbine setbacks such as icing, contamination, and surface deterioration, another motivation for the project comes from the experimental side. Wind tunnel testing is an arduous and tedious process of collection large amounts of data for later analysis. To conduct meaningful analyses, large amounts of data and comprehensive parameter sweeps are typically used to capture behavior of the flow fields and models as well as to meet the Nyquist stability criterion and maintain statistically significant dataset sizes. For all experimental endeavors, resources are limited: either tunnel hours, funding, manpower, or power and compressor reservoirs are finite, leading to an inability of researchers to explore every avenue and opportunity. As such, it is of particular interest to explore machine learning methods of duplicating or mimicking actual tests. Load cell tests provide a promising avenue for this due to the condensed form of the results. Three-component forces and loads are the result of sampling periods with biasing of the load cells required frequently between runs. For airfoil models, this collapses to two force components and one load moment. This simple output for complicated input parameters lends itself well to neural networks and other machine learning techniques that well compound variable inputs into outputs of the desired size, often on the smaller side. As such, this work utilizes a fully connected ANN to predict the lift and drag forces and the pitching moment of a wind turbine airfoil section in a wind tunnel for several hundred test cases with varying flow speed, angle of attack, and leading edge roughness configurations.

0.4 Methods and Procedure

The machine learning procedure used in the present study is composed of several steps:

- Data Pre-Processing
- Data Visualization
- Construction of the ANN
- Hyper Parameter Tuning

These steps are detailed in the present section as well as in the Results and Discussion section.

0.4.1 Data Pre-Processing

The data were separately provided in text files sorted based on the leading edge roughness cases considered. Each data file contained all the applicable

features for each test including loads, pitching moment, and pressure data. Each test typically consisted of a sweep of several angles of attack. The data were accumulated from all test runs, unneeded features were removed (e.g., duplicate loads and pitching moment), and new features with roughness information were added. The data was sorted into target vectors, input test parameters, and input pressure data. All sets were rescaled to span the range [0,1] as shown in Figure 1.

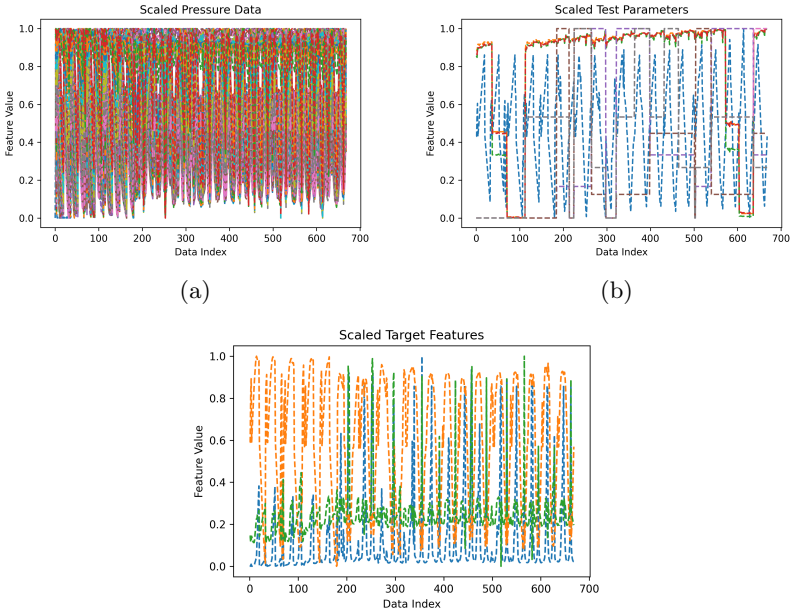


Figure 1: Features for the pressure data (a), test parameters (b), and target vector (c) rescaled to the range [0,1].

0.4.2 Data Visualization

See the following section for more information on data visualization.

0.4.3 Construction of the ANN

The details of the final ANN for the pressure data are shown in Table 1. The total model consisted of 668,803 parameters, or weights, and took roughly 0.6 milliseconds to compile, train, validate, and predict.

Property	Description
Input Layer	400 neurons; 64-column input vector
Hidden Layers	400 neurons per layer; 4 layers
Output Layer	3 neurons; 3-column output vector
Learning Rate	0.004
Activation Function	Rectified linear unit (ReLu)
Optimizer	Stochastic gradient descent (SGD)
Callbacks	Early stopping (patience = 50)
Regularization	L2 kernel regularization (1e-4)
Loss metric	MSE
Epochs	400

Table 1: Details of the final ANN model for the pressure data.

The details of the final ANN model for the test parameters are shown in Table 2. The total model consisted of 10,022,003 parameters, or weights, and took roughly 4.4 minutes to compile, train, validate, and predict.

Property	Description
Input Layer	1000 neurons; 8-column input vector
Hidden Layers	1000 neurons per layer; 10 layers
Output Layer	3 neurons; 3-column output vector
Learning Rate	0.004
Activation Function	Rectified linear unit (ReLu)
Optimizer	Stochastic gradient descent (SGD)
Callbacks	Early stopping (patience = 50)
Regularization	L2 kernel regularization (1e-4)
Loss metric	MSE
Epochs	1000

Table 2: Details of the final ANN model for the test parameters.

0.4.4 Hyper Parameter Tuning

Hyper parameter tuning was needed to achieve satisfactory regression of the data for the test parameter features. For the pressure data, a smaller version of the test parameter model was used with 4 hidden layers with 400 neurons each. Results of the hyper parameter tuning are shown in the Results and Discussion section. To streamline the tuning process, a smaller ANN was used on the pressure data features with 6.7% of the total number of weights of the larger test parameter ANN model. The tuning results were evaluated by comparing cases with loss time series results across the epochs and with heat maps of losses for all cases. In addition, the output vectors were plotted against the ground truth test data to compare the regressions. The most important hyper parameter tuned was the learning rate. When the rate was too high, all the

predictions collapsed to a single value yielding meaningless results that failed to match the trends. Other significant hyper parameters included the network architecture (e.g., number of layers and neurons per layer). Table summarizes the hyper parameter tuning procedure.

Hyper Parameter	**Cases Considered**
n_{layers}	[1, 2, 3, 4], [4, 5, 6], [6, 8, 10]
$n_{neurons}$	[50, 100, 200], [200, 300, 400], [400, 600, 1000]
n_{epoch}	[200, 300, 400]
Learning Rate (α)	[0.001, 0.002, 0.003, 0.004]
Optimizer	[SGD, Adam, RMSprop]
Activation Function	[ReLu, sigmoid, tanh, SeLu, exponential]
Dropout	[Before HL 1, after HLs 1, 2, 3, & 4]
Kernel Regularization	[None, L1, L2, L1+L2]

Table 3: Details of the final ANN model for the pressure data.

0.5 Dataset and Visualization

The main difficulty of using this dataset is its sheer size. Figure 2 shows the test matrix used for the bulk of the aerodynamic load testing. The total files comprise 3.30 GB of data. For simplicity, only the wind tunnel load data (excluding the microphone time series data) is considered. Figure 2 shows a summary of the wind tunnel test cases considered in this study.

Polar	Reynolds Nº	Surface condition					AoA range	Wake tr. Range (mm.)	Polar name
			Grain Size		LE length covered				
			P-grade	Av. Size (μm)	Lower	Upper			
1	3M	Clean	-	-	-	-	-12 to 15	0-100	N63418-Re3clean
2	2M	Clean	-	-	-	-	-13 to 15	140-220	N63418-Re2clean
3	1M	Clean	-	-	-	-	-15 to 15	140-220	N63418-p1Clean
4	3M	Tripped t1	-	-	8%	8%	-12 to 15	140-240	N63418-p3Trip025
5	3M	Tripped t2	-	-	8%	8%	-15 to 15		N63418-p3Trip06
6	3M	Distr. Rough	40	425	8%	8%	-14 to 15	220-300	N63418-p3wrapP40 N63418-p3wrapP40-3
7	3M	Distr. Rough	40	425	15%	15%	-15 to 15	150-240	N63418-p3wrapP40to15%
8	3M	Distr. Rough	240	53	4%	4%	-15 to 15	170-270	N63418-p3wrapP240_4% N63418-p3wrapP240_4%_3
9	3M	Distr. Rough	240	53	8%	8%	-14 to 16	150-250	N63418-p3wrapP240_8%
10	3M	Distr. Rough	240	53	15%	15%	-13 to 15	200-300	N63418-p3wrapP240_15%
11	3M	Distr. Rough	80	190	8%	8%	-13 to 15	140-240	N63418-p3wrapP80_8%
12	3M	Distr. Rough	80	190	15%	15%	-14 to 16	200-300	N63418-p3wrapP80_15%
13	3M	Distr. Rough	80	190	4%	4%	-13 to 16	180-280	N63418-p3wrapP80_4% N63-418-p3wrapP80_4%-2
14	3M	Distr. Rough	40	425	4%	4%	-13 to 17	150-220	N63-418-p3wrap_P40_4%
15	3M	Distr. Rough	240	53	8%	8%	-13.5 to 16	180-260	N63-418-p3wrap_P240_8%_onepart
16	2M	Distr. Rough	240	53	8%	8%	-13 to 18	180-260	N63-418-p2wrap_P240_8%_onepart
17	1M	Distr. Rough	240	53	8%	8%	-15 to 20	180-260	N63-418-p1wrap_P240_8%_onepart
18	3M	Distr. Rough	80	190	15%	4%	-13 to 17	220-300	N63-418-p3wrap_P80_15%ls_4%us

t1 0.25 mm 8% upper & lower
t2 0.6 mm 8% upper & lower

Figure 2: Overview of the test matrix used in the IRPWIND airfoil roughness dataset ([1] Pires et al.).

I combined cases from different tests and arranged the data with additional

features contained in the test descriptions detailing the roughness qualities. Compiling all of these, I have 669 data points with 72 features. The actual data has more; multiple values for the lift, drag, and moment coefficients are provided from multiple calculation methods, but only one set are used in this investigation. The target features of interest are the lift, drag, and moment coefficients. The input features are split into two groups: the test parameters and the pressure data. The test parameters include the angle of attack, the free stream Mach number, velocity, and dynamic pressure, the roughness grit and grade, and the percentages of the upper and lower surfaces covered at the leading edge. To visualize the dataset, I used the pairplot function in the Seaborn Python package to separately visualize the test parameters and the pressure data each. Figure 3 shows the pairplot of the eight test parameters. A few conclusions can be drawn from this data. First, most variables are in rows or columns on the plots indicating that they were parameters swept through the different tests to explore the effect of each individually. However, this means individual relations, coherence, and behavior are harder to discern visually. Second, some variables referencing flow speed are closely correlated, and the plots show nearly linear or second order polynomial fits. For this project, feature reduction was not conducted since the emphasis was instead placed on the hyper parameter tuning. If feature reduction or importance methods were to be done, it is quite likely that several of these features could be represented by a single one.

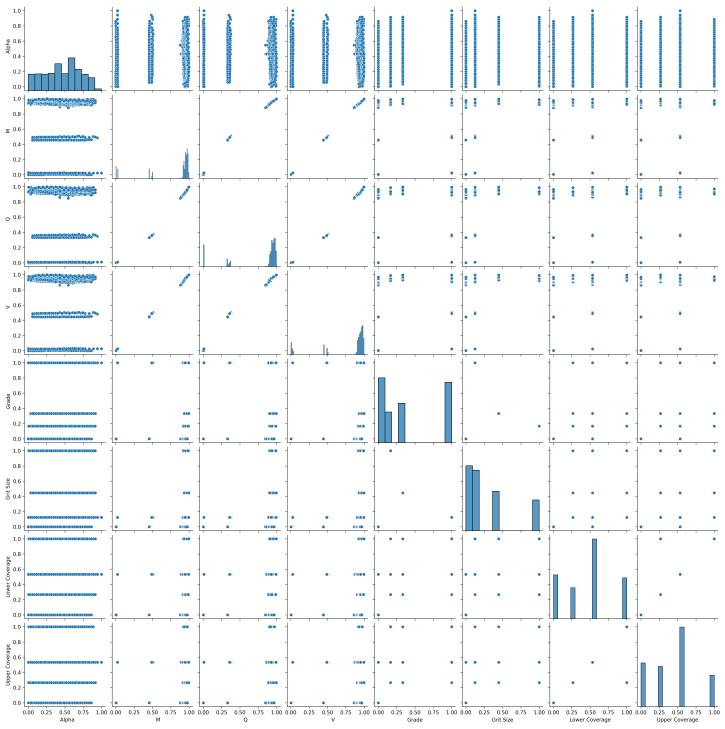


Figure 3: Seaborn pairplot of the test parameters.

Figure 4 displays the pairplot of the pressure data. The 64 by 64 matrix of plots is not terribly usefully in deciphering particular values of most pressure ports around the model. Nevertheless, the smooth transition of distribution between consecutive plots across the matrix indicates that the pressure ports were intelligently placed in the tests; there is no discontinuous jump in any of the ports. The static pressure port data is comprised of 23 pressure side measurements, 40 suction side measurements, and the position of the wake rake used to compute the drag downstream.

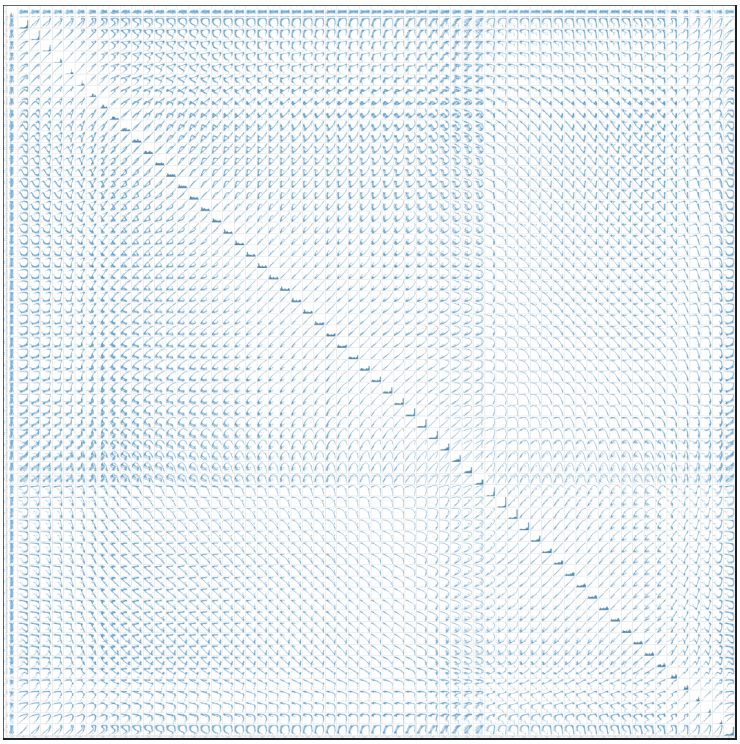


Figure 4: Seaborn pairplot of the pressure data.

0.6 Results and Discussion

To decide on the final architecture and hyper parameters, a comprehensive set of tests was conducted on the test parameter ANN to evaluate the performance of each. This process is discussed in the Hyper Parameter Tuning subsection. The final results are shown and evaluated in the Final Results subsection.

0.6.1 Hyper Parameter Tuning

The results of tuning the hyper parameters mentioned in the Methods and Procedure section are shown in the following section. Results are shown in three formats: time histories of the MSE loss function over the epochs for each case, heat maps of the final loss values for training and validation splits (70%/30% split), and drag polar plots of the predicted and actual lift and drag coefficients. To exemplify the effect of poor hyper parameter tuning, the results of a poorly tuned model are shown in Figure 5.

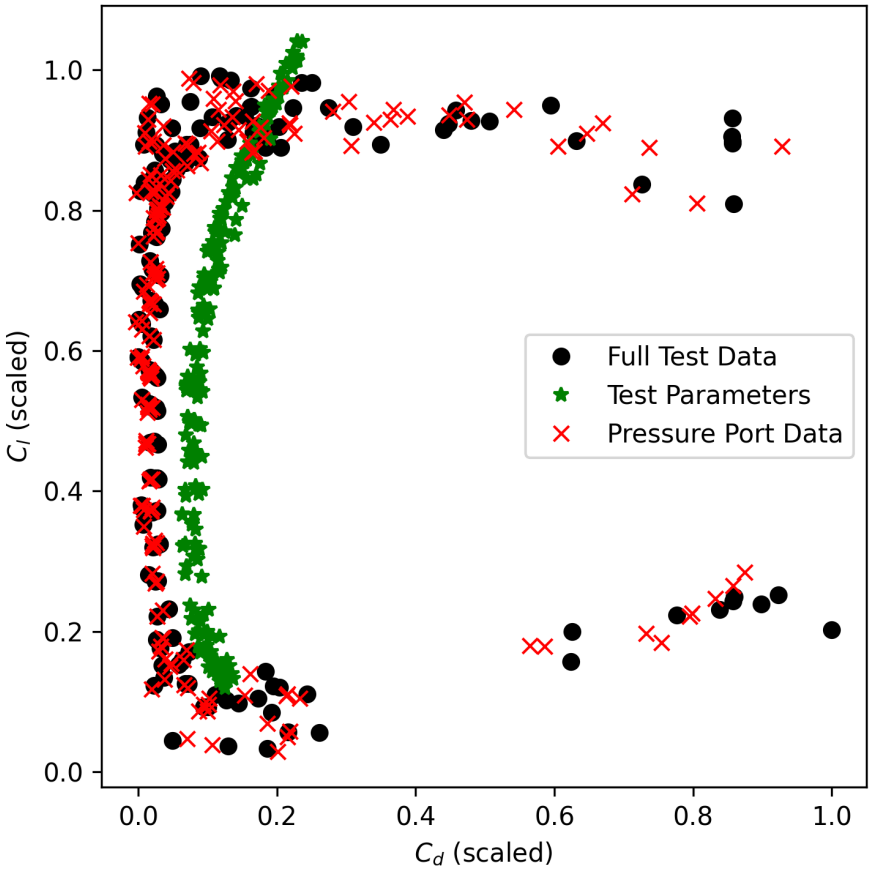


Figure 5: Drag polar of predicted C_l and C_d values from a poorly tuned iteration of the ANN for the test parameters (green star) compared to the test data (black circles). The pressure data predictions for an appropriate model are shown in the red exes.

The predicted values do not well conform to the ground truth C_l and C_d values from the data. Note that the pressure data ANN was much more readily tuned to the data. The task of using an ANN to predict the loads from pressure data is rather trivial for two reasons. For one, the loads and pitching moment of an airfoil can be readily obtained from pressure port data through classical integration of the correct pressure force components when the airfoil profile is known. In this sense, using machine learning achieves a worse end than performing the classical calculation. Second, predicting the target features from the pressure data defeats the purpose of reducing experiment time. The pressure data must still be taken from experiments while using just the test parameters would allow researchers to bypass some of the tests. Thus, only the

ANN receiving test parameter features was tuned properly and with significant diligence since it was more aligned with the motivation of this work.

It is worth noting that, for a majority of the hyper parameter tuning, the ANNs tested consisted of a smaller structure like the ANN with the pressure data features with a number of parameters O(1e5) in the interest of time. Only for the final tests was an ANN with a number of parameters O(1e7) tested since it was an order of magnitude more expensive computationally.

Number of Layers and Neurons per Layer

The following sets of tests in Figures 6 to 8, 9 to 11, and 12 to 14 demonstrate the results of testing the number of layers [1, 2, 3, 4], [4, 5, 6], and [6, 8, 10], respectively, and the numbers of neurons [50, 100, 200], [200, 300, 400], and [400, 600, 1000], respectively. These tests determined that the best configuration would have ten hidden layers with 1000 neurons per layer based on the drag polars and less so the heat maps.

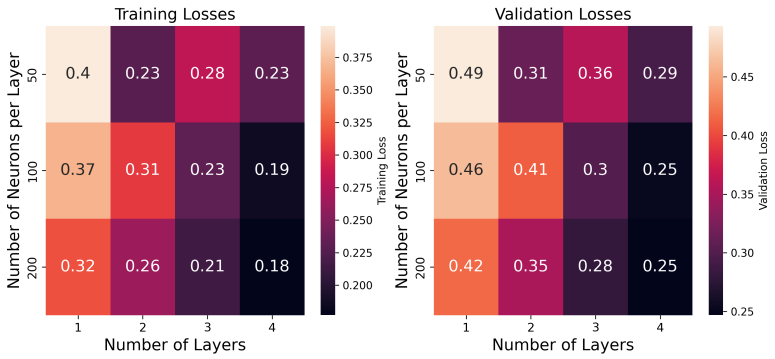


Figure 6: Time histories of the training and validation losses for the number of layers and neurons per layer test cases.

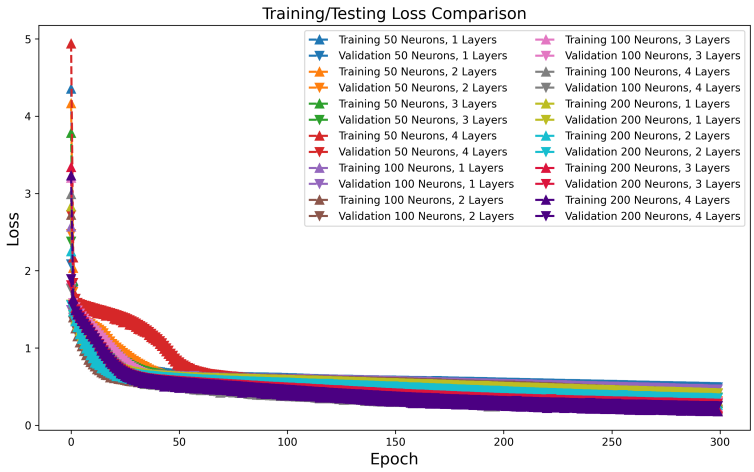


Figure 7: Heat maps of the training and validation losses for the number of layers and neurons per layer test cases.

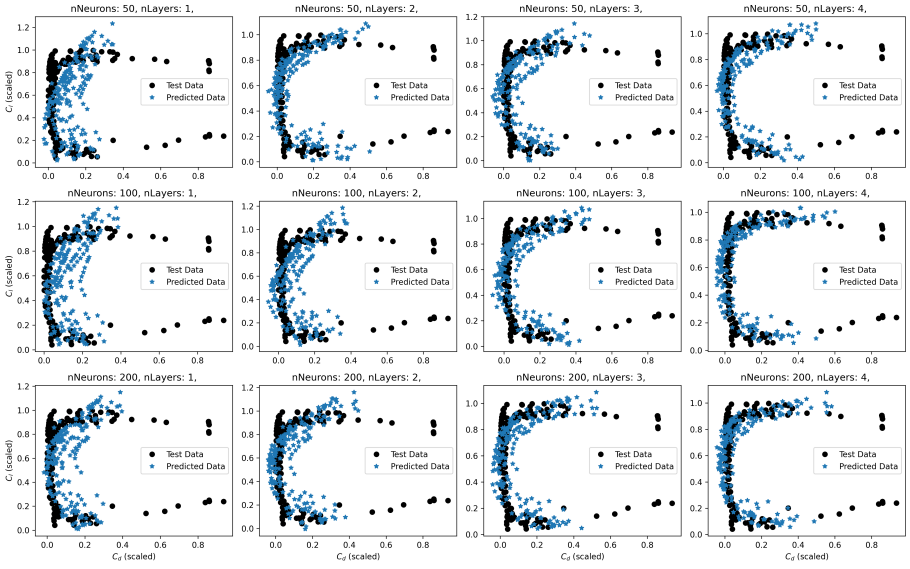


Figure 8: Drag polar of the predicted and actual C_l and C_d values for the number of layers and neurons per layer test cases.

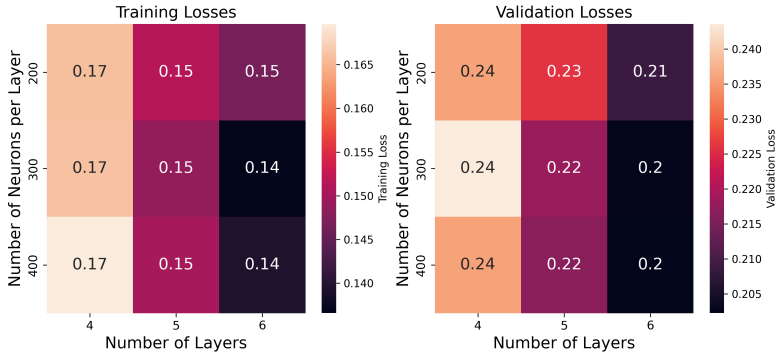


Figure 9: Time histories of the training and validation losses for the number of layers and neurons per layer test cases.

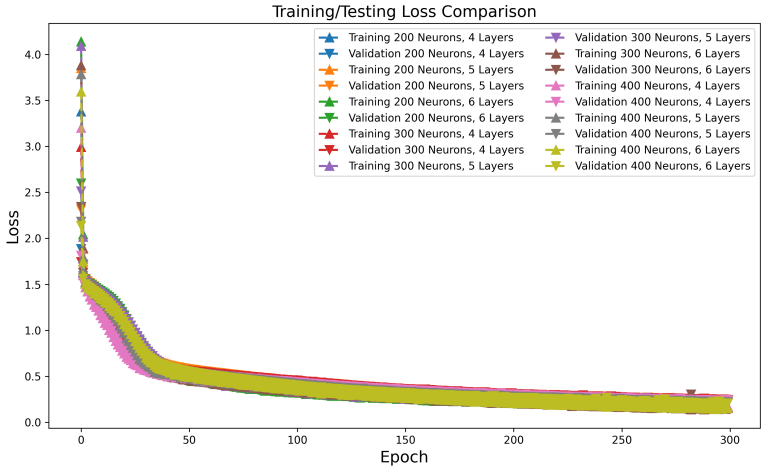


Figure 10: Heat maps of the training and validation losses for the number of layers and neurons per layer test cases.

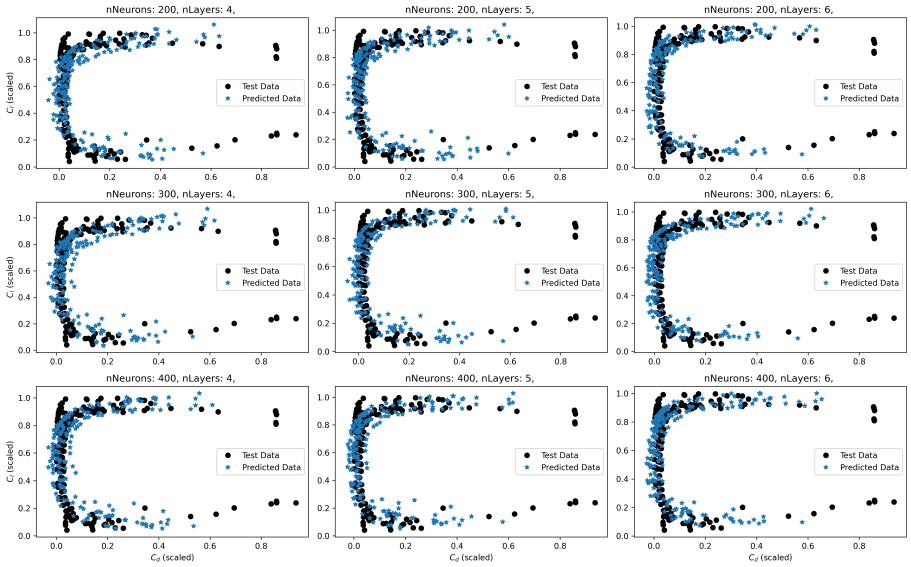


Figure 11: Drag polar of the predicted and actual C_l and C_d values for the number of layers and neurons per layer test cases.

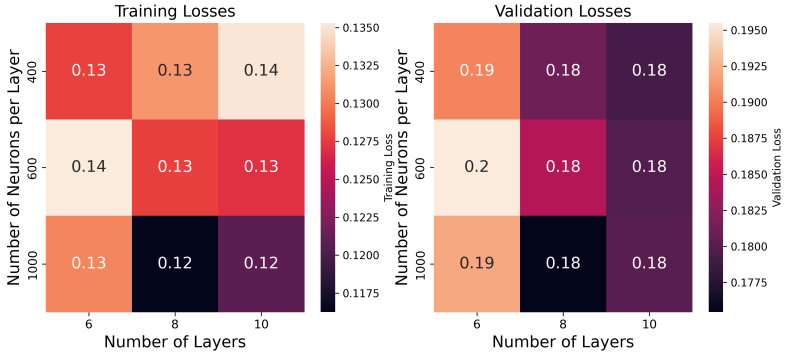


Figure 12: Time histories of the training and validation losses for the number of layers and neurons per layer test cases.

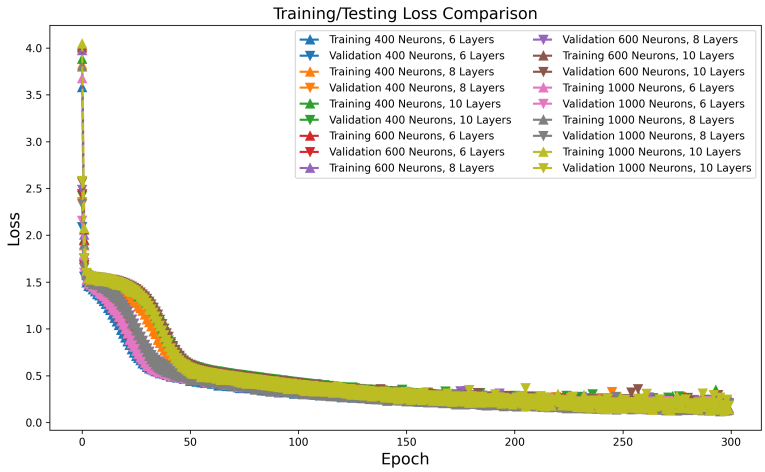


Figure 13: Heat maps of the training and validation losses for the number of layers and neurons per layer test cases.

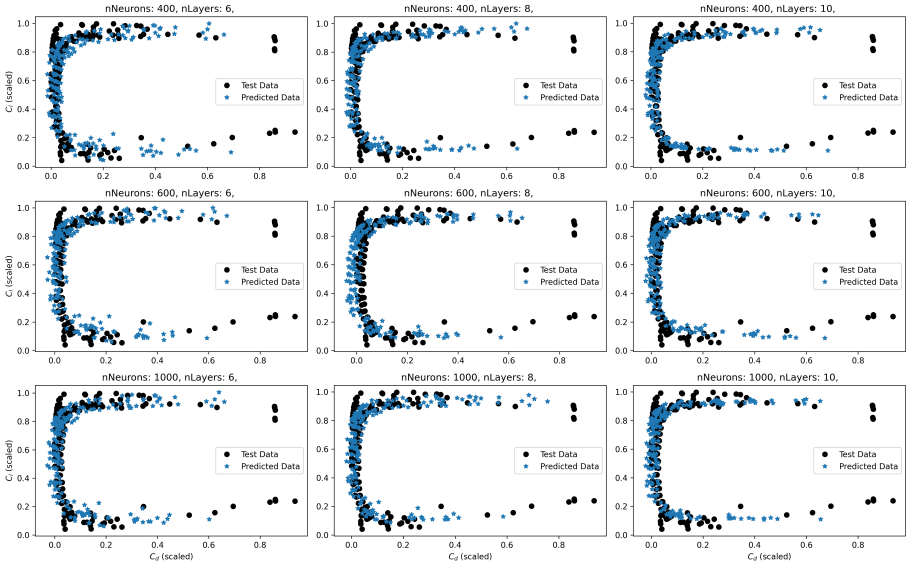


Figure 14: Drag polar of the predicted and actual C_l and C_d values for the number of layers and neurons per layer test cases.

Learning Rate and Number of Epochs

The following sets of tests in Figures 15 to 17 demonstrate the results of testing [200, 300, 400] epochs and learning rates of [0.001, 0.002, 0.003, 0.004]. These tests determined that the best configuration would have a minimum of 400 epochs (further tests found 1000 might be preferable) and $\alpha = 0.004$.

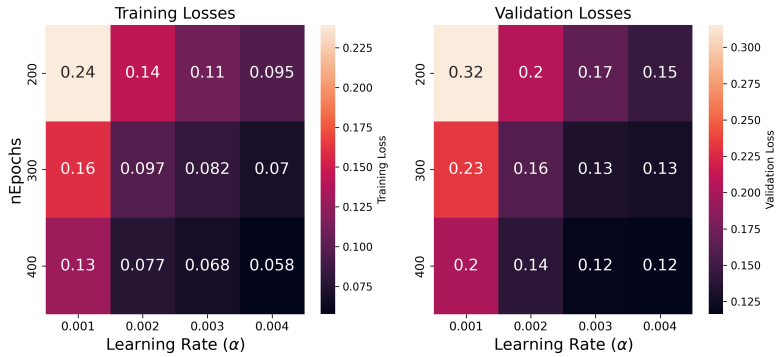


Figure 15: Time histories of the training and validation losses for the learning rate and number of epochs test cases.

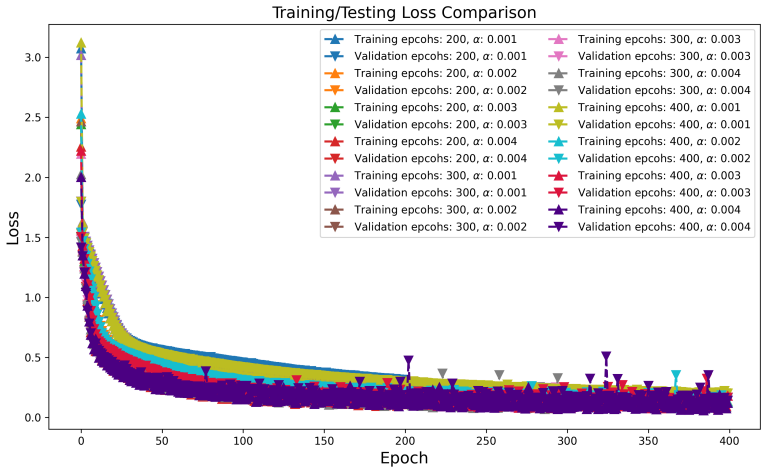


Figure 16: Heat maps of the training and validation losses for the learning rate and number of epochs test cases.

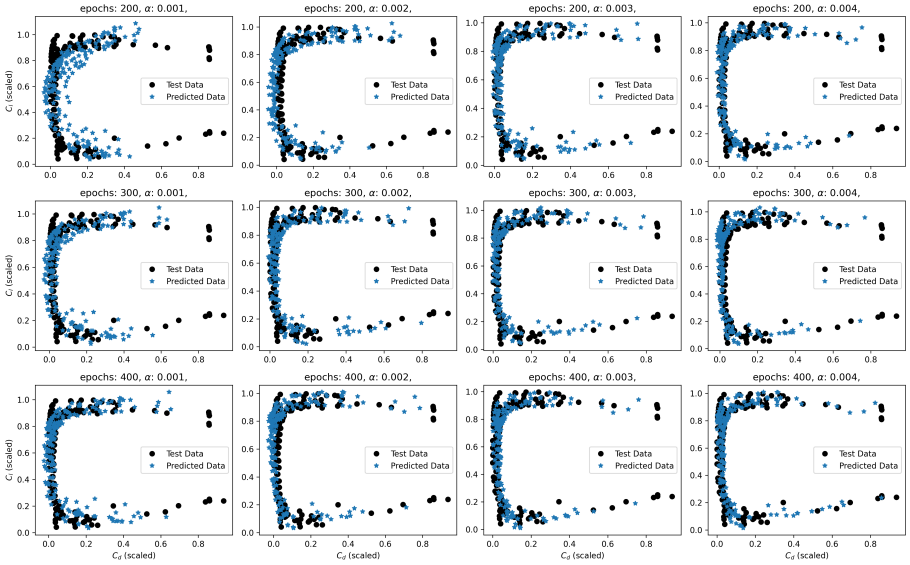


Figure 17: Drag polar of the predicted and actual C_l and C_d values for the learning rate and number of epochs test cases.

Kernel Regularization

The following sets of tests in Figures 18 to 20 demonstrate the results of testing kernel regularization types L1, L2, and L1 and L2. These tests determined that the best configuration would utilize L2 regularization.

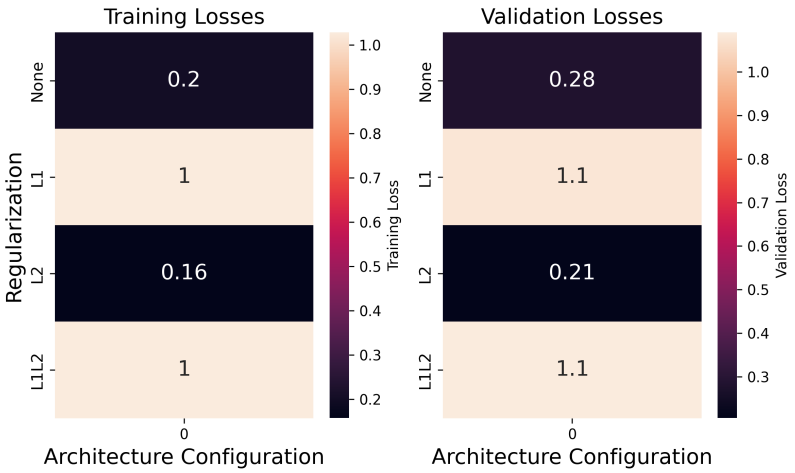


Figure 18: Time histories of the training and validation losses for the regularization test cases.

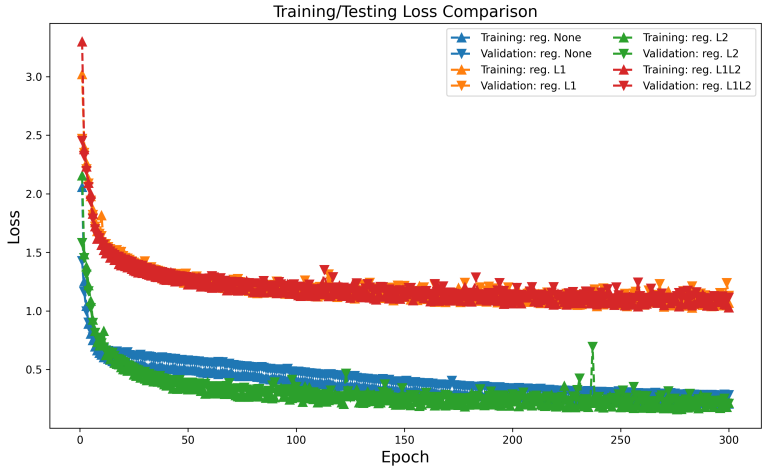


Figure 19: Heat maps of the training and validation losses for the regularization test cases.

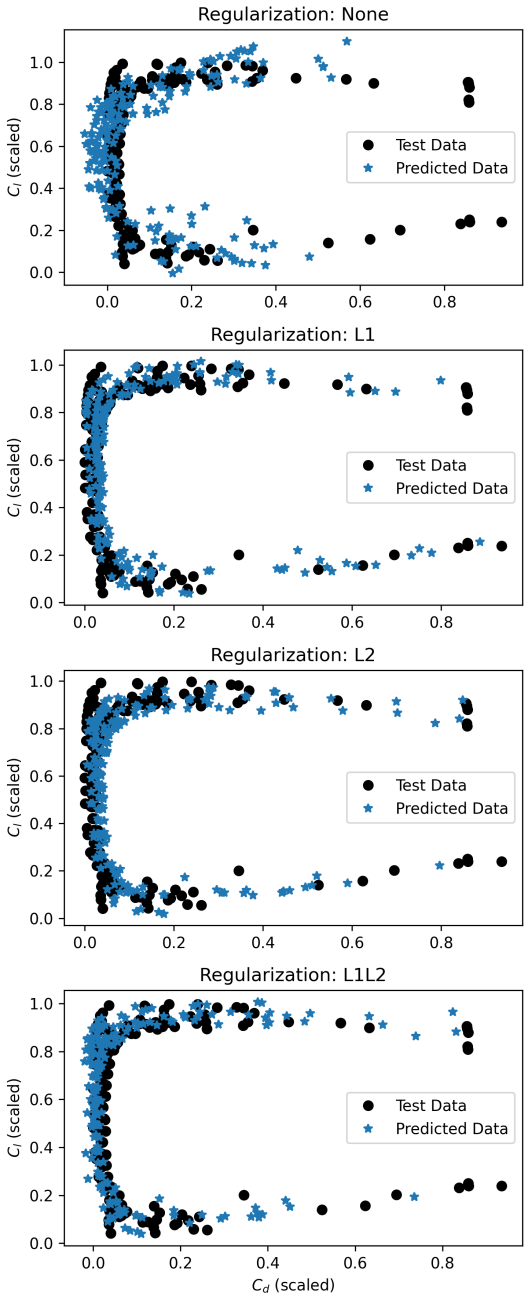


Figure 20: Drag polar of the predicted and actual C_l and C_d values for the regularization test cases.

Activation Functions and Optimizers

The following sets of tests in Figures 21 to 23 demonstrate the results of the testing stochastic gradient descent (SGD), Adam, and RMSprop optimizers and ReLu, sigmoid, hyperbolic tangent (tanh), SeLu, and exponential activation functions. These tests determined that the best configuration would utilize SGD with ReLu nonlinear activation. Note that in Figure 22, results are excluded for NaN loss values. Several of the time histories in Figure 21 are chaotic and high in MSE, indicating that they are not well suited to this problem. It was worth considering this because activation function and optimizer ended up impacting the data significantly, but the original ones chosen were the same as the final ones.

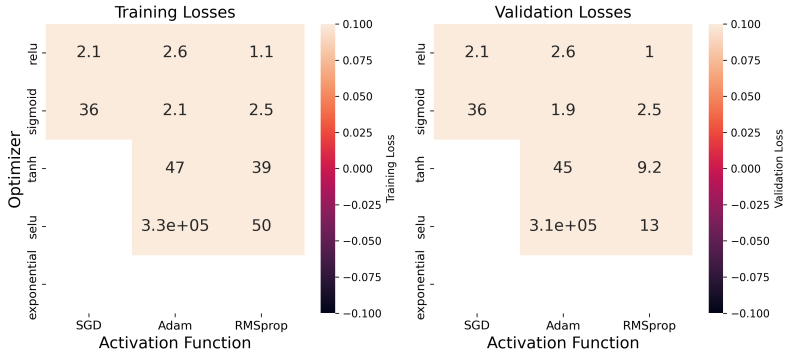


Figure 21: Time histories of the training and validation losses for the activation function and optimizer test cases.

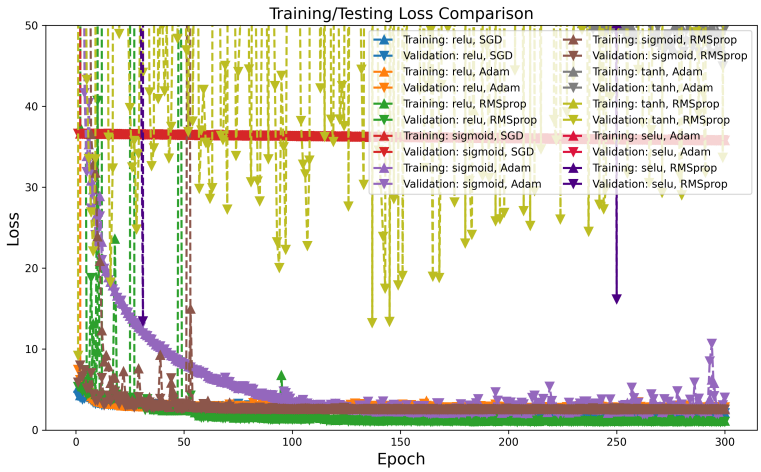


Figure 22: Heat maps of the training and validation losses for the activation function and optimizer test cases.

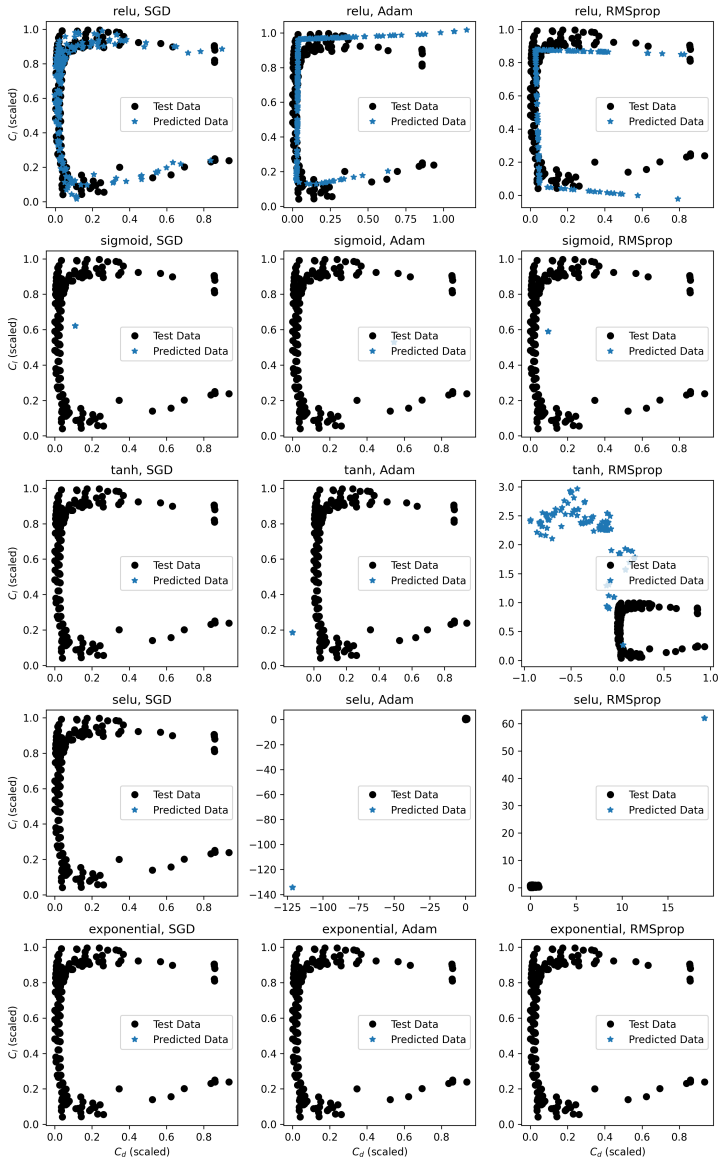


Figure 23: Drag polar of the predicted and actual C_l and C_d values for the activation function and optimizer test cases.

Dropout Layers

The following sets of tests in Figures 24 to 26 demonstrate the results of adding a dropout layer before hidden layer one, after hidden layers one, two, three, and four. These tests determined that the best configuration would include

a dropout layer after hidden layer four. Further tests could be conducted to consider the effect of multiple dropout layers.

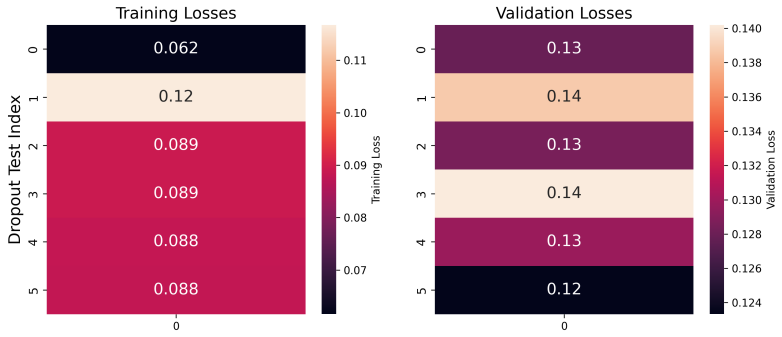


Figure 24: Time histories of the training and validation losses for the dropout test cases.

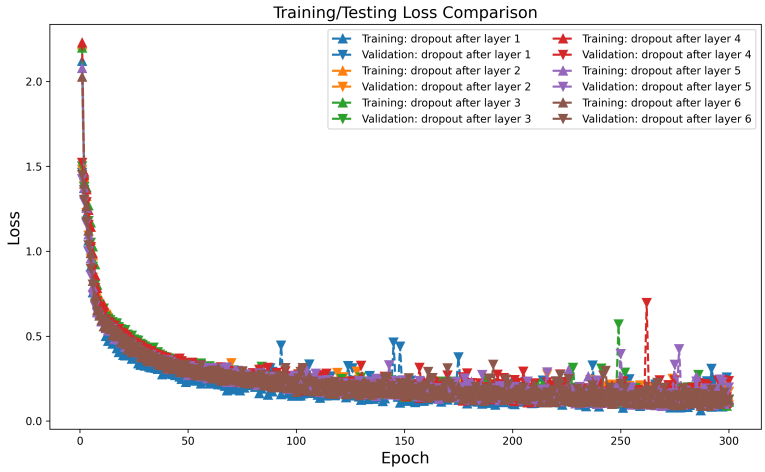


Figure 25: Heat maps of the training and validation losses for the dropout test cases.

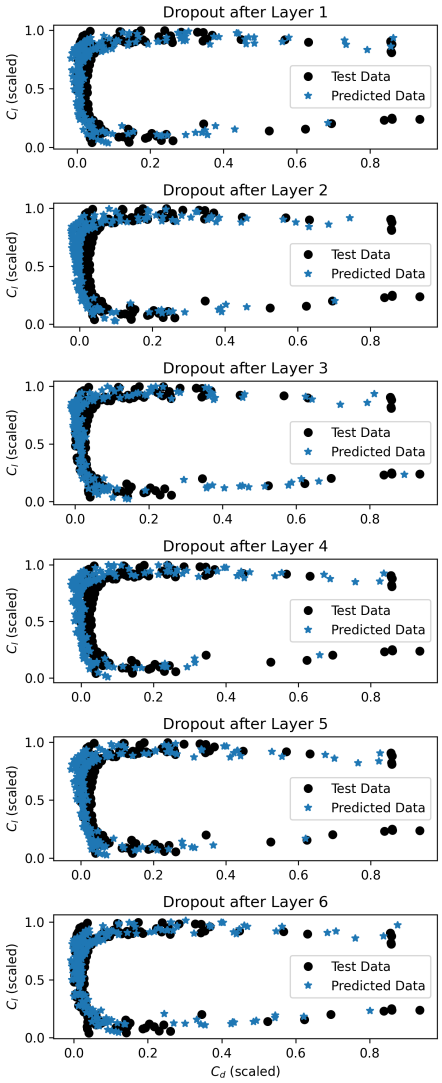


Figure 26: Drag polar of the predicted and actual C_l and C_d values for the dropout test cases.

0.6.2 Final Results

The results of the final models described in the Methods and Procedure section are shown in Figures.

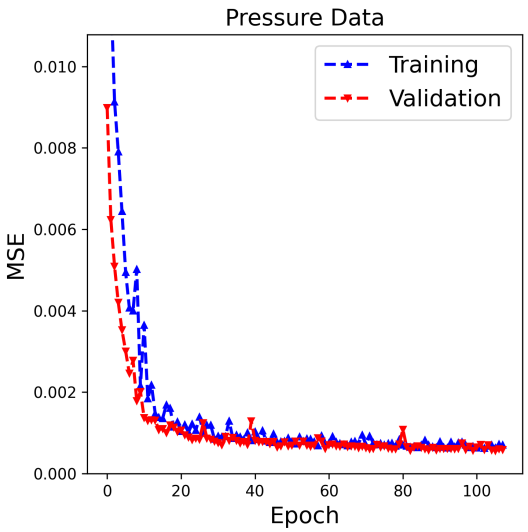


Figure 27: Time histories of the training and validation losses for the final pressure data ANN.

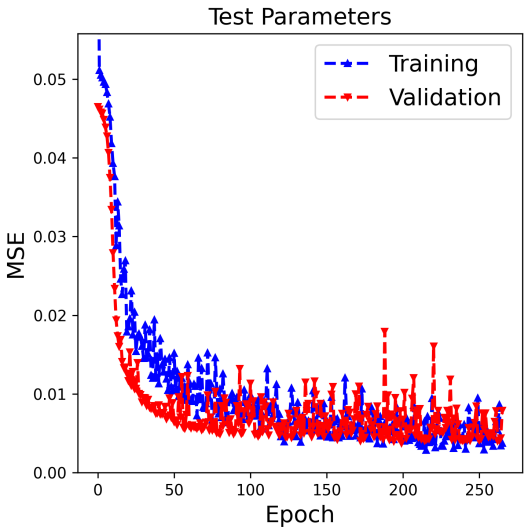


Figure 28: Time histories of the training and validation losses for the final test parameter ANN.

While both the test parameters and pressure data ANNs achieve convergence on low values of the MSE loss function, the test parameter ANN was less effective by an order of magnitude.

The following three figures (Figures 29 to 31) show the comparisons of the target vector predictions for the two ANNs and the actual target vector values from the data for three pairs of target features. It is important to note that both models struggle to well predict the highly nonlinear flow effects near the stall regime on the airfoil. These occur at high angles of attack represented by high values of C_l and C_d.

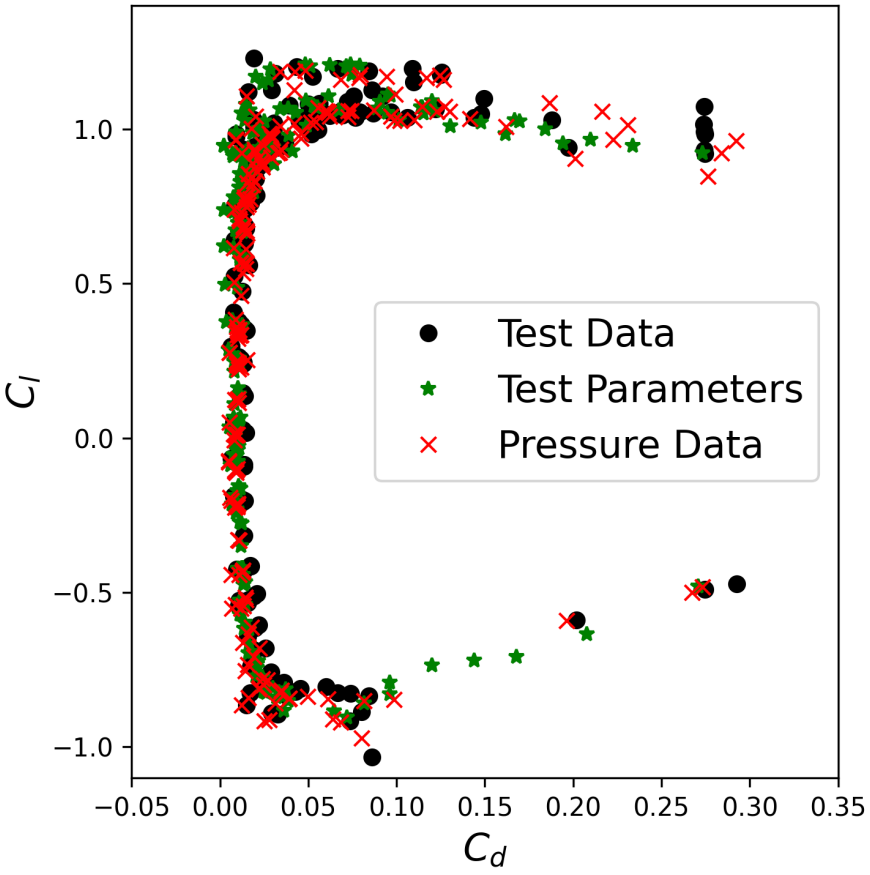


Figure 29: Drag polar of the predictions from the two ANNs and the actual data for C_l and C_d values.

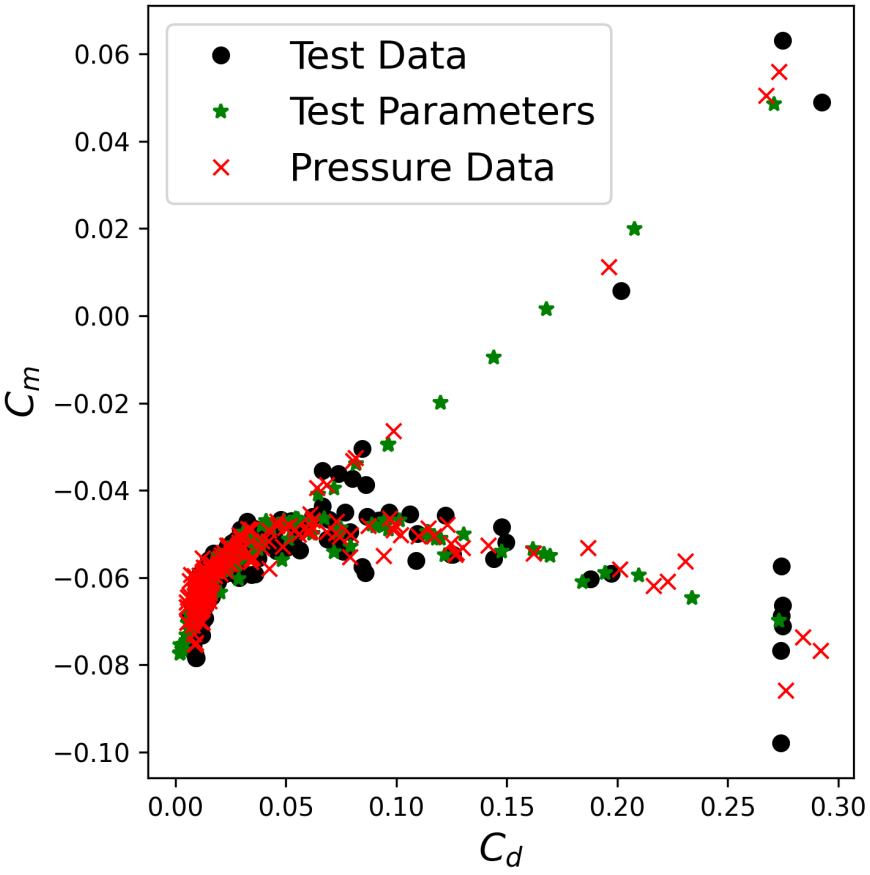


Figure 30: Drag polar of the predictions from the two ANNs and the actual data for C_d and C_m values.

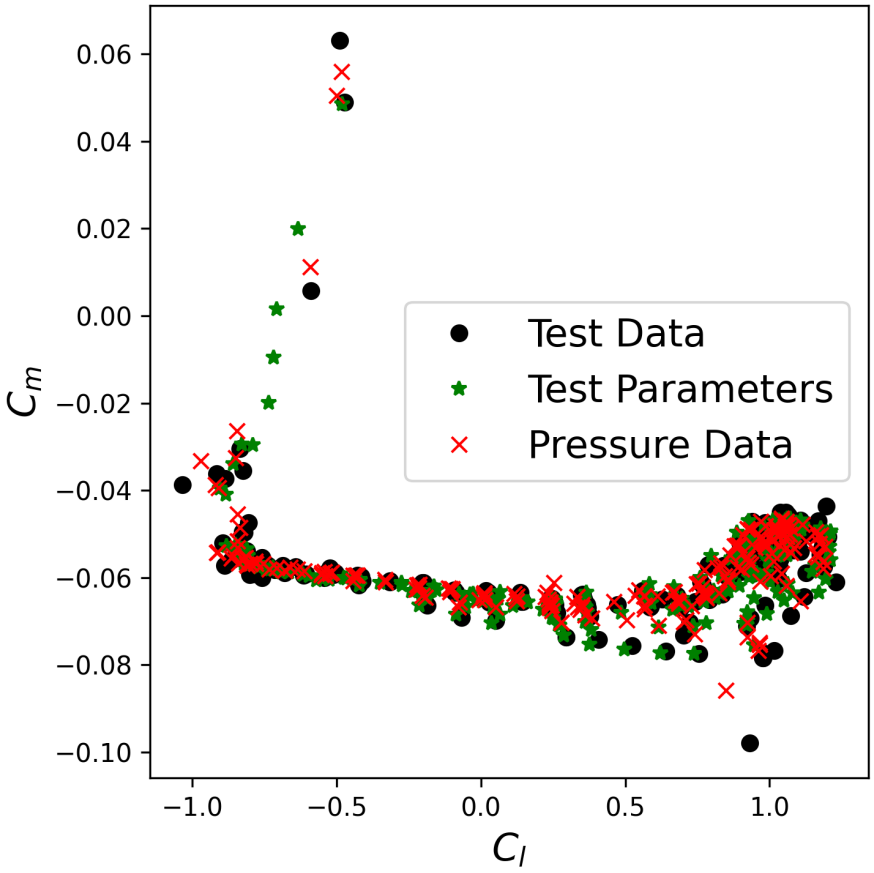


Figure 31: Drag polar of the predictions from the two ANNs and the actual data for C_l and C_m values.

0.7 Conclusion

This study successfully utilized features from both information on the test state, the test parameters, and static pressure port measurements, the pressure data, to predict the lift, drag, and moment coefficients of the wind turbine airfoil model in the wind tunnel. The test parameters ANN utilized roughly 10 million elements to achieve an MSE value one order of magnitude higher than that of the pressure data ANN, which utilized 1.5 orders of magnitude fewer model parameters and multiple orders of magnitude less time to compute. The pressure data ANN requires 64 input features compared to the 8 in the test parameter ANN (though it is worth noting that these numbers could be decreased with recursive feature elimination or principal component analysis). The pressure

data ANN is less valuable as a whole because it requires that experimental data be collected to predict, and it duplicates a know integration method. On the other hand, the test parameter ANN achieves an acceptable fit of the data with no experimental data collection required at a moderate to high computational cost. Despite accuracy concerns, the test parameter ANN quite satisfactorily performs the regression considering the limited information from the input feature vector. Ultimately, machine learning is a strong contender for improving the workflow of experimental aerodynamicists involved with load cell measurements in wind tunnels. In short, we may not yet be eliminating the majority of wind tunnel tests, but we are remarkably close to predicting simple load cell measurements.

0.8 Data and Code

The dataset can be found on the IRPWIND page of zenodo.org. Additionally, the GitHub repository contains all the relevant codes and data used in this work.

Bibliography

[1] O. Pires, X. Munduate, O. Ceyhan, M. Jacobs, and H. Snel, "Analysis of high reynolds numbers effects on a wind turbine airfoil using 2d wind tunnel test data," *Journal of Physics: Conference Series*, vol. 753, no. 2, p. 022047, sep 2016. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/753/2/022047

[2] E. Sagol, M. Reggio, and A. Ilinca, "Issues concerning roughness on wind turbine blades," *Renewable and Sustainable Energy Reviews*, vol. 23, pp. 514–525, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032113001366

[3] T. Wallenius and V. Lehtomäki, "Overview of cold climate wind energy: challenges, solutions, and future needs," *WIREs Energy and Environment*, vol. 5, no. 2, pp. 128–135, 2016. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wene.170

[4] T. F. Brooks, D. S. Pope, and M. A. Marcolini, "Airfoil self-noise and prediction," *NASA Reference Publication*, no. 1218, 1989.

[5] K. Lau, R. López, and E. Oñate, "A Neural Networks Approach to Aerofoil Noise Prediction," *International Center for Numerical Methods in Engineering*, vol. CIMNE Nº-3, 2009.

[6] O. Pires, X. Munduate, K. Boorsma, and H. Aagaard Madsen, "Experimental Investigation of Surface Roughness Effects and Transition on Wind Turbine Performance Dataset," Jun. 2018. [Online]. Available: https://doi.org/10.5281/zenodo.3482801