Aerodynamic Prediction and Optimization of Airfoils Using Machine Learning Techniques

Submitted by
Liew Li Heng

Department of
Mechanical Engineering

In partial fulfilment of the
requirements for the Degree of
Bachelor of Engineering
National University of Singapore

Session 2020/2021

Summary

This paper investigates a machine learning approach to optimize airfoils, using a Multilayer Perceptron (MLP), a type of Artificial Neural Network (ANN). A preliminary MLP is trained on an existing database, containing flow parameters, airfoil geometry parameters and airfoil aerodynamic coefficients. Subsequently, hyperparameter optimization is performed, to identify optimal hyperparameters such as the number of hidden layers, number of neurons per hidden layer, optimizer, learning rate and batch size. The final MLP is then trained, using the identified optimal hyperparameters.

The final MLP has a 7 layer architecture consisting of 1 input layer, 5 hidden layers and 1 output layer. Predictions made by the final MLP are compared with actual values in the database, as well as 2 airfoils excluded from the database. Results show that the final MLP is able to predict the aerodynamic coefficients very accurately, with a mean squared error of 1.86×10^{-6} on the validation set. Also, the final MLP was able to accurately predict the actual aerodynamic coefficients of 2 airfoils, which are excluded from the database.

The final MLP is then used for airfoil optimization, for both singlepoint and multipoint optimization. Multiple starting airfoils and optimizers are used, to increase the chances of obtaining a global optimum. Numerous airfoils were produced, each performing extremely well, possessing desirable lift and drag coefficients. The results show that this approach is very efficient in terms of computational time, without sacrificing prediction accuracy. It thus constitutes an attractive alternative method in airfoil optimization, especially when existing data is readily available.

Acknowledgements

The author would like to wholeheartedly thank the project supervisor, Professor Shu Chang, for his indispensable guidance and support; and Mr Vinothkumar Sekar, for their discussions and assistance throughout this study.

Table of Contents

List of Figures

List of Tables

List of Symbols

α	Angle of attack
σ	Activation function
a_j^l	Value of neuron j in hidden layer l
a_k^{l-1}	Value of neuron k in hidden layer $l-1$
b_j^l	Bias term of the jth neuron in the lth layer
c	Cost function
C_D	Drag coefficient
C_L	Lift coefficient
$P1$	First digit of NACA 4-digit airfoil
$P2$	Second digit of NACA 4-digit airfoil
$P3$	Third and fourth digits of NACA 4-digit airfoil
Re	Reynolds number
Re_{crit}	Critical Reynolds number
w_{jk}^l	Weight of the connection from the kth neuron in the $(l-1)$th layer to the jth neuron in the lth layer
$C_{L\,fixed}$	Fixed lift coefficient

1 Introduction

Airfoils play an indispensible role in the design of components, such as aircraft wings, helicopter rotor blades and flight control surfaces. The overall performance of these components are heavily dependent on the airfoil characteristics, which are encapsulated by two non-dimensional aerodynamic coefficients [1]. These two aerodynamic coefficients are the lift coefficient (C_L), and drag coefficient (C_D). Since these aerodynamic coefficients largely determine how well an airfoil performs, several approaches have been developed to analyze them.

One conventional approach in predicting these aerodynamic coefficients, is by solving the discretized Reynolds Averaged Navier-Stokes (RANS) equations. These equations form the basis of turbulence modelling in Computational Fluid Dynamics (CFD), which predicts the aerodynamic coefficients based on the airfoil geometry, Reynolds number (Re) and airfoil angle of attack (α). This physics-based approach is typically done on software such as ANSYS and OpenFOAM, which model turbulent flow around the airfoil. Thus, CFD is often utilized to find airfoil shapes that possess the most optimal aerodynamic coefficients. After performing CFD, the airfoil candidates undergo wind tunnel testing, which aims to validate their performance before final airfoil selection and operation. Although CFD methods are accurate, and have reduced the amount of wind tunnel testing, they can be time consuming when used for airfoil optimization [2]–[4]. Therefore, there is an existing need for alternative approaches, to perform fast and accurate optimization of airfoils.

Recently, Artificial Neural Networks (ANNs) are emerging as an alternative approach that circumvents physics-based modelling. ANNs are increasingly popular in aerodynamics, due to the availability of training data, which can be used for creating data-driven surrogate models to perform optimization.

1.1 Literature Review

The utilization of ANNs has only been recently explored in aerodynamics. Several studies have concluded that ANNs possess the potential to augment, or surpass conventional physics-based modelling [5]–[10]. This review investigates the usage of ANNs in substituting the RANS equations, and performing airfoil optimization.

1.1.1 Turbulence modelling and aerodynamic prediction

Researchers in the field of turbulence modelling have extensively explored ANNs as surrogate models, to bypass the RANS equations [2]–[4], [6], [8], [11]–[14]. However, the success of ANNs in modelling turbulent flow varies. Some ANNs achieved poor predictive ability [11], [12], while other ANNs behaved as surrogate models which met, or surpassed conventional RANS models, in prediction accuracy [4], [13]. Thuerey et al. concluded that accuracy can be improved, by investigating different network architectures, such as increased number of layers or neurons [12]. It is also widely known in the literature of the field of machine learning, that hyperparameter optimization is a key component in producing a high performing ANN [15]–[19].

1.1.2 Airfoil optimization

There have been attempts to use ANNs for aerodynamic prediction, design, and optimization [7], [20]–[26]. A recent study shows that ANNs were able to perform fast, and highly accurate, aerodynamic coefficient predictions of airfoils [20]. ANNs were also able to optimize airfoils many times faster than conventional approaches [21], [22], with the potential to shorten design pro-

cesses and reduce costs [22]. Some studies utilized ANNs in Unmanned Aerial Vehicle (UAV) aerodynamic design [27], [28], concluding that ANNs decreased computation time in the design process. In particular, Mazhar et al. [28] indicated that, using ANNs achieved a greater degree of accuracy in structural design, compared to conventional methods. However, ANNs were mostly used in the mentioned literature only for singlepoint optimization, and rarely utilised for multipoint optimization. Thus, the usage of ANNs as a surrogate model to perform airfoil optimization, has not been fully investigated.

1.2 Objective and scope of study

Therefore, the aim of this study is to assess the capability of ANNs as a surrogate model in performing fast, and accurate airfoil optimization. The ANN is primarily used to predict aerodynamic coefficients, bypassing the RANS equations. Firstly, the ANN will be trained using high-fidelity CFD data, which is validated before usage. Next, in order to maximise the prediction accuracy of the ANN, hyperparameter optimization is performed. Subsequently, the predictions made by the trained ANN, will be compared to actual CFD data, to assess its accuracy. Finally, both singlepoint and multipoint optimization, will be conducted for NACA 4-digit airfoils for a small, but focused Re range of $10^4 - 10^5$. The range of α is from $0° - 14°$. NACA 4-digit airfoils are chosen specifically, because their airfoil geometry is parametrized.

2 Details on Machine Learning

2.1 Multilayer Perceptron

A Multilayer Perceptron (MLP) is a type of feed forward ANN, consisting of an input layer, output layer and several hidden layers. Figure 1 shows a typical schematic of an MLP. Basic concepts about MLPs provided by Nielsen [29] are discussed in the following.

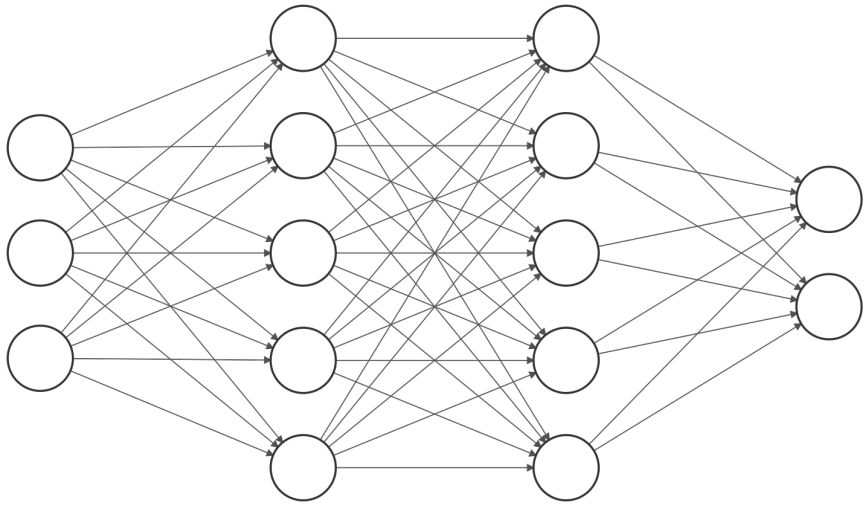


Figure 1. Schematic of a MLP

In a MLP, the hidden layer l consists of j neurons and each neuron has a numeric value a_j^l. This value is mathematically related to numeric value a_k^{l-1} of the kth neuron in the previous hidden layer $n-1$ through some activation function σ. This is represented in equation 1.

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

w_{jk}^l is the weight of the connection from the kth neuron in the $(l-1)$th layer

to the jth neuron in the lth layer. b_j^l is bias of the jth neuron in hidden layer l. The weights express the importance of an input neuron's value to a particular output neuron, while the bias is a measure of how easy a particular neuron outputs a value of 1. The values of the weights and biases determine the final output of a MLP, and the cost function c is used to quantify the deviation of the MLP's output, against the actual values in the dataset. The cost function c, is sometimes known as the quadratic cost function, or Mean Squared Error (MSE). In order to determine the values of these weights and biases, the cost function c is minimized, through an algorithm called gradient descent during training, which adjusts the MLP's weights and biases. The adjustments are computed by taking the gradient of c, in a process known as backpropagation. As the MLP is trained by seeing more data, c is eventually minimized and reaches convergence. The MLP's ability to adjust the weights and biases, in order to minimize c until convergence, is dependent on hyperparameters. Such hyperparameters include: the number of hidden layers, number of neurons per hidden layer, optimizer, learning rate, activation function and batch size.

3 Methodology

There are namely 3 steps conducted in this study. A high fidelity database is first developed, containing NACA 4-digit airfoils for a range of α and Re under turbulent, incompressible flow conditions. The database is validated with existing literatures, to ensure its accuracy. Secondly, a MLP is trained using this database and undergoes hyperparameter optimization. The final MLP's performance will then be assessed, based on actual CFD results. Lastly, the MLP is utilized to perform airfoil optimization for both singlepoint and multipoint optimization. Figure 2 shows a flow chart of this process.

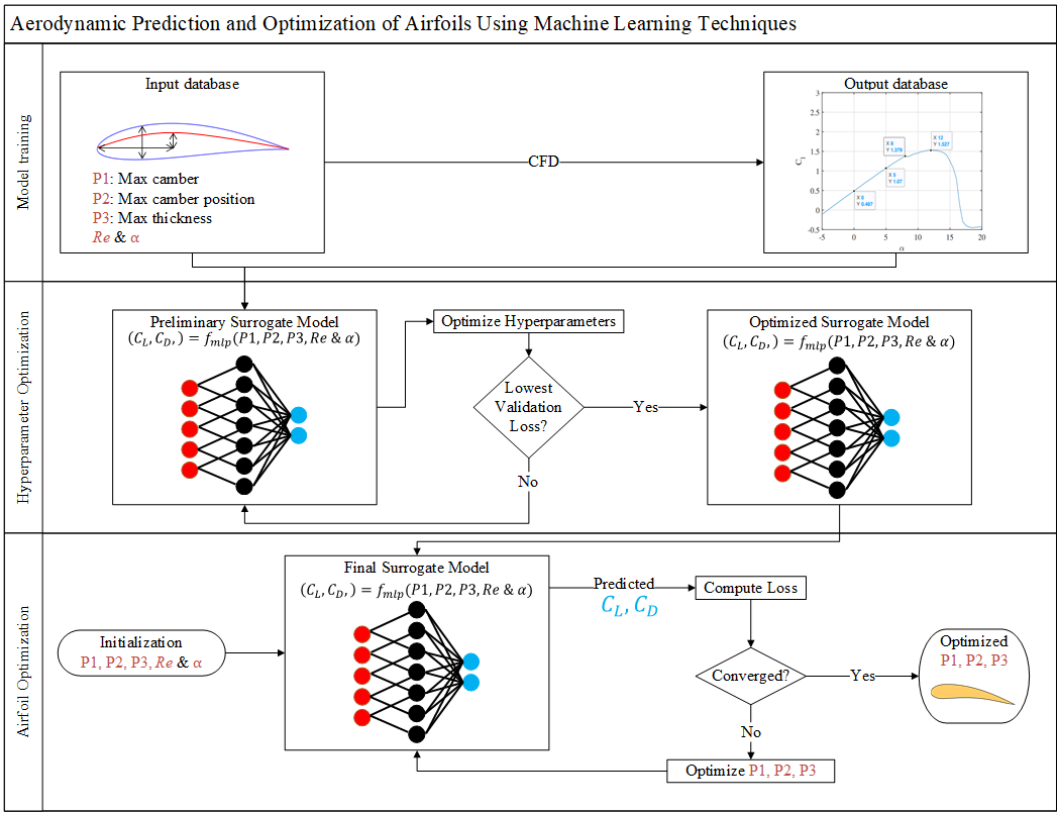


Figure 2. Methodology flowchart

3.1 Database of CFD results

A total of 481 unique NACA 4-digit airfoils were used in this study. The geometry of these airfoils are extracted from UIUC Airfoil Coordinates Database in Cartesian coordinates, and can be expressed by 3 parameters $P1$, $P2$, $P3$ shown in Table 1.

Table 1. Airfoil geometry parameters of NACA 4-digit series

Parameter	Representation	Range	Example: NACA4412
$P1$	Maximum Camber	0 - 6	4
$P2$	Position of Maximum Camber w.r.t chord	0 - 6	4
$P3$	Maximum Thickness	6 - 30	12

The CFD simulations were executed in OpenFOAM, to predict C_L and C_D for

all 481 airfoils, at varying α from $0° - 14°$, for a Re range of $10^4 - 10^5$. Since the critical Reynolds number (Re_{crit}) is difficult to determine for different airfoils, a flow with $Re > 10^4$ was assumed to be turbulent according to literature [30]. Thus, all cases were run using the Spalart-Allmaras Reynolds-averaged Navier-Stokes turbulent model in OpenFOAM, on the National Supercomputing Centre's server automated by Python. C_L and C_D were compiled into the database, together with flow parameters Re and α. In total, there are 19797 unique flow simulation data. This database has already been validated by Jiang Tao [31] with existing literatures. Based on this information, an inference can be made that the 19797 data sets are accurate, and of high-fidelity.

3.2 Preliminary MLP

A preliminary MLP is first built to predict C_L and C_D, using the CFD database. The preliminary MLP consists of 1 input layer, 6 hidden layers and 1 output layer as shown in Figure 3

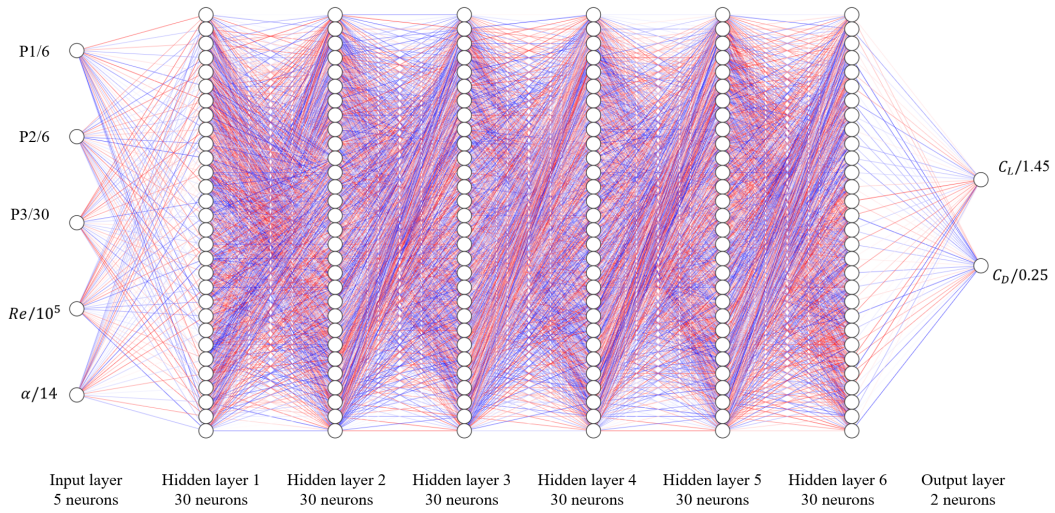


Figure 3. Preliminary MLP

The input layer has 5 neurons and uses airfoil geometry parameters $P1$, $P2$

and $P3$ as input, as well as flow paramaters Re and α. All 5 input parameters are normalized to a value between 0 - 1 before insertion to the MLP. Equation 2 is used for normalization:

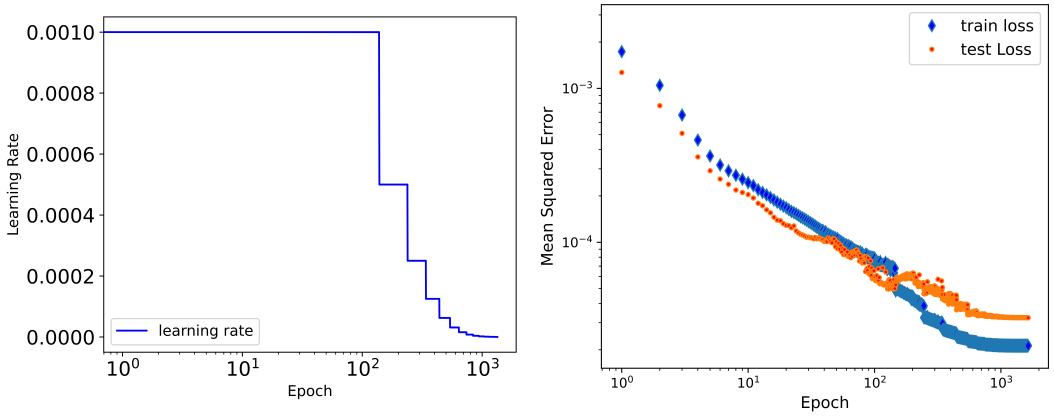
$$P_{nom} = \left(\frac{P - P_{min}}{P_{max} - P_{min}} \right) \tag{2}$$

where P is the actual input, P_{max} and P_{min} are the maximum and minimum values of P respectively. The training settings for the preliminary MLP are given in Table 2.

Table 2. Preliminary MLP training settings

Training Settings	Activation	Optimizer	Learning rate	Learning rate decay	Validation Split	Batch Size
MLP settings	Relu	Adam	2.5×10^{-4}	0.5 for every 100 epochs	0.1	32

The preliminary model converged in 1642 epochs with a final validation MSE of 3.23×10^{-5}. Convergence history is shown in Figures 4a and 4b.



(a) Learning rate decay vs epoch

(b) MSE vs epoch

Figure 4. Preliminary MLP convergence history

3.3 Hyperparameter optimization

A MLP's performance is largely governed by its architecture, consisting of the number of hidden layers, number of neurons per layer, choice of optimizer, choice of learning rate and choice of batch size. There are several methods of hyperparameter optimization. One method is grid searching, where values of the hyperparameters are chosen intentionally, in a systematic manner. Multiple MLPs would be trained using these selected values, and the validation MSE would be compared between the different MLPs, to determine the best hyperparameter values. A thorough grid search has the highest probability of finding the best neural network architecture, however it is time consuming to perform a detailed grid search. To expedite this process, this study has limited the number of neurons and layers to a range of $10-100$ and $1-10$ respectively. Also, each hidden layer will possess the same amount of neurons, and a step size of 10 is chosen, in order to simplify the grid search.

In TensorFlow, ANNs are not reproducible, unless the random number generators are seeded with the same value before training. In order to ensure the hyperparameter models are reproducible to some degree, a fixed seed value of 16154×10^5 is chosen. The hyperparameter optimization will then be performed with 1000 epochs, as that is the number required to guarantee convergence, observed in the convergence history of the preliminary MLP.

3.3.1 Number of neurons and layers

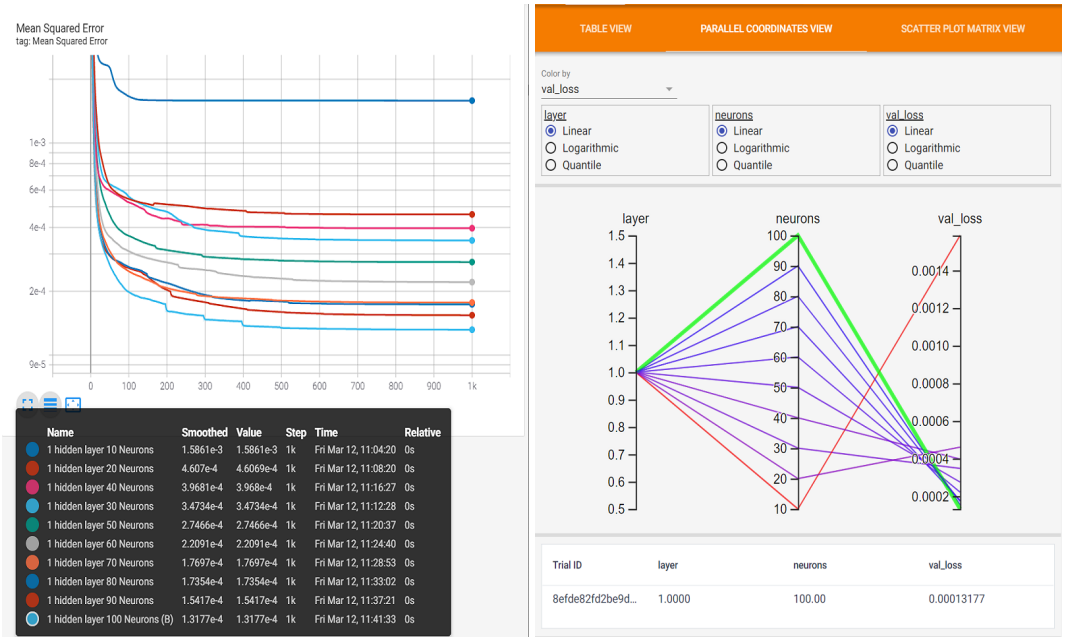
The first component of hyperparameter optimization, determines the best number of hidden layers and neurons per layer for the MLP.

Table 3. Grid search of neurons and hidden layers [1]

1 hidden layer and no. of neurons		2 hidden layers and no. of neurons		3 hidden layers and no. of neurons		4 hidden layers and no. of neurons		5 hidden layers and no. of neurons	
10	$5 \times 10 \times 2$	**10**	$5 \times 10 \times 10 \times 2$	**10**	$5 \times 10^3 \times 2$	**10**	$5 \times 10^4 \times 2$	**10**	$5 \times 10^5 \times 2$
20	$5 \times 20 \times 2$	**20**	$5 \times 20 \times 20 \times 2$	**20**	$5 \times 20^3 \times 2$	**20**	$5 \times 20^4 \times 2$	**20**	$5 \times 20^5 \times 2$
30	$5 \times 30 \times 2$	**30**	$5 \times 30 \times 30 \times 2$	**30**	$5 \times 30^3 \times 2$	**30**	$5 \times 30^4 \times 2$	**30**	$5 \times 30^5 \times 2$
40	$5 \times 40 \times 2$	**40**	$5 \times 40 \times 40 \times 2$	**40**	$5 \times 40^3 \times 2$	**40**	$5 \times 40^4 \times 2$	**40**	$5 \times 40^5 \times 2$
50	$5 \times 50 \times 2$	**50**	$5 \times 50 \times 50 \times 2$	**50**	$5 \times 50^3 \times 2$	**50**	$5 \times 50^4 \times 2$	**50**	$5 \times 50^5 \times 2$
60	$5 \times 60 \times 2$	**60**	$5 \times 60 \times 60 \times 2$	**60**	$5 \times 60^3 \times 2$	**60**	$5 \times 60^4 \times 2$	**60**	$5 \times 60^5 \times 2$
70	$5 \times 70 \times 2$	**70**	$5 \times 70 \times 70 \times 2$	**70**	$5 \times 70^3 \times 2$	**70**	$5 \times 70^4 \times 2$	**70**	$5 \times 70^5 \times 2$
80	$5 \times 80 \times 2$	**80**	$5 \times 80 \times 80 \times 2$	**80**	$5 \times 80^3 \times 2$	**80**	$5 \times 80^4 \times 2$	**80**	$5 \times 80^5 \times 2$
90	$5 \times 90 \times 2$	**90**	$5 \times 90 \times 90 \times 2$	**90**	$5 \times 90^3 \times 2$	**90**	$5 \times 90^4 \times 2$	**90**	$5 \times 90^5 \times 2$
100	$5 \times 100 \times 2$	**100**	$5 \times 100 \times 100 \times 2$	**100**	$5 \times 100^3 \times 2$	**100**	$5 \times 100^4 \times 2$	**100**	$5 \times 100^5 \times 2$
6 hidden layers and no. of neurons		**7 hidden layers and no. of neurons**		**8 hidden layers and no. of neurons**		**9 hidden layers and no. of neurons**		**10 hidden layers and no. of neurons**	
10	$5 \times 10^6 \times 2$	**10**	$5 \times 10^7 \times 2$	**10**	$5 \times 10^8 \times 2$	**10**	$5 \times 10^9 \times 2$	**10**	$5 \times 10^{10} \times 2$
20	$5 \times 20^6 \times 2$	**20**	$5 \times 20^7 \times 2$	**20**	$5 \times 20^8 \times 2$	**20**	$5 \times 20^9 \times 2$	**20**	$5 \times 20^{10} \times 2$
30	$5 \times 30^6 \times 2$	**30**	$5 \times 30^7 \times 2$	**30**	$5 \times 30^8 \times 2$	**30**	$5 \times 30^9 \times 2$	**30**	$5 \times 30^{10} \times 2$
40	$5 \times 40^6 \times 2$	**40**	$5 \times 40^7 \times 2$	**40**	$5 \times 40^8 \times 2$	**40**	$5 \times 40^9 \times 2$	**40**	$5 \times 40^{10} \times 2$
50	$5 \times 50^6 \times 2$	**50**	$5 \times 50^7 \times 2$	**50**	$5 \times 50^8 \times 2$	**50**	$5 \times 50^9 \times 2$	**50**	$5 \times 50^{10} \times 2$
60	$5 \times 60^6 \times 2$	**60**	$5 \times 60^7 \times 2$	**60**	$5 \times 60^8 \times 2$	**60**	$5 \times 60^9 \times 2$	**60**	$5 \times 60^{10} \times 2$
70	$5 \times 70^6 \times 2$	**70**	$5 \times 70^7 \times 2$	**70**	$5 \times 70^8 \times 2$	**70**	$5 \times 70^9 \times 2$	**70**	$5 \times 70^{10} \times 2$
80	$5 \times 80^6 \times 2$	**80**	$5 \times 80^7 \times 2$	**80**	$5 \times 80^8 \times 2$	**80**	$5 \times 80^9 \times 2$	**80**	$5 \times 80^{10} \times 2$
90	$5 \times 90^6 \times 2$	**90**	$5 \times 90^7 \times 2$	**90**	$5 \times 90^8 \times 2$	**90**	$5 \times 90^9 \times 2$	**90**	$5 \times 90^{10} \times 2$
100	$5 \times 100^6 \times 2$	**100**	$5 \times 100^7 \times 2$	**100**	$5 \times 100^8 \times 2$	**100**	$5 \times 100^9 \times 2$	**100**	$5 \times 100^{10} \times 2$

[1] $5 \times X^n \times 2$ represents a MLP with an input layer of 5 neurons, n hidden layers of X neurons each and a output layer of 2 neurons.

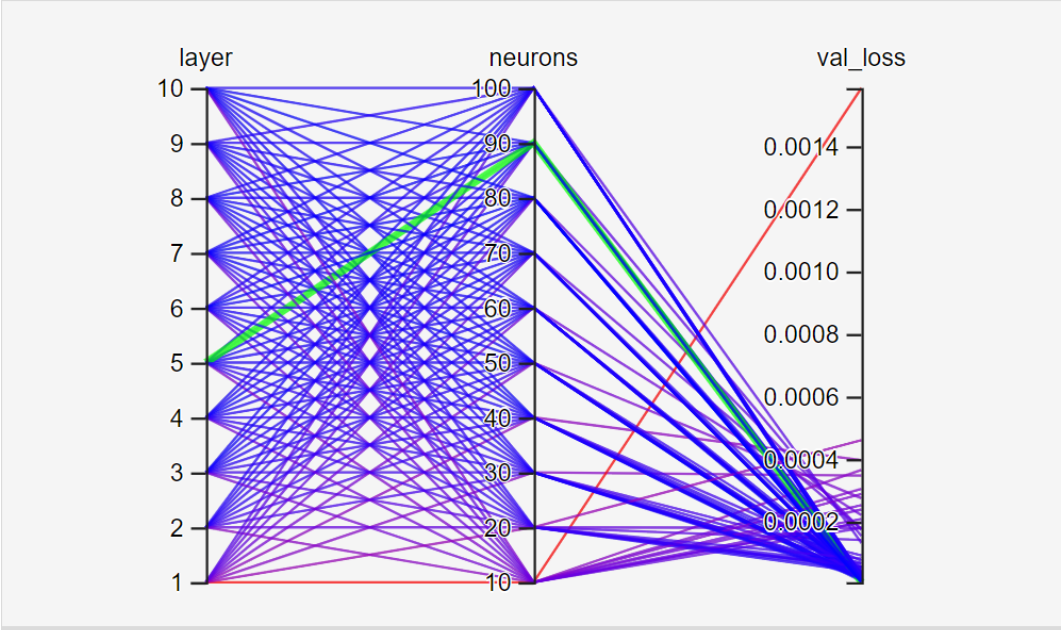
A total of 100 MLPs are trained for 1000 epochs, each possessing a hidden layer ranging from $1 - 10$ and the number of neurons per layer from $10 - 100$ as shown in Table 3. The training settings are identical to the ones used for the preliminary MLP. For each iteration of hidden layer, validation MSE per epoch of the MLPs were plotted on TensorBoard, to visualise and compare MLP performance.



(a) Validation MSE vs epoch (b) Final validation MSE

Figure 5. Hyperparameter study of various neurons in 1 hidden layer

The TensorBoard scalar view shows the validation MSE vs epoch, for each iteration of MLP in the grid search. The graphs indicate the presence of overfitting, and general behaviour of the validation MSE, as the network trains with each epoch. It is common to observe that the best performing MLP for an early number of epochs, does not result in the best performing MLP at the end of training. Using TensorBoard's hyperparameter parallel coordinate view, the best MLP architecture in each iteration of layers can be determined. For a MLP with 1 hidden layer, 100 neurons gives the best performance, with a validation MSE of 1.32×10^{-4} as shown in Figure 5b. The results for the other hidden layers from $2-10$, can be found in the Appendix. Once the grid search is complete, the 100 MLPs are compared simultaneously as shown in Figure 6.



Trial ID	layer	neurons	val_loss
a06a8f16afb00...	5.0000	90.000	0.0000062760

Figure 6. Validation MSE for all 100 MLPs

In general, an increase in the number of neurons correlate with a lower validation MSE as shown in Figure 6. Morever, an increase in the number of hidden layers also correlates with a lower validation MSE, however this trend did not continue for MLPs with more than 3 hidden layers. By selecting the best performing MLP for each iteration of hidden layers, it can be seen that the optimum number of neurons per layer for each iteration is different, as Table 4 shows.

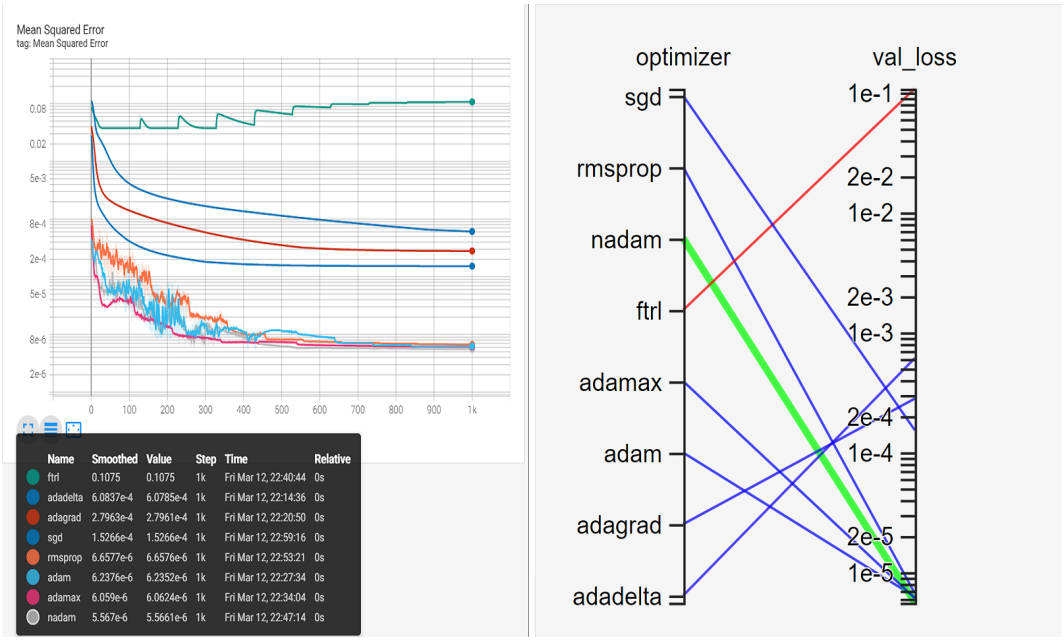
Table 4. Best performing MLP in every hidden layer iteration

Number of hidden layers	Neurons per layer	Validation MSE	Epochs trained
1	100	1.32×10^{-4}	1000
2	100	2.78×10^{-5}	1000
3	90	9.72×10^{-6}	1000
4	80	1.11×10^{-5}	1000
5	90	6.27×10^{-6}	1000
6	80	6.58×10^{-6}	1000
7	80	8.98×10^{-6}	1000
8	60	8.74×10^{-6}	1000
9	40	1.40×10^{-5}	1000
10	80	6.83×10^{-6}	1000

The best performing MLP in Table 4 consisted of 5 hidden layers, with 90 neurons per hidden layer. The best performing MLP is desirable for this study, as it would ensure the highest accuracy in predictions, essential for airfoil optimization. One reason why MLPs with 6 hidden layers or more did not perform better, may be because of the difficulty experienced by larger networks in generalising training data. Larger networks tend to memorize data, and are not as nimble in changing weights and biases during training, compared to smaller networks. However, a network too small may be unable to properly generalize data, as it lacks complexity.

3.3.2 Optimizer

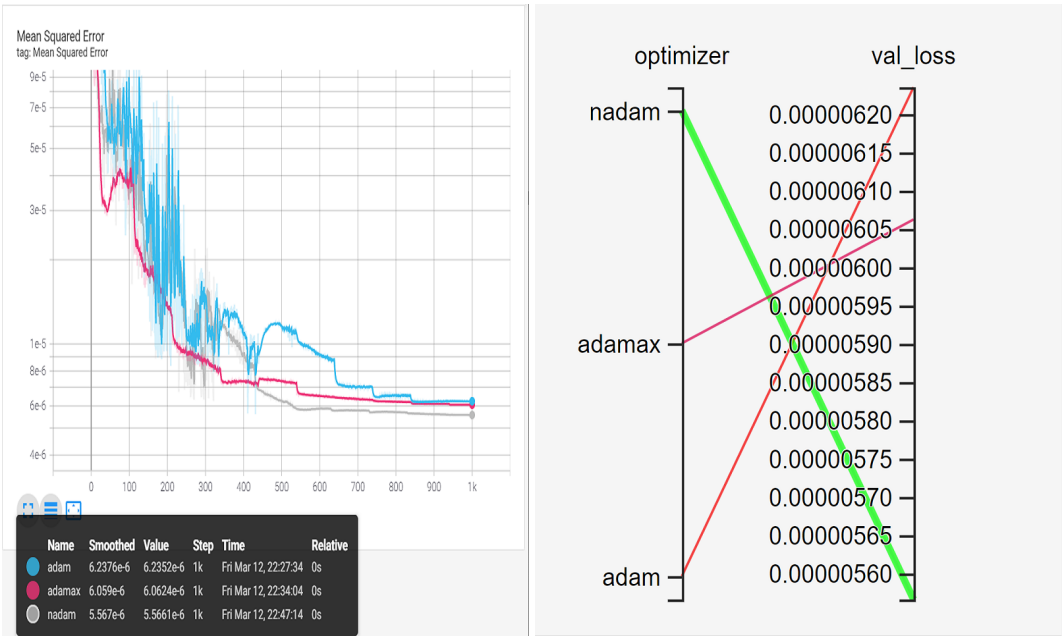
In TensorFlow, there are 6 different optimizers, each with unique algorithms, and some specifically perform better at certain tasks. Given that the best MLP architecture in the previous section was 5 hidden layers with 90 neurons per hidden layer, 6 MLPs were trained across 6 different optimizers with this architecture.



(a) Validation MSE vs epoch (b) Final validation MSE

Figure 7. Hyperparameter study of different optimizers

Figure 7 indicates that optimizers Ftrl, Adadelta, Adagrad and SGD were inferior to optimizers RMSprop, Adam, Adamax and Nadam.



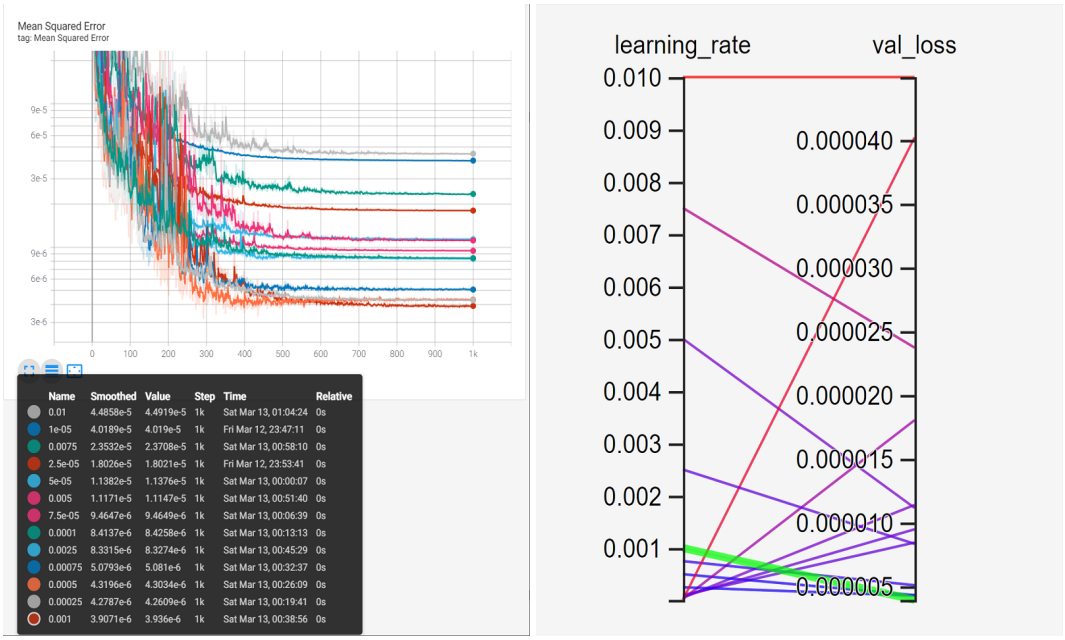
(a) Validation MSE vs epoch (b) Final validation MSE

Figure 8. Comparison of optimizers Adam, Adamax and Nadam

Closer inspection in the scalars view shown by Figure 8a shows that, the best performing optimizer was the Nadam optimizer, producing a final validation MSE of 5.57×10^{-6}. Furthermore, the graphs show that Adamax and Nadam were more stable than Adam, as they experienced less fluctuation. Ultimately, Nadam is the preferred optimizer of choice for the final MLP.

3.3.3 Learning rate

The learning rate of an optimizer, determines how well a MLP converges to an optimum set of weights and biases. An excessively high learning rate may miss this optimum set of weights and biases, while a MLP may never converge at all with an excessively low learning rate.



(a) Validation MSE vs epoch (b) Final validation MSE

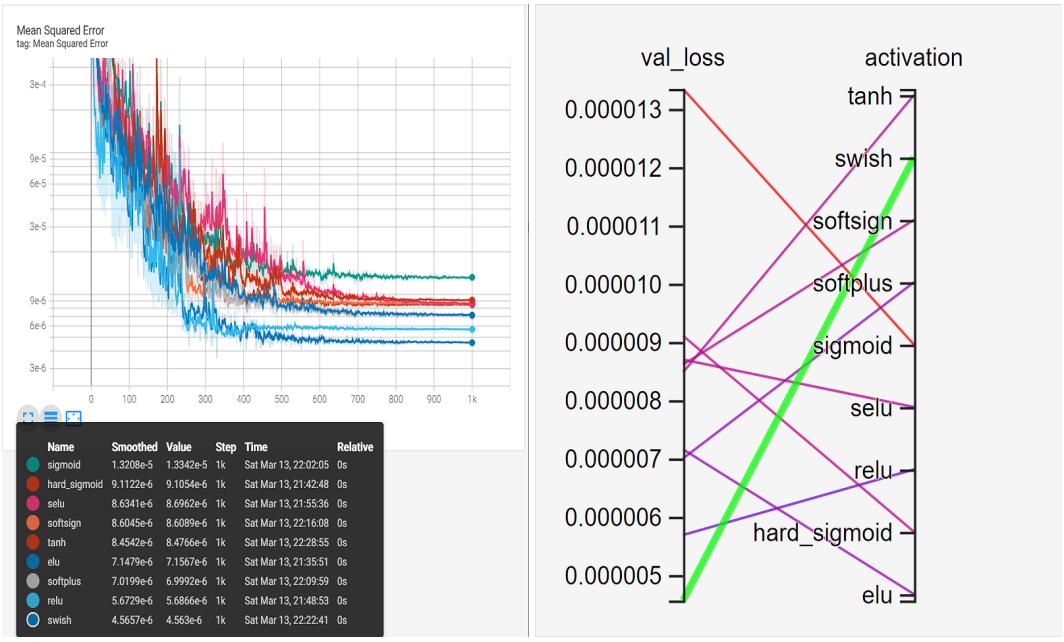
Figure 9. Hyperparameter study of different learning rates for Nadam

13 different learning rates were studied on the $5 \times 90^5 \times 2$ MLP with Nadam optimizer, ranging from 1×10^{-5} to 0.01. As shown in Figure 9b, the learning rate of 0.001 is the most optimal to train the MLP using the training dataset,

further reducing the validation MSE to 3.936×10^{-6}. As a result, a choice of 0.001 is chosen as the learning rate of the final MLP.

3.3.4 Activation function

TensorFlow provides numerous activation functions, which allow MLPs to have non-linear characteristics. Thus, the choice of activation function can impact a MLP's performance. A total of 9 activation functions were studied on the $5 \times 90^5 \times 2$ MLP model, with Nadam optimizer of learning rate 0.001 shown in Figure 10.



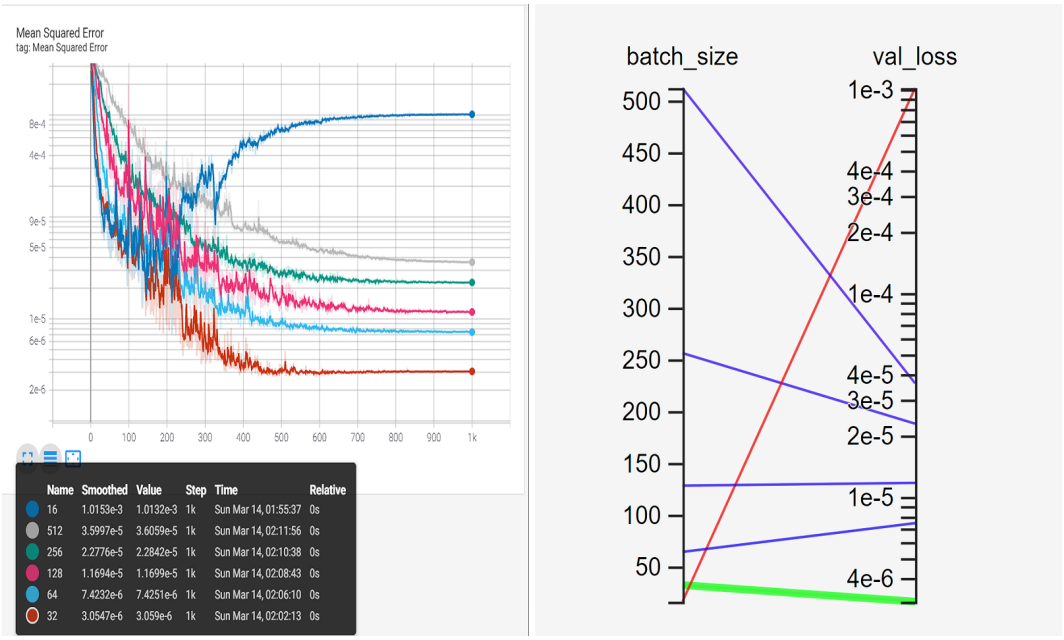
(a) Validation MSE vs epoch (b) Final validation MSE

Figure 10. Hyperparameter study of different activation functions

In accordance with Figure 10a, the Swish activation function, is the best performing activation function. Therefore, Swish is selected as the activation function of choice for the final MLP.

3.3.5 Batch size

The impact of batch size on ANN performance of the ANN is also investigated.



(a) Validation MSE vs epoch (b) Final validation MSE

Figure 11. Hyperparameter study of different batch sizes

The graphs shown in Figure 11 indicate that, a batch size of 32 provides the lowest validation MSE for the $5 \times 90^5 \times 2$ MLP, with Nadam optimizer of learning rate 0.001 and Swish activation function. In conclusion, the final MLP will consist of 5 hidden layers of 90 neurons each, Nadam optimizer of learning rate 0.001, Swish activation function and a batch size of 32.

3.4 Airfoil optimization

Airfoil optimization is performed on Python using the SciPy library. Airfoils are optimized for different objectives: maximize C_L, minimize C_D, maximize $\frac{C_L}{C_D}$ and minimize C_D for a fixed $C_{L_{fixed}}$. The solution space given by the MLP depends on five variables, with α and Re being the two fixed variables given by

the optimization problem. The optimization algorithm therefore aims to find the wing parameters $P1$, $P2$ and $P3$, which generates the best airfoil shape that meets the requirements. In order to prevent the solutions from being trapped into local optima, a multi-start approach is taken as well as utilising different optimizers. If all the starting airfoils and optimizers converge to a particular solution, it is highly likely that solution is the global optimum.

3.4.1 Details on SciPy

SciPy is an open-source Python library, used for scientific computing and technical computing. It contains modules for optimization, which provides functions for minimizing or maximizing objective functions, sometimes subjected to contraints. The functions in SciPy are able solve nonlinear problems, with support for local and global optimization algorithms. Table 5 shows the types of optimization algorithms and their characteristics.

Table 5. List of SciPy optimizers

Optimizer Name	**Acronym**	**Type**	**Method**
Limited memory Broyden –Fletcher–Goldfarb–Shanno	L-BFGS-B	Local	Gradient based [32], [33]
Sequential Least Squares Programming	SLSQP	Local	Gradient based [34]
Truncated Newton	TNC	Local	Gradient based [35], [36]
Powell	Powell	Local	Non-gradient based [37], [38]
Basin-hopping	basinhopping	Global	Can be both gradient or non-gradient based [39]
Differential Evolution	Differential Evolution	Global	Non-gradient based [40]
Simplicial Homology Global Optimization	SHGO	Global	Non-gradient based [41]
Dual Annealing	Dual Annealing	Global	Non-gradient based [42], [43]

3.4.2 Single point optimization

In some situations, the airfoil may be designed specifically just for one purpose or design point. Examples include racing cars or gliders. The objective function for single optimization provided by Nemec [44] is given in Equation 3:

$$\begin{aligned} &\min_{P1,P2,P3} f_{singlepoint}(P1, P2, P3, Re, \alpha) \\ &\qquad f_{singlepoint}(P1, P2, P3, Re, \alpha) = (target - predicted)^2 \\ &\text{s.t. } (0 < P1 < 6), (0 < P2 < 6), (6 < P3 < 30), (10^4 < Re < 10^5), (0^\circ < \alpha < 14^\circ) \end{aligned} \tag{3}$$

where $target$ can be C_L, C_D, $\frac{C_L}{C_D}$ and $\frac{C_{L\,fixed}}{C_D}$ for a fixed C_L value. Re and α are fixed values set by the design point.

For conducting single point optimization, Ukken's paper [45] will be used as a reference, where single point optimizations were performed for C_L, C_D and $\frac{C_L}{C_D}$. The results obtained by the author are listed in the Table below.

Table 6. Single point optimization design points

$f_{singlepoint}$	Starting Airfoil	Re	α (°)	Result obtained by Ukken
maximize C_L	NACA0008	10^4	2	0.28
minimize C_D	NACA0008	10^4	0	0.0319
maximize $\frac{C_L}{C_D}$	NACA0008	10^4	2	10

The same single point optimizations will be performed in this study using the conditions given in Table 6.

3.4.3 Multipoint optimization

Most aircraft however, are designed to operate at multiple design points, such as over multiple flight profiles. It is thus essential to optimize the airfoil across all design points, such that the chosen airfoil would ensure maximum performance. According to Nemec [44], multipoint optimization requires modifying the objective function, into a weighted sum as given by Equation (4). The weights associated to the design points, indicate how important that design point is, with regard to the total flight profile.

$$\min_{P1,P2,P3} f_{multipoint}(P1, P2, P3, Re_i, \alpha_i)$$

$$f_{multipoint}(P1, P2, P3, Re_i, \alpha_i) = \sum_{i=1}^{n} w_i (target_i - predicted_i)^2$$

$$\text{s.t. } (0 < P1 < 6), (0 < P2 < 6), (6 < P3 < 30), (10^4 < Re_i < 10^5), (0^\circ < \alpha_i < 14^\circ) \tag{4}$$

where $target$ can be C_L, C_D, $\frac{C_L}{C_D}$ and $\frac{C_{L\,fixed}}{C_D}$ for a fixed C_L value. w_i, Re_i and α_i are fixed values of design point i for n number of design points.

To demonstrate multipoint optimization, a UAV flight profile is composed with 3 stages, namely takeoff, cruise and landing along with their corresponding α using Ajaj et al. as a reference [46]. The top speed of the UAV is taken with reference to McEvoy et al. [47] at a value of $11.11 ms^{-1}$. The chord length of the airfoil chosen is $0.12m$, giving the Re values shown in Table 7 below for a low altitude flight path. The multipoint objective is to maintain a fixed $C_{L\,fixed}$ value, while reducing C_D as much as possible, for each of the design points. A reasonable $C_{L\,fixed}$ target value is chosen based on the results for NACA4412 airfoil. Table 7 shows the details of the multipoint optimization.

Table 7. Multipoint optimization design points

Design Point	Airspeed (ms^{-1})	Reynolds number	α (°)	NACA4412 C_D	NACA4412 C_L	Design Weight w_i	C_{Lfixed}	$f_{multipoint}$
Takeoff	5.555	46921	10.9	0.065	1.227	$\frac{1}{6}$	1.3	$\frac{C_{Lfixed}}{C_D}$
Cruise	11.111	93850	5	0.036	0.847	$\frac{2}{3}$	1	$\frac{C_{Lfixed}}{C_D}$
Landing	4.444	37536	13.9	0.113	1.258	$\frac{1}{6}$	1.3	$\frac{C_{Lfixed}}{C_D}$

4 Results and Discussion

4.1 Final MLP performance

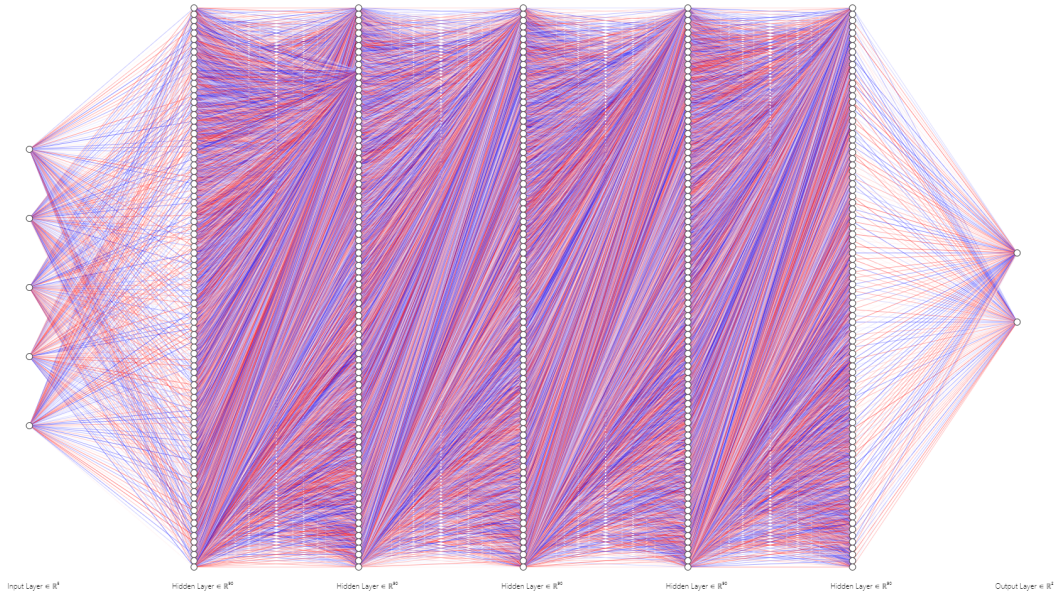
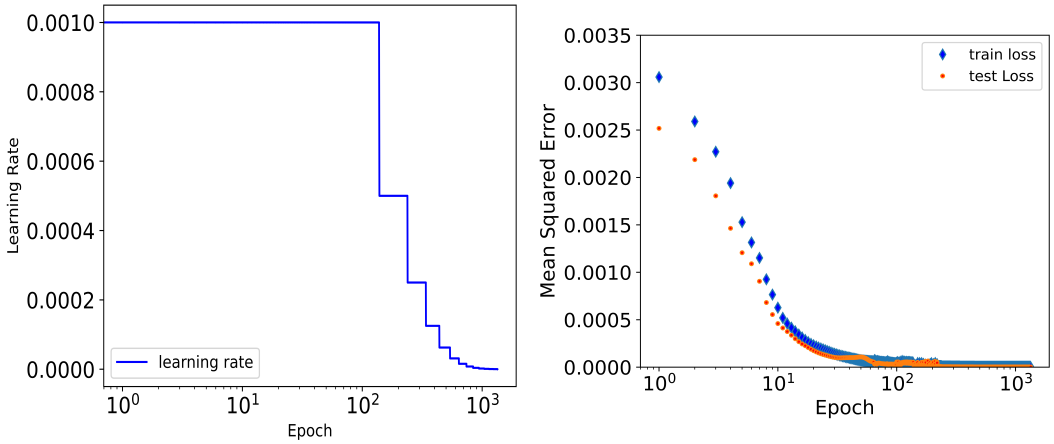


Figure 12. Final MLP architecture

The final MLP model consisting of 5 hidden layers each with 90 neurons, was trained and converged in 1340 epochs with a validation MSE of 1.86×10^{-6}. Convergence history is given by Figure 13.

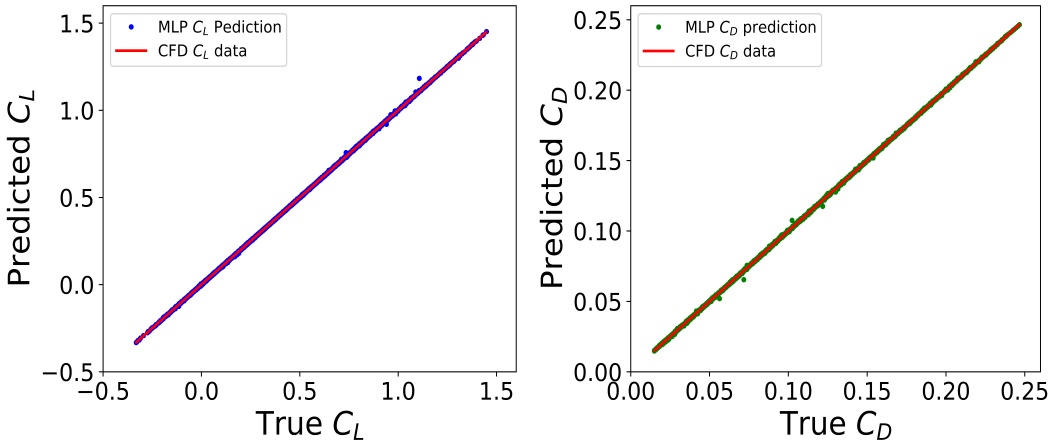


(a) Learning rate decay vs epoch

(b) MSE vs epoch

Figure 13. Final MLP convergence history

Using the entire database as input, the predictions made by the model are plotted against the actual CFD database values, for C_L and C_D as shown in figures 14a and 14b respectively.

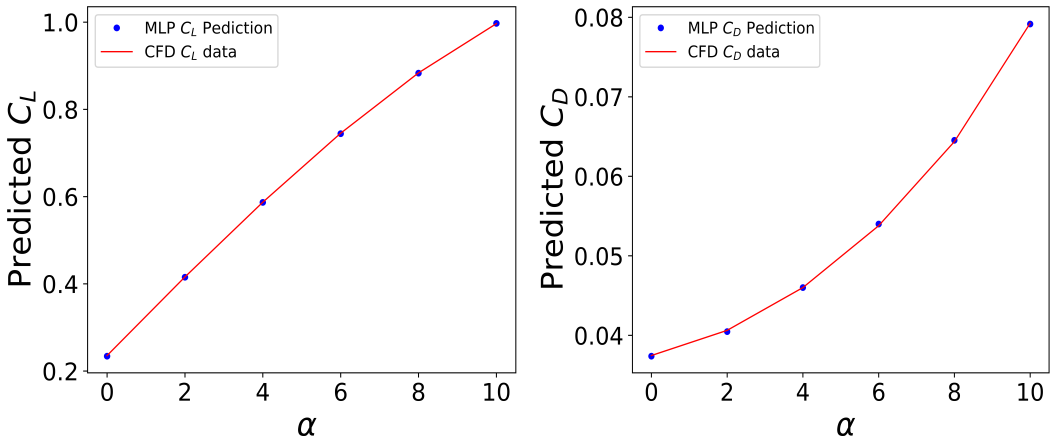


(a) C_L predictions vs CFD data

(b) C_D predictions vs CFD data

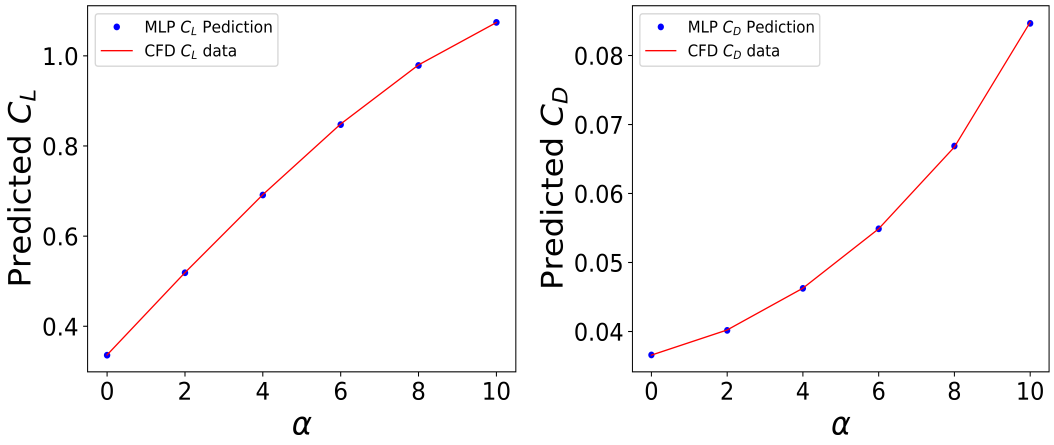
Figure 14. MLP predictions vs CFD data

The final MLP is able to accurately predict aerodynamic coefficients, with only slight deviations from the actual values. Furthermore, the final MLP was able to predict two airfoils: NACA4514 and NACA5412, which are non-existent in the database, very accurately, shown by Figure 15 and 16.



(a) C_L vs α (b) C_D vs α

Figure 15. MLP predictions vs CFD data for NACA4514 at $Re = 2 \times 10^4$



(a) C_L vs α (b) C_D vs α

Figure 16. MLP predictions vs CFD data for NACA5412 at $Re = 2 \times 10^4$

Therefore, it is evident that the final MLP model is extremely effective as a surrogate model, possessing high prediction accuracy. However, there are a few outliers as seen in Figure 14. These reason for these outliers is inconclusive. The MLP may be unable to fit all of these points with the final weights and biases, as these may be edge cases caused by abnormal meshes or diverged CFD simulations. Assumptions of the Re_{crit} of the airfoils, may have also played a role.

4.2 Single point optimization

The overall results for single point optimization are given in Table 8

Table 8. Summarized results for single point optimization

Target	Reynolds number	α (°)	Result obtained by Ukken [45]	**NACA0018 (base)**	**MLP optimized airfoil**
C_L	10^4	2	0.28	0.18	0.57
C_D	10^4	0	0.0319	0.036	0.034
$\frac{C_L}{C_D}$	10^4	2	10.2	4.67	11.9

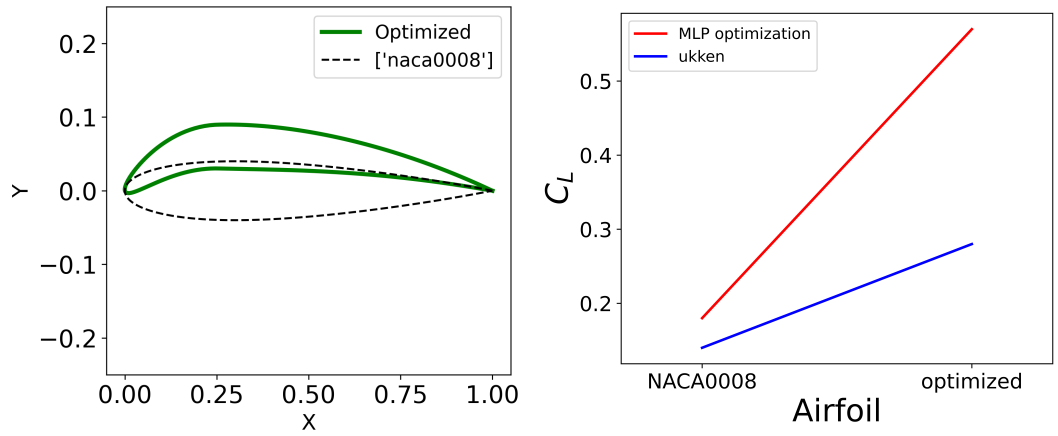
4.2.1 Optimize for maximum lift coefficient

Performing the optimization using the MLP, the results of $P1$, $P2$ and $P3$ for different starting airfoils and optimizers are shown in Table 9.

Table 9. Optimized parameters by optimizers and starting airfoils for C_L

Starting Airfoil	**Optimzer**	L-BFGS-B	SLSQP	TNC	Powell	basinhopping (L-BFGS-B)	Differential Evolution	SHGO	Dual Annealing
NACA0008	$P1$	6	0	0	6	6	6	6	6
	$P2$	2.54	0	0	1.05	2.48	2.57	2.25	2.55
	$P3$	6	8	8	6	6	6	6	6
NACA0012	$P1$	6	0	0	6	6	6.03	6	6
	$P2$	2.52	0	0	1.49	2.25	2.67	2.25	2.55
	$P3$	6	12	12	6	6	6.07	6	6
NACA1408	$P1$	6	1	1	6	6	6	6	6
	$P2$	2.55	4	4	2.55	2.60	2.99	2.25	2.55
	$P3$	6	8	8	6	6	6.1	6	6
NACA2408	$P1$	6	2	2	6	6	6	6	6
	$P2$	2.55	4	4	2.55	2.78	2.67	2.25	2.55
	$P3$	6	8	8	6	6	6.07	6	6
NACA2412	$P1$	6	2	2	6	6	6	6	6
	$P2$	2.55	4	4	2.55	2.57	3.06	2.25	2.55
	$P3$	6	12	12	6	6	6	6	6
NACA2424	$P1$	6	2	2	6	6	6	6	6
	$P2$	2.55	4	4	2.55	2.83	2.66	2.25	2.55
	$P3$	6	24	24	6	6	6.14	6	6
NACA4412	$P1$	6	4	4	6	6	6	6	6
	$P2$	2.55	4	4	2.55	3.29	2.9	2.25	2.55
	$P3$	6	12	12	6	6	6.03	6	6
NACA4424	$P1$	6	4	4	6	6	6	6	6
	$P2$	2.55	4	4	2.55	2.3	2.59	2.25	2.55
	$P3$	6	24	24	6	6	6.05	6	6
NACA6409	$P1$	6	6	6	6	6	5.99	6	6
	$P2$	2.55	4	4	2.55	2.07	2.38	2.25	2.55
	$P3$	6	9	9	6	6	6.17	6	6
NACA6412	$P1$	6	6	6	6	6	5.99	6	6
	$P2$	2.55	4	4	2.55	2.62	2.91	2.25	2.55
	$P3$	6	12	12	6	6	6.02	6	6
Convergence		Yes	No	No	No	No	No	Yes	Yes
Mean final error		1.2301	1.5649	1.5649	1.2344	1.2306	1.2317	1.2308	1.2301

Only optimizers L-BFGS-B, SHGO and Dual Annealing, were able to converge to a solution for all starting airfoils. Since the error for Dual Annealing optimizer is the lowest, the airfoil generated from Dual Annealing optimizer is chosen as the optimized airfoil, shown by Figure 17. However, it is noted that SHGO performs optimization much faster than Dual Annealing, with very little performance loss.



(a) Optimized airfoil geometry

(b) Comparison of results

Figure 17. Single point optimization for C_L at $Re = 10^4$ and $\alpha = 2°$

With reference to figure 17, the optimized airfoil has a greater lift coefficient of 0.57 predicted by the MLP. This is a C_L improvement from NACA0008.

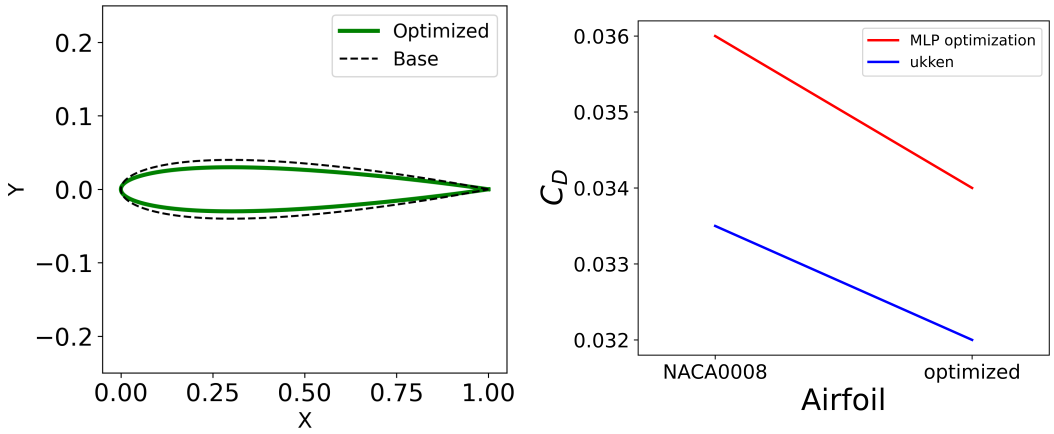
4.2.2 Optimize for minimum drag coefficient

The optimization results for minimum drag are shown in Table 10. Optimizers basinhopping, SHGO and Dual Annealing converged to a solution. However, the results from basinhopping and SHGO possessed a lower mean final error of 2.922×10^{-2}, as compared to the mean final error of 2.924×10^{-2} from Dual Annealing. Thus, the optimized parameters $P1$, $P2$ and $P3$ were 0, 6, 6 respectively.

Table 10. Optimized parameters by optimizers and starting airfoils for C_D

Starting Airfoil	Optimizer	L-BFGS-B	SLSQP	TNC	Powell	basinhopping (L-BFGS-B)	Differential Evolution	SHGO	Dual Annealing
NACA0008	$P1$	0	0	0	0	0	0.02	0	0.02
	$P2$	0	0	0	1.6	6	5.95	6	6
	$P3$	6	8	8	6	6	6	6	6
NACA0012	$P1$	0	0	0	1.1	0	0.2	0	0.02
	$P2$	0	0	0	1.78	6	5.84	6	6
	$P3$	6	12	12	6	6	6	6	6
NACA1408	$P1$	0	1	1	0	0	0.05	0	0.02
	$P2$	6	4	4	1.64	6	1.79	6	6
	$P3$	6	8	8	6	6	6	6	6
NACA2408	$P1$	0	2	2	0	0	0.1	0	0.02
	$P2$	6	4	4	1.64	6	5.96	6	6
	$P3$	6	8	8	6	6	6	6	6
NACA2412	$P1$	0	2	2	0	0	0.07	0	0.02
	$P2$	6	4	4	1.64	6	5.89	6	6
	$P3$	6	12	12	6	6	6.07	6	6
NACA2424	$P1$	0	2	2	0	0	5.96	0	0.02
	$P2$	6	4	4	1.64	6	2	6	6
	$P3$	6	24	24	6	6	6.28	6	6
NACA4412	$P1$	0	4	4	0	0	0.08	0	0.02
	$P2$	6	4	4	1.65	6	5.9	6	6
	$P3$	6	12	12	6	6	6	6	6
NACA4424	$P1$	0	4	4	0	0	0	0	0.02
	$P2$	1.63	4	4	1.63	6	6	6	6
	$P3$	6	24	24	6	6	6	6	6
NACA6409	$P1$	0	6	6	0	0	0.11	0	0.02
	$P2$	1.63	4	4	1.64	6	5.76	6	6
	$P3$	6	9	9	6	6	6	6	6
NACA6412	$P1$	0	6	6	0	0	0.01	0	0.02
	$P2$	1.63	4	4	1.63	6	2.05	6	6
	$P3$	6	12	12	6	6	6	6	6
Convergence		No	No	No	No	Yes	No	Yes	Yes
Mean final error		2.935×10^{-2}	4.497×10^{-2}	4.497×10^{-2}	2.936×10^{-2}	2.922×10^{-2}	2.931×10^{-2}	2.922×10^{-2}	2.924×10^{-2}

As shown in Figure 18, the C_D of the optimized airfoil was 0.034, a reduction from 0.036 with NACA0008 as the starting airfoil. The optimized airfoil also possesses the minimum allowable thickness in this study, with no camber.



(a) Optimized airfoil geometry

(b) Comparison of results

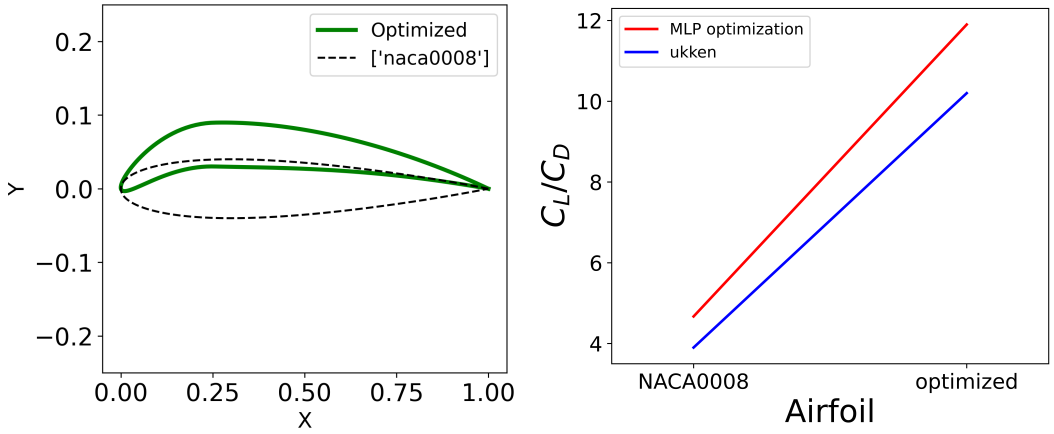
Figure 18. Single point optimization for C_D at $Re = 10^4$ and $\alpha = 0°$

4.2.3 Optimize for maximum lift-drag ratio

Table 11. Optimized parameters by optimizers and starting airfoils for $\frac{C_L}{C_D}$

Starting Airfoil	**Optimzer**	L-BFGS-B	SLSQP	TNC	Powell	basinhopping (L-BFGS-B)	Differential Evolution	SHGO	Dual Annealing
NACA0008	$P1$	6	0	0	6	6	6	6	6
	$P2$	2.54	0	0.375	1.05	3.08	2.66	2.25	2.52
	$P3$	6	8	8	6	6	6	6	6
NACA0012	$P1$	6	0	0	6	6	5.99	6	6
	$P2$	2.52	0	0	1.49	2.88	3.05	2.25	2.43
	$P3$	6	12	12	6	6	6	6	6
NACA1408	$P1$	6	1	1	6	6	5.98	6	6
	$P2$	2.55	4	4	2.55	3.05	2.78	2.25	2.55
	$P3$	6	8	8	6	6	6	6	6
NACA2408	$P1$	6	2	2	6	6	6	6	6
	$P2$	2.55	4	4	2.55	2.69	2.47	2.25	2.55
	$P3$	6	8	8	6	6	6.06	6	6
NACA2412	$P1$	6	2	2	6	6	6	6	6
	$P2$	2.55	4	4	2.55	2.73	2.48	2.25	2.55
	$P3$	6	12	12	6	6	6.03	6	6
NACA2424	$P1$	6	2	2	6	6	6	6	6
	$P2$	2.55	4	4	2.55	1.96	2.55	2.25	2.55
	$P3$	6	24	24	6	6	6.1	6	6
NACA4412	$P1$	6	4	4	6	6	6	6	6
	$P2$	2.55	4	4	2.55	2.05	2.79	2.25	2.56
	$P3$	6	12	12	6	6	6	6	6
NACA4424	$P1$	6	4	4	6	6	6	6	6
	$P2$	2.55	4	4	2.55	2.38	2.78	2.25	2.56
	$P3$	6	24	24	6	6	6.16	6	6
NACA6409	$P1$	6	6	6	6	6	5.99	6	6
	$P2$	2.54	4	4	2.55	2.51	2.30	2.25	2.55
	$P3$	6	9	9	6	6	6.04	6	6
NACA6412	$P1$	6	6	6	6	6	6	6	6
	$P2$	2.55	4	4	2.55	2.03	2.32	2.25	2.54
	$P3$	6	12	12	6	6	6	6	6
Convergence		Yes	No	No	No	No	No	Yes	Yes
Mean final error		1.0301	1.3651	1.3646	1.0344	1.0310	1.0310	1.0308	1.0301

The results for optimizing lift to drag ratio are obtained shown in Table 11. Converged solutions were obtained for optimizers L-BFGS-B, SHGO and Dual Annealing. The optimizers which produced the lowest mean final error, are from L-BFGS-B and Dual Annealing. Using the parameters $P1 = 6$, $P2 = 2.55$, $P3 = 6$ as the optimized airfoil, a $\frac{C_L}{C_D}$ ratio of 11.9 was attained, an increase from the initial starting value of 4.67 shown in Figure 19.



(a) Optimized airfoil geometry

(b) Comparison of results

Figure 19. Single point optimization for $\frac{C_L}{C_D}$ at $Re = 10^4$ and $\alpha = 2°$

In general, for singlepoint optimization, SHGO and Dual Annealing performed well, converging to solutions with a low final error. As Ukken used a different approach in optimization, such as a different CFD solver, and different optimization methods, the results are not directly comparable. However, the optimization results achieved by the MLP surrogate model show that, this approach is perfectly capable in performing singlepoint optimization.

4.3 Multipoint optimization

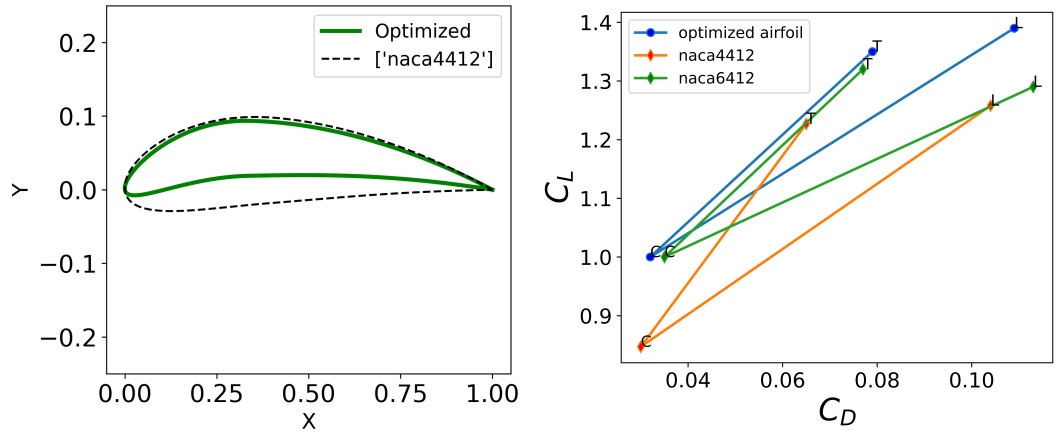
4.3.1 Optimizing for a flight profile

Table 12. Multipoint optimization results for a flight profile

Design Point	Airspeed (ms^{-1})	Reynolds number	α (°)	Design Weight w_i	$C_{L_{fixed}}$	$f_{multipoint}$	**Optimized airfoil** C_L	**Optimized airfoil** C_D	$\frac{C_{L_{fixed}}}{C_D}$
Takeoff	5.555	46921	10.9	$\frac{1}{6}$	1.3	$\frac{C_{L_{fixed}}}{C_D}$	**1.35**	**0.079**	**17.1**
Cruise	11.111	93850	5	$\frac{2}{3}$	1	$\frac{C_{L_{fixed}}}{C_D}$	**1**	**0.032**	**31.25**
Landing	4.444	37536	13.9	$\frac{1}{6}$	1.3	$\frac{C_{L_{fixed}}}{C_D}$	**1.39**	**0.109**	**12.8**

Results for the optimized airfoil for multipoint optimization are given in Table 12. The airfoil was optimized using the SHGO optimizer, and converged to

a solution $P1 = 5.625$, $P2 = 3.375$ and $P3 = 7.5$, using the 10 starting airfoils. The MLP predictions for each design point on Figure 20b show that, the optimized airfoil satisfied all $C_{L\,fixed}$ requirements, and possessed minimal C_D values. It was able to outperform airfoils NACA4412 and NACA6412, for the overall flight profile.



(a) Optimized airfoil geometry

(b) Results for 3 design points[1]

Figure 20. Multipoint optimization across 3 design points

[1] T=Takeoff, C=Cruise, L=Landing

The results obtained for multipoint optimization, demonstrates the capability of this approach in performing airfoil optimization. Optimization can be performed for multiple design points, subjected to constraints such as fixed C_L values. Crucially, this approach does not consume copious amounts of time. The main benefit of this approach, is primarily, the reduction of optimization time, summarized in Table 13.

Table 13. Comparison of CPU times

Computation type	MLP training time	MLP prediction time	Conventional methods [48]
CPU time (Intel i7-9700K at 4.7Ghz) CPU hours	0.15	3.33×10^{-5}	7.0

5 Conclusion

In this study, an approach was presented to perform airfoil optimization, using machine learning techniques. The approach consists of three steps. Firstly, a high-fidelity database consisting of CFD simulation data is first developed. The database is verified for accuracy and validated with existing literature. Secondly, Hyperparameter optimization is conducted, in order to implement the best MLP architecture and training settings. The MLP model is then trained, and verified for its performance in predicting aerodynamic coefficients. Lastly, the MLP is fed into an optimization script using SciPy, to perform airfoil optimization.

Performance of the final MLP shows that conducting hyperparameter optimization significantly improves the performance of the MLP, and that there is no hard and fast rule with determining the best hyperparameters. The performance of the final MLP shows that it performs aerodynamic coefficient predictions very accurately, being almost identical to the actual CFD values. The MLP is also able to predict aerodynamic coefficients of airfoils, outside of the database.

The results obtained by this approach shows that it is able to perform both singlepoint and multipoint optimization, without the usage of CFD in the optimization process. This approach was also able to conduct multipoint optimization with constraints, such as fixing the C_L value. The main benefit of this approach is the large reduction in time taken to perform airfoil optimization. Therefore, this approach opens a path for highly effective airfoil optimization, which conventional methods may struggle to perform.

6 Recommendations

6.1 Extension to transonic flow regime

Transonic flow is a challenging conundrum for airfoil optimization, as there is difficulty in incorporating both subsonic and supersonic flow theories simultaneously. Transonic flow wind tunnel experiments are also costly, as special equipment is required to generate sonic flow. Morever, it is a region where many commercial and military aircraft operate, and therefore is an impactful area of study. This study has shown that an ANN can perform well as a surrogate model for airfoil optimization, approximating the non linear RANS equations with acceptable accuracy. Thus, the prediction capability of ANNs can be explored in the future for transonic flow regimes, where it is more difficult to perform optimization.

6.2 Extension to other airfoil sections

The scope of this study only focuses on airfoil optimization for NACA four-digit wing sections, as it is easily parametrized into three individual numbers which describe the wing geometry entirely. In addition, parametrized wing sections allow for an easy optimization process. Thus, this study can be extended by including other NACA wing sections, such as the five-digit series and six-digit series. In addition, the NACA four-digit wing sections in the database can be extended to include edge cases like highly cambered airfoils, or airfoils with thicker sections.

7 References

[1] J. D. Anderson, *Fundamentals of aerodynamics*, 2nd ed, ser. McGraw-Hill series in aeronautical and aerospace engineering. New York: McGraw-Hill, 1991, 772 pp., ISBN: 978-0-07-001679-8.

[2] L. Zhu, W. Zhang, J. Kou, and Y. Liu, "Machine learning methods for turbulence modeling in subsonic flows around airfoils," *Physics of Fluids*, vol. 31, no. 1, p. 015 105, Jan. 1, 2019, Number: 1 Publisher: American Institute of Physics, ISSN: 1070-6631. DOI: 10.1063/1.5061693. [Online]. Available: https://aip-scitation-org.libproxy1.nus.edu.sg/doi/10.1063/1.5061693 (visited on 01/13/2021).

[3] A. P. Singh, S. Medida, and K. Duraisamy, "Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils," *AIAA Journal*, vol. 55, no. 7, pp. 2215–2227, Apr. 27, 2017, Number: 7 Publisher: American Institute of Aeronautics and Astronautics, ISSN: 0001-1452. DOI: 10.2514/1.J055595. [Online]. Available: https://arc-aiaa-org.libproxy1.nus.edu.sg/doi/10.2514/1.J055595 (visited on 01/13/2021).

[4] J. Ling, A. Kurzawski, and J. Templeton, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," *Journal of Fluid Mechanics*, vol. 807, pp. 155–166, Nov. 2016, Publisher: Cambridge University Press, ISSN: 0022-1120, 1469-7645. DOI: 10.1017/jfm.2016.615. [Online]. Available: http://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/reynolds-averaged-turbulence-modelling-using-deep-neural-networks-with-embedded-invariance/0B280EEE89C74A7BF651C422F8FBD1EB (visited on 01/13/2021).

[5] M. Frank, D. Drikakis, V. Charissis, and t. l. w. o. i. a. n. w. Link to external site, "Machine-learning methods for computational science and engineering," *Computation*, vol. 8, no. 1, p. 15, 2020, Number: 1 Num Pages: 15 Place: Basel, Switzerland Publisher: MDPI AG. DOI: http://dx.doi.org.libproxy1.nus.edu.sg/10.3390/computation8010015. [Online]. Available: http://search.proquest.com/publiccontent/docview/2373440829/abstract/F88F731C324C4429PQ/1 (visited on 01/13/2021).

[6] K. Fukami, K. Fukagata, and K. Taira, "Assessment of supervised machine learning methods for fluid flows," *Theoretical and Computational Fluid Dynamics*, vol. 34, no. 4, pp. 497–519, Aug. 1, 2020, Number: 4, ISSN: 1432-2250. DOI: 10.1007/s00162-020-00518-y. [Online]. Available:

https://doi.org/10.1007/s00162-020-00518-y (visited on 01/13/2021).

[7] E. Andrés-Pérez, "Data mining and machine learning techniques for aerodynamic databases: Introduction, methodology and potential benefits," *Energies (19961073)*, vol. 13, no. 21, p. 5807, Nov. 2020, Number: 21 Publisher: MDPI Publishing, ISSN: 19961073. DOI: 10.3390/en13215807. [Online]. Available: http://libproxy1.nus.edu.sg/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=147300142&site=ehost-live (visited on 01/13/2021).

[8] J. N. Kutz, "Deep learning in fluid dynamics," *Journal of Fluid Mechanics*, vol. 814, pp. 1–4, Mar. 2017, Publisher: Cambridge University Press, ISSN: 0022-1120, 1469-7645. DOI: 10.1017/jfm.2016.803. [Online]. Available: https://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/deep-learning-in-fluid-dynamics/F2EDDAB89563DE5157FC4B8342AD9C70 (visited on 01/13/2021).

[9] H. F. S. Lui and W. R. Wolf, "Construction of reduced-order models for fluid flows using deep feedforward neural networks," *Journal of Fluid Mechanics*, vol. 872, pp. 963–994, Aug. 2019, Publisher: Cambridge University Press, ISSN: 0022-1120, 1469-7645. DOI: 10.1017/jfm.2019.358. [Online]. Available: http://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/construction-of-reducedorder-models-for-fluid-flows-using-deep-feedforward-neural-networks/ECEC52E32AEBBEA049CF26D6C79EE394 (visited on 01/13/2021).

[10] S. Pawar, O. San, B. Aksoylu, A. Rasheed, and T. Kvamsdal, "Physics guided machine learning using simplified theories," *Physics of Fluids*, vol. 33, no. 1, p. 011 701, Jan. 1, 2021, Publisher: American Institute of Physics, ISSN: 1070-6631. DOI: 10.1063/5.0038929. [Online]. Available: http://aip.scitation.org/doi/full/10.1063/5.0038929 (visited on 01/14/2021).

[11] D. I. Ignatyev and A. N. Khrabrov, "Neural network modeling of unsteady aerodynamic characteristics at high angles of attack," *Aerospace Science and Technology*, vol. 41, pp. 106–115, Feb. 1, 2015, ISSN: 1270-9638. DOI: 10.1016/j.ast.2014.12.017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1270963814002703 (visited on 01/13/2021).

[12] N. Thuerey, K. Weißenow, L. Prantl, and X. Hu, "Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows," *AIAA Journal*, vol. 58, no. 1, pp. 25–36, 2020, Number: 1

Publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/1.J058291, ISSN: 0001-1452.
DOI: 10.2514/1.J058291. [Online]. Available: https://doi.org/10.2514/1.J058291 (visited on 01/13/2021).

[13] H. Eivazi, H. Veisi, M. H. Naderi, and V. Esfahanian, "Deep neural networks for nonlinear model order reduction of unsteady flows," *Physics of Fluids*, vol. 32, no. 10, p. 105 104, Oct. 1, 2020, Publisher: American Institute of Physics, ISSN: 1070-6631.
DOI: 10.1063/5.0020526. [Online]. Available: http://aip.scitation.org/doi/full/10.1063/5.0020526 (visited on 01/14/2021).

[14] J. Ling and J. Templeton, "Evaluation of machine learning algorithms for prediction of regions of high reynolds averaged navier stokes uncertainty," *Physics of Fluids*, vol. 27, no. 8, p. 085 103, Aug. 1, 2015, Publisher: American Institute of Physics, ISSN: 1070-6631.
DOI: 10.1063/1.4927765. [Online]. Available: http://aip.scitation.org/doi/full/10.1063/1.4927765 (visited on 01/14/2021).

[15] T. Agrawal, *Hyperparameter Optimization in Machine Learning: Make Your Machine Learning and Deep Learning Models More Efficient*. Apress, 2021, ISBN: 978-1-4842-6578-9.
DOI: 10.1007/978-1-4842-6579-6. [Online]. Available: https://www.springer.com/gp/book/9781484265789 (visited on 02/05/2021).

[16] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," p. 25,

[17] P. Probst, A.-L. Boulesteix, and B. Bischl, "Tunability: Importance of hyperparameters of machine learning algorithms," p. 32,

[18] N. Schilling, M. Wistuba, L. Drumond, and L. Schmidt-Thieme, "Joint model choice and hyperparameter optimization with factorized multilayer perceptrons," in *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, ISSN: 1082-3409, Nov. 2015, pp. 72–79. DOI: 10.1109/ICTAI.2015.24.

[19] M. H. M. Tarik, M. Omar, M. F. Abdullah, and R. Ibrahim, "Optimization of neural network hyperparameters for gas turbine modelling using bayesian optimization," in *5th IET International Conference on Clean Energy and Technology (CEAT2018)*, Sep. 2018, pp. 1–5. DOI: 10.1049/cp.2018.1308.

[20] W. Chen, K. Chiu, and M. D. Fuge, "Airfoil design parameterization and optimization using bézier generative adversarial networks," *AIAA Journal*, vol. 58, no. 11, pp. 4723–4735, 2020, Number: 11 Publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/1.J059317, ISSN: 0001-1452.

DOI: 10.2514/1.J059317. [Online]. Available: https://doi.org/10.2514/1.J059317 (visited on 01/13/2021).

[21] M. A. Bouhlel, S. He, and J. R. R. A. Martins, "Scalable gradient–enhanced artificial neural networks for airfoil shape design in the subsonic and transonic regimes," *Structural and Multidisciplinary Optimization*, vol. 61, no. 4, pp. 1363–1376, Apr. 1, 2020, Number: 4, ISSN: 1615-1488. DOI: 10.1007/s00158-020-02488-5. [Online]. Available: https://doi.org/10.1007/s00158-020-02488-5 (visited on 01/13/2021).

[22] A. Shahrokhi and A. Jahangirian, "A surrogate assisted evolutionary optimization method with application to the transonic airfoil design," *Engineering Optimization*, vol. 42, no. 6, pp. 497–515, Jun. 1, 2010, Number: 6 Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/03052150903305468, ISSN: 0305-215X. DOI: 10.1080/03052150903305468. [Online]. Available: https://doi.org/10.1080/03052150903305468 (visited on 01/13/2021).

[23] J. Li, M. Zhang, J. R. R. A. Martins, and C. Shu, "Efficient aerodynamic shape optimization with deep-learning-based geometric filtering," *AIAA Journal*, vol. 58, no. 10, pp. 4243–4259, 2020, Number: 10 Publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/1.J059254, ISSN: 0001-1452. DOI: 10.2514/1.J059254. [Online]. Available: https://doi.org/10.2514/1.J059254 (visited on 01/13/2021).

[24] R. M. Paiva, A. R. D. Carvalho, C. Crawford, and A. Suleman, "Comparison of surrogate models in a multidisciplinary optimization framework for wing design," *AIAA Journal*, vol. 48, no. 5, pp. 995–1006, 2010, Number: 5 Publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/1.45790, ISSN: 0001-1452. DOI: 10.2514/1.45790. [Online]. Available: https://doi.org/10.2514/1.45790 (visited on 01/13/2021).

[25] X. Hui, J. Bai, H. Wang, and Y. Zhang, "Fast pressure distribution prediction of airfoils using deep learning," *Aerospace Science and Technology*, vol. 105, p. 105 949, Oct. 1, 2020, ISSN: 1270-9638. DOI: 10.1016/j.ast.2020.105949. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1270963820306313 (visited on 01/13/2021).

[26] S. Suresh, S. N. Omkar, V. Mani, and T. N. Guru Prakash, "Lift coefficient prediction at high angle of attack using recurrent neural network," *Aerospace Science and Technology*, vol. 7, no. 8, pp. 595–602, Dec. 1, 2003, Number: 8, ISSN: 1270-9638. DOI: 10.1016/S1270-9638(03)00053-1. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1270963803000531 (visited on 01/13/2021).

[27] A. Boutemedjet, M. Samardžić, L. Rebhi, Z. Rajić, and T. Mouada, "UAV aerodynamic design involving genetic algorithm and artificial neural network for wing preliminary computation," *Aerospace Science and Technology*, vol. 84, pp. 464–483, Jan. 1, 2019, ISSN: 1270-9638. DOI: 10.1016/j.ast.2018.09.043. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1270963818302359 (visited on 01/13/2021).

[28] F. Mazhar, A. M. Khan, I. A. Chaudhry, and M. Ahsan, "On using neural networks in UAV structural design for CFD data fitting and classification," *Aerospace Science and Technology*, vol. 30, no. 1, pp. 210–225, Oct. 1, 2013, Number: 1, ISSN: 1270-9638. DOI: 10.1016/j.ast.2013.08.005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1270963813001454 (visited on 01/13/2021).

[29] M. A. Nielsen, "Neural networks and deep learning," 2015, Publisher: Determination Press. [Online]. Available: http://neuralnetworksanddeeplearning.com (visited on 03/17/2021).

[30] K. Yousefi and A. Razeghi, "Determination of the critical reynolds number for flow over symmetric NACA airfoils," in *2018 AIAA Aerospace Sciences Meeting*, Kissimmee, Florida: American Institute of Aeronautics and Astronautics, Jan. 8, 2018, ISBN: 978-1-62410-524-1. DOI: 10.2514/6.2018-0818. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2018-0818 (visited on 03/17/2021).

[31] J. Tao, "A data driven approach for fast airfoil analysis using machine learning technique," National University of Singapore, 2018.

[32] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-b: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software*, vol. 23, no. 4, pp. 550–560, Dec. 1, 1997, ISSN: 0098-3500. DOI: 10.1145/279232.279236. [Online]. Available: http://doi.org/10.1145/279232.279236 (visited on 03/10/2021).

[33] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing*, vol. 16, no. 5, p. 19, Sep. 1995, Num Pages: 19 Place: Philadelphia, United States Publisher: Society for Industrial and Applied Mathematics, ISSN: 10648275. DOI: http://dx.doi.org.libproxy1.nus.edu.sg/10.1137/0916069. [Online]. Available: http://search.proquest.com/docview/921565075/abstract/F42F7D2AC4A948C7PQ/1 (visited on 03/10/2021).

[34] D. Kraft and DFVLR-FB, *A software package for sequential quadratic programming*. Köln, 1988.

[35] "Quasi-newton methods," in *Numerical Optimization*, ser. Springer Series in Operations Research and Financial Engineering, J. Nocedal and S. J. Wright, Eds., New York, NY: Springer, 2006, pp. 135–163, ISBN: 978-0-387-40065-5. DOI: 10.1007/978-0-387-40065-5_6. [Online]. Available: https://doi.org/10.1007/978-0-387-40065-5_6 (visited on 03/24/2021).

[36] S. G. Nash, "Newton-type minimization via the lanczos method," *SIAM Journal on Numerical Analysis*, vol. 21, no. 4, pp. 770–788, 1984, Publisher: Society for Industrial and Applied Mathematics, ISSN: 00361429. [Online]. Available: http://www.jstor.org.libproxy1.nus.edu.sg/stable/2157008 (visited on 03/24/2021).

[37] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The Computer Journal*, vol. 7, no. 2, pp. 155–162, Jan. 1, 1964, ISSN: 0010-4620. DOI: 10.1093/comjnl/7.2.155. [Online]. Available: https://doi.org/10.1093/comjnl/7.2.155 (visited on 03/10/2021).

[38] L. F. Shampine, "Numerical recipes, the art of scientific computing. by w. h. press, b. p. flannery, s. a. teukolsky, and w. t. vetterling," *The American Mathematical Monthly*, vol. 94, no. 9, pp. 889–893, Nov. 1, 1987, Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00029890.1987.12000737, ISSN: 0002-9890. DOI: 10.1080/00029890.1987.12000737. [Online]. Available: https://doi.org/10.1080/00029890.1987.12000737 (visited on 03/10/2021).

[39] B. Olson, I. Hashmi, K. Molloy, and A. Shehu, "Basin hopping as a general and versatile optimization framework for the characterization of biological macromolecules," *Advances in Artificial Intelligence*, vol. 2012, e674832, Dec. 4, 2012, Publisher: Hindawi, ISSN: 1687-7470. DOI: 10.1155/2012/674832. [Online]. Available: https://www.hindawi.com/journals/aai/2012/674832/ (visited on 03/10/2021).

[40] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, Jan. 12, 1997. DOI: 10.1023/A:1008202821328.

[41] S. C. Endres, C. Sandrock, and W. W. Focke, "A simplicial homology algorithm for lipschitz optimisation," *Journal of Global Optimization*, vol. 72, no. 2, pp. 181–217, Oct. 1, 2018, ISSN: 0925-5001. DOI: 10.1007/s10898-018-0645-y. [Online]. Available: https://doi.org/10.1007/s10898-018-0645-y (visited on 03/24/2021).

[42] Y. Xiang and X. G. Gong, "Efficiency of generalized simulated annealing," *Physical Review E*, vol. 62, no. 3, pp. 4473–4476, Sep. 1, 2000, Publisher: American Physical Society. DOI: 10.1103/PhysRevE.62.4473. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.62.4473 (visited on 03/10/2021).

[43] Y. Xiang, S. Gubian, B. Suomela, and J. Hoeng, "Generalized simulated annealing for global optimization: The GenSA package," *The R Journal*, vol. 5, no. 1, p. 13, 2013, ISSN: 2073-4859. DOI: 10.32614/RJ-2013-002. [Online]. Available: https://journal.r-project.org/archive/2013/RJ-2013-002/index.html (visited on 03/10/2021).

[44] M. Nemec, D. W. Zingg, and T. H. Pulliam, "Multipoint and multi-objective aerodynamic shape optimization," *AIAA Journal*, vol. 42, no. 6, pp. 1057–1065, Jun. 2004, ISSN: 0001-1452, 1533-385X. DOI: 10.2514/1.10415. [Online]. Available: https://arc.aiaa.org/doi/10.2514/1.10415 (visited on 03/20/2021).

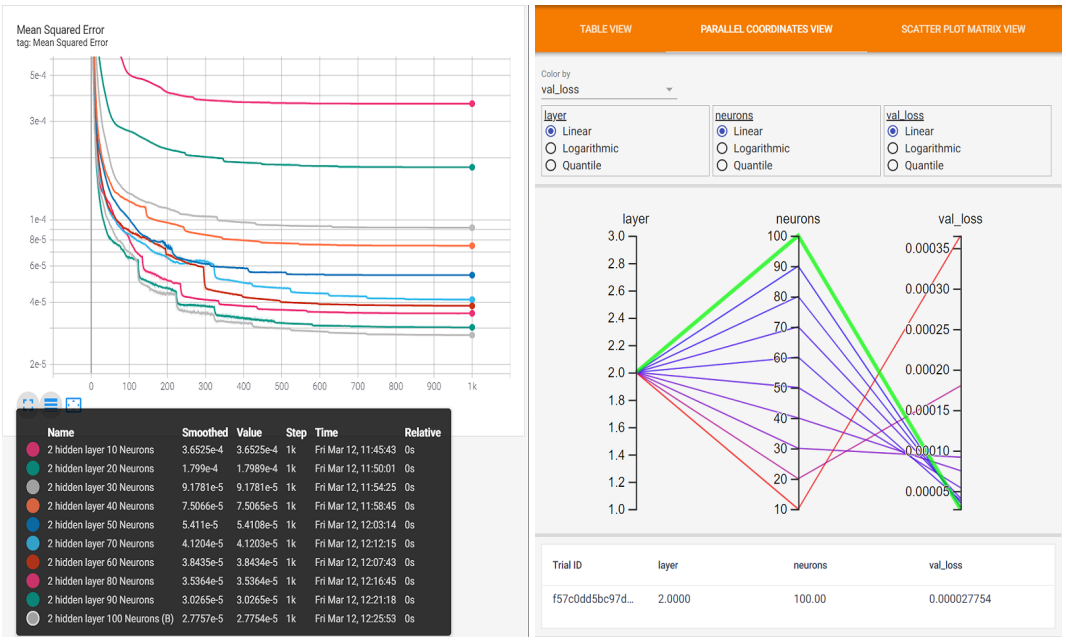
[45] M. G. Ukken and M. Sivapragasam, "Aerodynamic shape optimization of airfoils at ultra-low reynolds numbers," *Sādhanā*, vol. 44, no. 6, p. 130, Jun. 2019, ISSN: 0256-2499, 0973-7677. DOI: 10.1007/s12046-019-1115-z. [Online]. Available: http://link.springer.com/10.1007/s12046-019-1115-z (visited on 03/19/2021).

[46] R. Ajaj, E. Saavedra Flores, M. Friswell, A. Isikveren, G. Allegri, S. Adhikari, and H. Chaouk, "An integrated conceptual design study using span morphing technology," *Journal of Intelligent Material Systems and Structures*, vol. 25, pp. 989–1008, Apr. 22, 2014. DOI: 10.1177/1045389X13502869.

[47] J. McEvoy, G. Hall, and P. Mcdonald, "Evaluation of unmanned aerial vehicle shape, flight path and camera type for waterfowl surveys: Disturbance effects and species recognition," *PeerJ*, vol. 4, e1831, Mar. 21, 2016. DOI: 10.7717/peerj.1831.

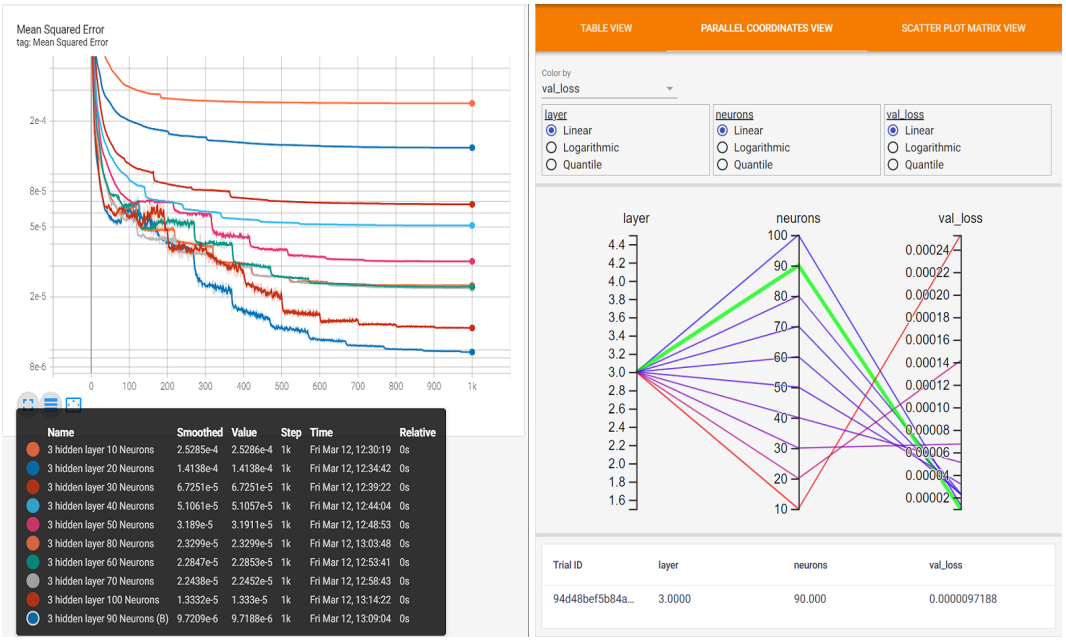
[48] E. Tashnizi, A. Akhavan Taheri Borojeni, and M. H. Hekmat, "Investigation of the adjoint method in aerodynamic optimization using various shape parameterization techniques," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 32, pp. 176–186, Jun. 1, 2010. DOI: 10.1590/S1678-58782010000200012.

8 Appendices



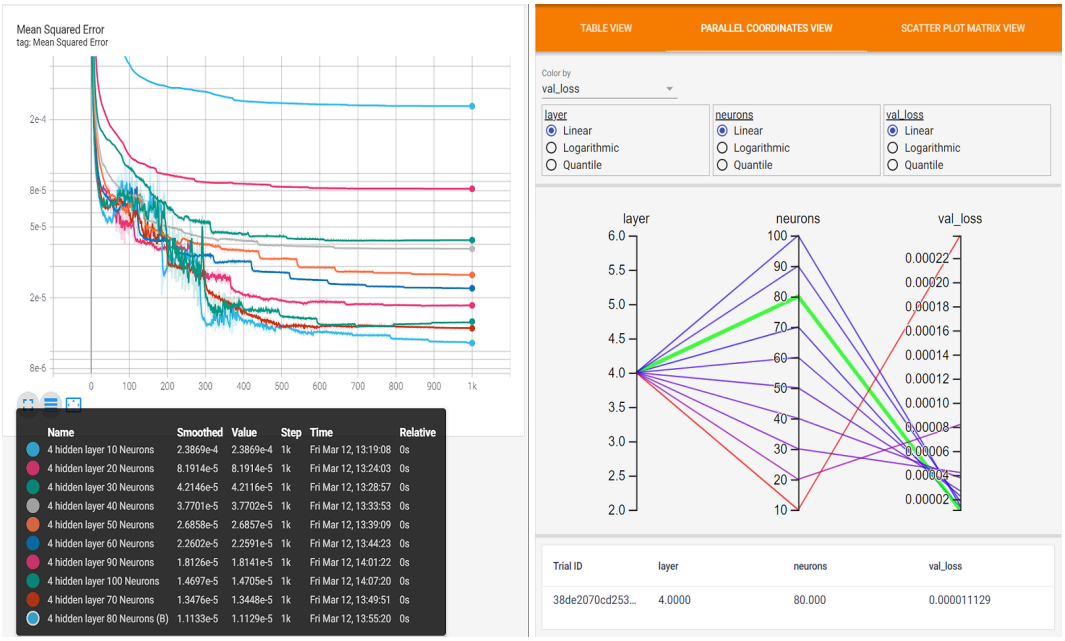
(a) Validation MSE vs epoch (b) Final validation MSE

Figure 21. Hyperparameter study of various neurons in 2 hidden layers



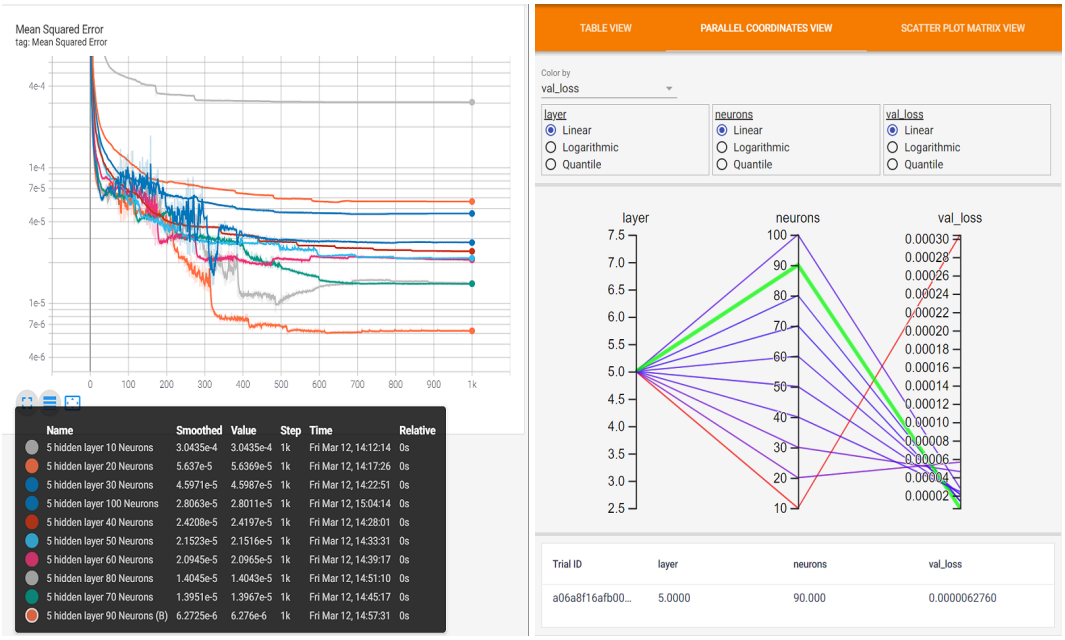
(a) Validation MSE vs epoch (b) Final validation MSE

Figure 22. Hyperparameter study of various neurons in 3 hidden layers



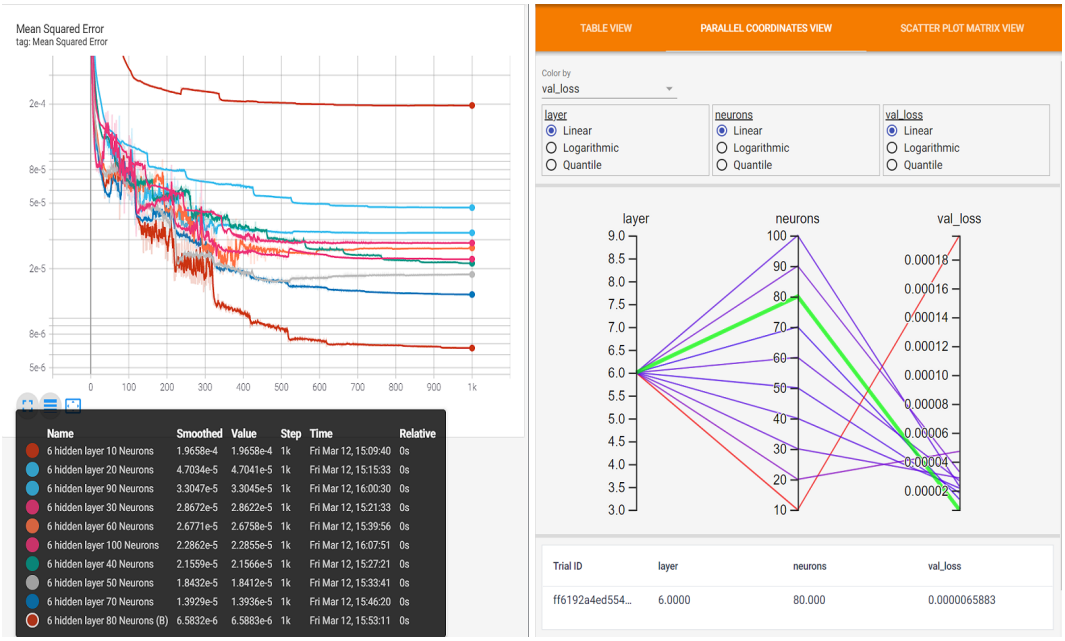
(a) Validation MSE vs epoch (b) Final validation MSE

Figure 23. Hyperparameter study of various neurons in 4 hidden layers



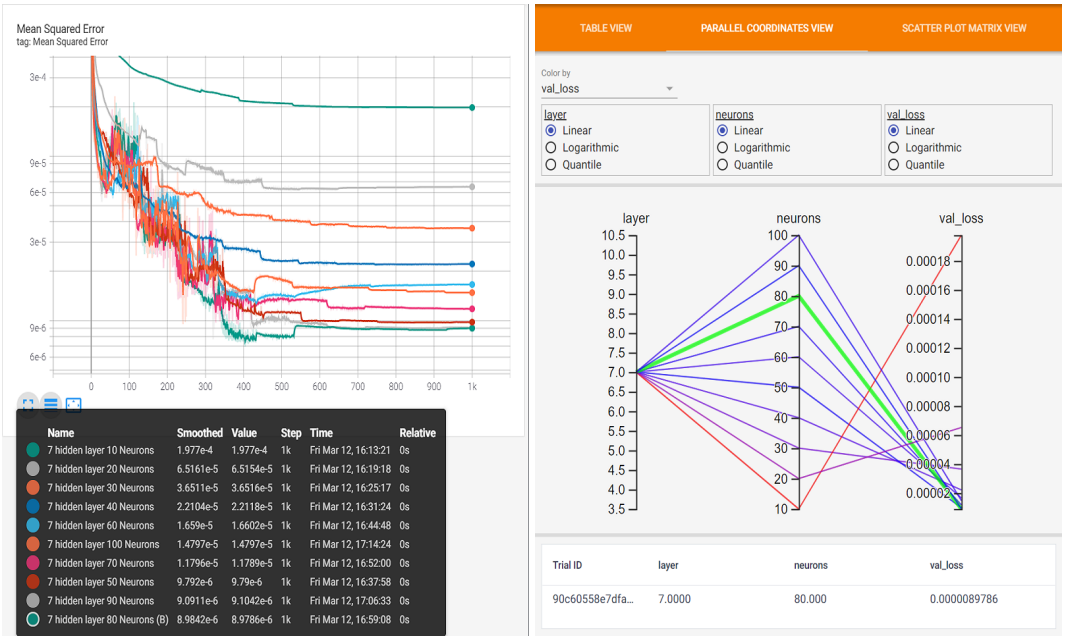
(a) Validation MSE vs epoch (b) Final validation MSE

Figure 24. Hyperparameter study of various neurons in 5 hidden layers



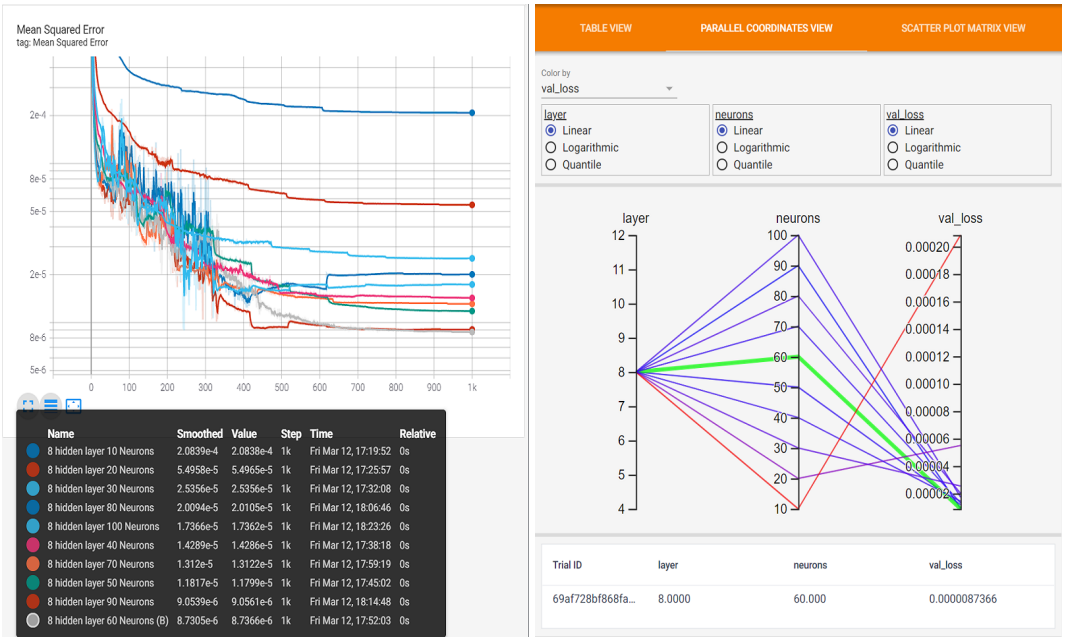
(a) Validation MSE vs epoch (b) Final validation MSE

Figure 25. Hyperparameter study of various neurons in 6 hidden layers



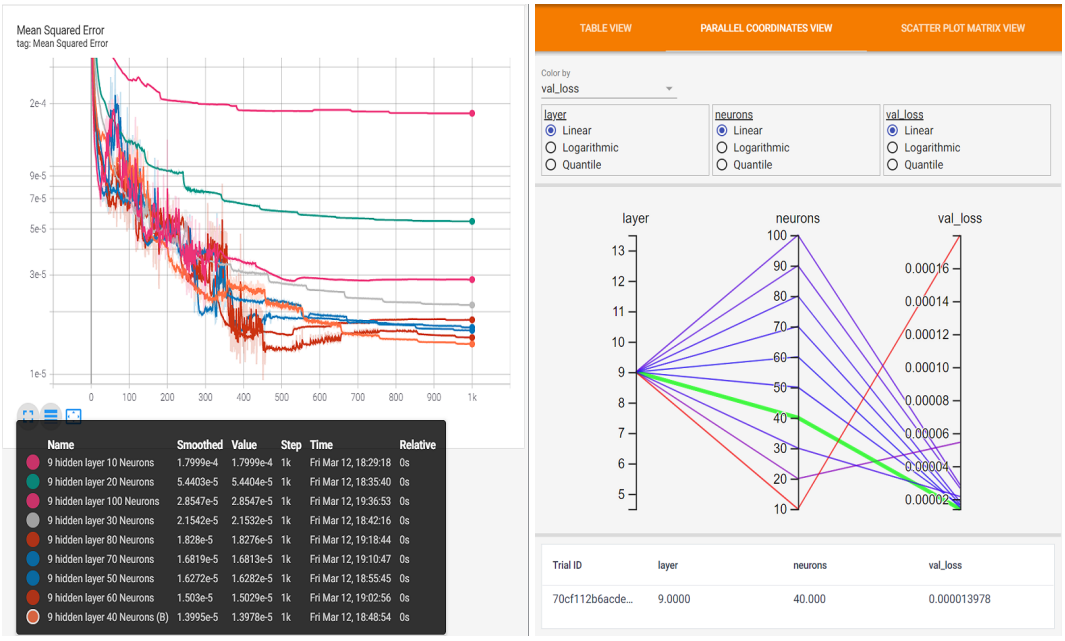
(a) Validation MSE vs epoch (b) Final validation MSE

Figure 26. Hyperparameter study of various neurons in 7 hidden layers



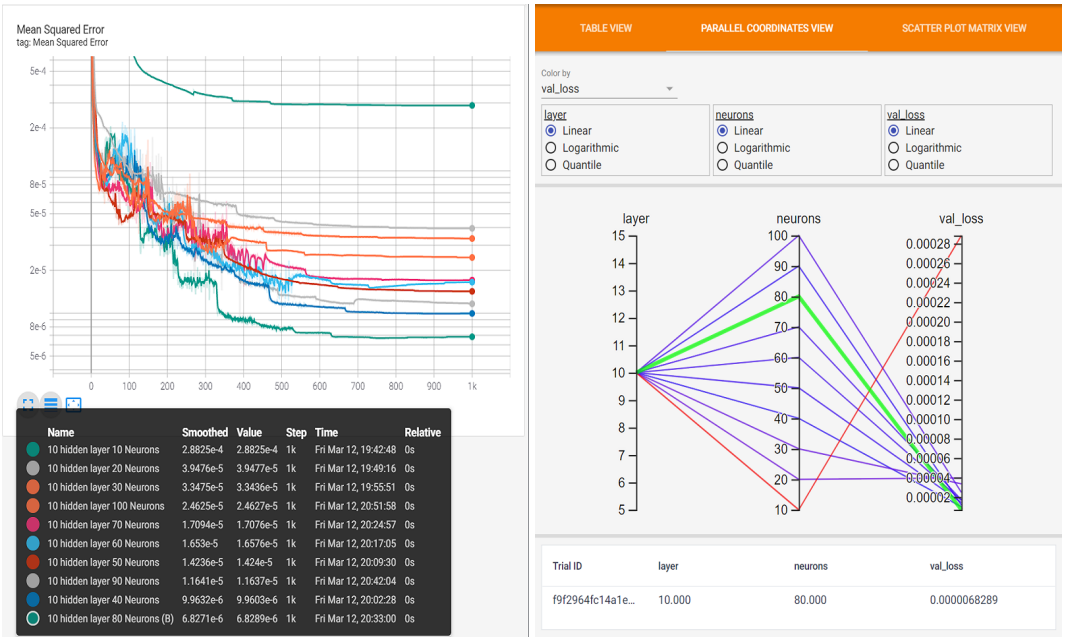
(a) Validation MSE vs epoch (b) Final validation MSE

Figure 27. Hyperparameter study of various neurons in 8 hidden layers



(a) Validation MSE vs epoch (b) Final validation MSE

Figure 28. Hyperparameter study of various neurons in 9 hidden layers



(a) Validation MSE vs epoch (b) Final validation MSE

Figure 29. Hyperparameter study of various neurons in 10 hidden layers