Exploring the Use of Artificial Neural Networks for Prediction of Flight Dynamics Characteristics Based on DATCOM-generated Datasets

Gavin Feeney and Michael McCarthy
School of Engineering
University of Limerick, Limerick, Ireland

Abstract
A series of Artificial Neural Networks utilising the DeepLearning4J library and based on the aircraft stability characteristic estimation function of Datcom+ Pro are developed, and applied to several variably-sized input-output cases. Both Mesh-On-Object and Object-On-Mesh approaches are considered, using stochastic variations of the stock Boeing 737-300 Datcom+ Pro model. The Artificial Neural Network approach demonstrates strong correlation with the expected output relative to the size of the learning dataset, with indications of scalability to problems such as Computational Fluid Dynamics and Finite Element Analysis. The potential capability of Artificial Neural Networks with more robust and flexible aircraft performance estimation software is illustrated by the ability to combine results for multiple sections.
Keywords: Artificial Neural Networks; DATCOM; Stability & Control, Computational Fluid Dynamics.

1 Introduction

The stability and control characteristics of aircraft have been studied for several decades, using methods ranging from the semi-empirical approaches in the DATCOM report of the 1960s [1] through to advanced Computational Fluid Dynamics (CFD) techniques of the present day [2-4]. Such methods are used to predict the so-called "stability derivatives" of an aircraft from its geometry and flight conditions. These stability derivatives are the rates of change of forces and moments with respect to flight variables such as speed, angle of attack, pitch rate and so on. They allow the prediction of the aircraft flight dynamics and are central to the methodologies employed by several commercial flight simulation programmes. Datcom+ Pro [5] is a computer program implementing the DATCOM methods, essentially the same as the DIGITAL DATCOM program produced in the 1970s, with some enhanced input and output features. The program has been shown to perform quite well in comparison with accurate techniques that use the Navier-Stokes equations for predicting stability derivatives of conventionally-shaped aircraft [6-7]. However, predictions for unconventional aircraft, or performance in non-linear flight regimes, require computationally expensive CFD models with dense meshes.

The capability of Artificial Neural Networks (ANNs) to solve a wide range of problems has increased dramatically recently with advances in both hardware and software [8-10]. ANNs are being utilised for a wide range of applications from image and voice recognition, to medical imaging and the playing of board/video games [11-16]. Of interest here is the capability of training an ANN to perform the same function as a mathematical formula, even if the formula itself is not known [17]. Jin [18] refers to this as function approximation, whereby the ANN can predict the correct output for a given input either inside or outside the training range. Research exists on the topic of comparing CFD results with an equivalent ANN such as in Alizadehdakhel et al. [19] although the ANNs utilised have been extremely small in both size and scope, and generally reduce the input and output from large 2- or 3-dimensional datasets of variables, to a small select number of variables which describe the inflow and outflow. An approach known as Kriging, whereby a large stochastic problem is approximated by a mathematical model, has also been shown to provide good agreement with Finite Element Analysis (FEA) and experimental results [20-24] with the use of ANNs. However, the models created are again limited in size and scope, and restricted to small sets of boundary conditions. El-Beltagy et al. [25] illustrates several issues regarding the inflexibility of current FEA and CFD approaches involving ANNs, such as varying the size and shape of the problem. Jin [18] highlights these points when discussing the possible techniques for performing function approximation with complex problems. Fernández et al. [26] investigated which functional approximation technique is best when employed to approximate an FEA bolted-connection failure problem, and concluded that Multi-Layer Perceptrons (MLPs), which are a type of ANN, are the best approach. The output in Fernández et al. was limited to three variables describing the forces and stresses at failure.

1.1 Argument for the use of ANNs

ANNs are known as a "black box" approach as the internal workings of a large ANN are usually intractable [27]. Algorithms have been created for constructing decision trees from ANNs [28] and hence achieving "grey box" status, although no simple formulaic form exists. Despite this limitation, ANNs have the advantage of requiring no knowledge of the underlying equations if used for function approximation, and a more efficient approach can be created for an acceptably accurate result [18]. In comparison to CFD where large problems can take weeks to solve, the largest ANNs predict results within the magnitude of milliseconds to seconds [29]. However, these ANNs can take days or weeks to train, but significant progress has been made recently in dedicated hardware [9]. For ANNs that train using computer-generated datasets, techniques exist to allow training to execute in parallel to dataset generation. DeepLearning4J [30], which is the open-source library for ANN code used in this paper, includes this feature although no literature to date appears to have employed this technique with CFD or FEA problems.

1.2 Objectives and Structure

This paper aims to investigate the feasibility of utilising ANNs for the accurate prediction of aircraft stability characteristics. The performance of ANNs will be evaluated on the following topics: input and output size, input format, training dataset size, aircraft section combination, and parallel or loop learning. As per Fernández et al. [26], MLPs will be the type of ANN used. As generating a large stochastic full-aircraft dataset using CFD would require excessive computational and time resources, the datasets will be generated using Datcom+ Pro. This paper is organised as follows: Firstly, the ANN methodology and a sample of the flight dynamics governing equations are presented. Next the results are shown and discussed. Finally, the conclusions of the report and future work suggestions are given. Overall, the ANN approach shows good performance and is mostly limited by issues with Datcom+ Pro. Strong indications of the potential for using ANNs with more accurate aircraft dynamics analysis tools such as CFD are illustrated. All source code is written in Java using the DeepLearning4J library and available at github.com/gavinUL/NNDatcom under the Apache 2.0 license.

2 Governing Equations for Datcom+ Pro Analysis

Datcom+ Pro implements the methods in the original DATCOM report [1], which consisted of equations, graphical methods and look-up tables. It is not possible to define the methods for each derivative here due to the extreme length of that report. However, to illustrate the complexity of the formulae that the ANN must emulate, an abridged version of the method to calculate one derivative ($C_{L\alpha,WB}$) is shown. $C_{L\alpha,WB}$ is the partial derivative of the coefficient of lift with respect to angle of attack for a combined wing and body, and is calculated from [31]:

$$C_{L\alpha,WB} = \left[K_N + K_{W(B)} + K_{B(W)}\right] C_{L\alpha,e} \frac{S_{exp}}{S} \tag{1}$$

Here K_N, $K_{W(B)}$ and $K_{B(W)}$ are the ratios of the nose lift, the wing-in-the-presence-of-the-body lift, and the body-in-the-presence-of-the-wing lift respectively, to the exposed (that portion outside the fuselage) wing lift, $C_{L\alpha,e}$. S_{exp} and S are the exposed wing area and theoretical wing area (i.e. including area inside fuselage) respectively. K_N is obtained from:

$$K_N = \left(\frac{C_{L\alpha,N}}{C_{L\alpha,e}}\right)\frac{S}{S_{exp}} \tag{2}$$

where:

$$C_{L\alpha,N} = \frac{2(k_2 - k_1)S_{B,max}}{S} \tag{3}$$

Here $S_{B,max}$ is the maximum frontal area of the body (fuselage), and $k_2 - k_1$ is the "apparent mass constant". $k_2 - k_1$ is obtained from a graph or equivalent look-up table with fineness ratio of the nose as input, where:

$$Fineness\ ratio = \frac{l_n}{b_{f,max}} \tag{4}$$

Here l_n is the length of the nose, and $b_{f,max}$ is the maximum width of the fuselage. Both $K_{W(B)}$ and $K_{B(W)}$ are calculated from look-up tables although they can also be approximated by:

$$K_{W(B)} = 0.1714\left(\frac{b_{f,max}}{b}\right)^2 + 0.8326\left(\frac{b_{f,max}}{b}\right) + 0.9974 \tag{5}$$

$$K_{B(W)} = 0.7810\left(\frac{b_{f,max}}{b}\right)^2 + 1.1976\left(\frac{b_{f,max}}{b}\right) + 0.0088 \tag{6}$$

Here b is the wing span of the aircraft. Finally, the lift-curve slope of the exposed wing can be calculated as:

$$C_{L\alpha,e} = \frac{2\pi A_e}{2 + \sqrt{\frac{{A_e}^2\beta^2}{k^2}\left(1 + \frac{\left(\tan\Lambda_{c/2}\right)^2}{\beta^2}\right)}} \tag{7}$$

Here A_e is the exposed aspect ratio, defined as:

$$A_e = \frac{b^2}{S_{exp}} \tag{8}$$

β is defined as $\sqrt{1 - M^2}$ where M is the Mach number, k is defined as $a_0/2\pi$ where a_0 is the two-dimensional (i.e. airfoil) lift-curve slope, and $\Lambda_{c/2}$ is the sweep at half-chord. The two-dimensional lift-curve slope is given by:

$$a_0 = \frac{1.05}{\beta}\left[\frac{a_0}{(a_0)_{theory}}\right](a_0)_{theory} \tag{9}$$

The parameter $a_0/(a_0)_{theory}$ represents the empirical correction factor for viscous effects and is read from a look-up table, and varies with Reynold's number. The trailing-edge angle ϕ'_{TE} is dependent on airfoil geometry. $(a_0)_{theory}$ refers to the two-dimensional theoretical (excluding viscous effects) lift-curve slope, which is also found from a look-up table requiring the thickness of the wing. Adding to the complexity, equations (2-9) are not applicable to all cases. For example, (6) only holds for supersonic speeds if the following inequality is true:

$$\beta A_e(1 + \lambda_e)[(\tan\Lambda_{LE}/\beta) + 1] \geq 4 \tag{10}$$

A description of a sample of coefficients and derivatives is shown in ***Table 1***. Coefficients have only one subscript and are a dimensionless measure of a certain value such as lift, drag, and moment. Derivatives have two subscripts and are also a dimensionless measure of the change in the first value due to a change of the second value. No control derivatives (e.g. due to elevator or aileron inputs) will be analysed due to the large amount of time and complexity required to achieve a relatively small amount of additional knowledge.

Symbol	Explanation	Symbol	Explanation
C_L	Lift coefficient	C_{m_q}	Pitch moment derivative w.r.t. pitch velocity
C_d	Drag coefficient	$C_{L_{\dot{\alpha}}}$	Lift force derivative w.r.t. rate of change of angle of attack
C_m	Pitching moment coefficient	$C_{m_{\dot{\alpha}}}$	Pitch moment derivative w.r.t. rate of change of angle of attack
C_{y_β}	Side force derivative w.r.t. sideslip angle	C_{l_p}	Roll moment derivative with respect to roll velocity
C_{l_β}	Roll moment derivative w.r.t. sideslip angle	C_{n_r}	Yaw moment derivative w.r.t. yaw velocity
C_{L_q}	Lift force derivative w.r.t. pitch velocity	C_{l_r}	Roll moment derivative w.r.t. yaw velocity

Table 1 *- Explanation of common coefficients and derivatives for aircraft performance and stability analysis.*

The importance of each derivative and coefficient are not of great significance to this paper, as the ability of an ANN to accurately predict the values of the derivatives and coefficients is the main topic of the paper. Of interest for aircraft section combination is equation (1) as it illustrates interference effects which do not make combining sections a simple case of addition.

3 Governing Equations for ANNs

An illustration of a simple ANN is shown for reference in ***Fig.1***. All ANNs used in this paper are Multi-Layer Perceptrons (MLPs).

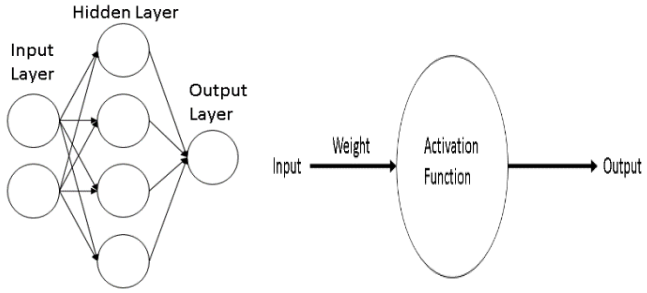

Fig.1 *- Illustration of a simple artificial neural network with fully-connected nodes (left), and a view of each node (right).*

The equation which describes the mathematical function at a node (the circles in ***Fig.1***) is:

$$y = f\left(\sum x_i w_i\right) \tag{11}$$

Here y is the output from the node, x is the input to the node, w refers to the weighting for the connection, and f is the activation function. The input can be from a previous layer, or else for the first layer it is the values of the training dataset. The ANN learns by adjusting the weightings for each connection so that the output matches the expected output, based on the gradients described:

$$\frac{dE}{dw} = \frac{dE}{dA} \times \frac{dA}{dw} \tag{12}$$

Here, E is the error for the entire ANN and A is the activation function. Common activation functions, such as sigmoid and hyperbolic tangent, are detailed in Appendix A. Terms such as "learning rate" and "batch size" which will be used throughout the report are described in **Table 2.**

Term	Description
Learning Rate	The rate at which old beliefs are forgotten in favour of new beliefs.
Batch Size	The number of samples passed to the ANN from which it learns.
Epoch	A complete run-through of all the batches in a dataset
Training Dataset	The dataset which the ANN learns to solve with.
Validation Dataset	The dataset which measures how well the ANN has learned to solve a problem.

Table 2 *- Explanation of common ANN-related terms.*

3.1 ANN Performance Evaluation

There are several methods of analysing the performance of an ANN, ranging from measures which can only be compared to models with the same input type (e.g. different magnitudes of error expected), to measures which can give an indication of accuracy regardless of input. An example of a measure that is dependent on the input type is the Mean Absolute Error (MAE), as described in:

$$MAE = \frac{\sum_{i=1}^{n} |p_i - a_i|}{n} \tag{13}$$

Here, n is the number of samples, p_i is the predicted answer, and a_i is the actual answer. The closer the value is to zero, the more accurate the model is. Mean Squared Error (MSE) and Root-Mean Squared Error (RMSE) are two similar measures of model accuracy.

$$MSE = \frac{\sum_{i=1}^{n} (p_i - a_i)^2}{n} \tag{14}$$

$$RMSE = \sqrt{MSE} \tag{15}$$

RMSE produces results that are similar in magnitude to MAE although slightly larger. RMSE is the most commonly used measure of the three, although with maximum expected values for accuracy (such as caused by decimal place precision) RMSE becomes less useful than MAE. The coefficient of determination R^2 (how well the predicted output "curve" fits the real output "curve") is an important general tool used to assess the performance of ANNs. A value of 1 indicates perfect fitting, while a value of 0 indicates that the ANN does not explain the output at all. It is defined as:

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST} \tag{16}$$

Here, SSR refers to the sum of squares regression, where y' is the predicted value, and $\bar{y'}$ is the mean predicted value.

$$SSR = \sum (y' - \bar{y'})^2 \tag{17}$$

SST refers to the sum of squares total, where y is the actual value and $\bar{y}$ is the mean actual value.

$$SST = \sum (y - \bar{y})^2 \tag{18}$$

SSE refers to the sum of squares error:

$$SSE = \sum (y - \bar{y'})^2 \tag{19}$$

Negative values can also exist for R^2 which indicate that taking the average of the output is more accurate than using the ANN's predictions. It is important as it can be used to compare models whose errors are measured in different units and scales. The final measure, which is used by DeepLearning4J to find the best performing model, is the score which is a summation of the MSE of all the outputs, defined as:

$$Score = \sum_{i=1}^{m} MSE_m \tag{20}$$

Here m refers to each output.

4 Datcom Generated Datasets

To generate a dataset of aircraft combinations with random characteristics, the variables of the stock 737-300 model were altered stochastically within defined ranges. Ranges were required because certain combinations would either cause errors in Datcom+ Pro or otherwise do not make sense. For example, the semi-span of the portion of the wing outside the breakpoint (seen in ***Fig.2***) being larger than the full semi-span itself. The variables can be summarised into four groups:

1. Flight conditions i.e. altitude, Mach number.
2. Body i.e. the size and shape of the fuselage.
3. Wing and tail i.e. the size, shape, position of wing, horizontal stabiliser, vertical stabiliser.
4. Airfoils for the wing and tail surfaces.

In total, 132 values in the base input file could be stochastically varied if all four groups were enabled. To prevent errors when running the files, certain restrictions were put in place. The

boundary conditions were kept within normal limits, such as keeping the Mach number subsonic. The body was scaled between 75% and 150% of the stock size in terms of both cross-sectional area and length, although allowances had to be made for tail placement as a result. Similarly, the values for wings and tail were scaled between 50% and 200% of their stock values, with exceptions made for overlapping values such as the semi-span and outer panel semi-span as seen in ***Fig.2*** (only a sample of values related to the wing planform are shown, excluded are variables such as sweep and incidence). The range for all variables is documented in Appendix G.

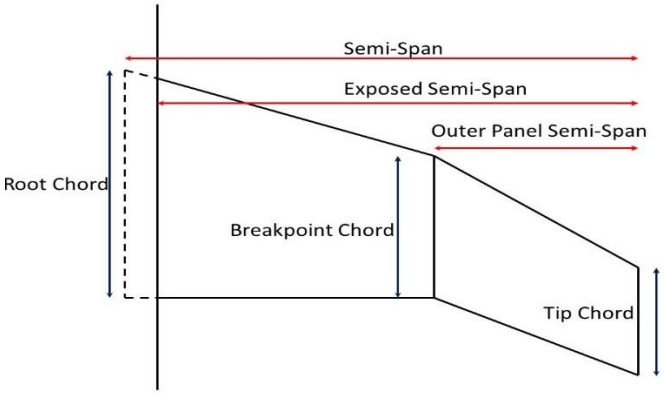

Fig.2 *- Illustration of a subset of the wing planform measurements, with dotted lines indicating hidden inside fuselage.*

The dotted lines in ***Fig.2*** indicate the section of the wing inside the fuselage, as measurements are taken from the centreline. Finally, all airfoils were kept as NACA 4-series airfoils for standardising input parameters to the ANN. The tail airfoils were kept symmetrical with their thickness varied, and the airfoil for the wing had both its camber and thickness values varied. ***Fig.3*** shows a simplification of how a Datcom+ Pro dataset is created.

```
Begin
while(number of combinations<required combinations)
    Generate and keep log of random variables
    Create and run Datcom+ Pro input file using variables
    Import result from output
end while
Remove errors
Scale input and output between 0 and 1
Send all data to .csv file
End
```

Fig.3 *- Pseudocode describing operation of dataset creation.*

As mentioned previously, certain combinations of variables in Datcom+ Pro can result in errors. Some errors are reported and caught by Datcom+ Pro itself, and these results are easily removed from the analysis. In other cases, the errors are not reported but the output is clearly incorrect. These errors can include but are not limited to:

- No values being output at all.
- All values output being equal to zero.
- Values that are orders of magnitude outside of expected ranges (some combinations output coefficients of lift as high as 58.3, where a value above 1.6 would be improbable given the range of airfoils used).

Code was written to handle the most obvious incorrect outputs although it is likely that a percentage of the remaining outputs may contains errors that exist within the range of normal values. The source of the errors was not pinpointed after logging of the blacklisted combinations, and in some cases running these combinations individually did not always result in an error or the same output. There is no documentation on the cause of these errors, although the Datcom+ Pro manual describes several avoidable error cases [5].

4.1 Dataset Types

The four input types and three output types considered in this paper are shown in ***Table 3***. Each input type is chosen to emulate a standard input case for techniques such as FEA and CFD, where they are usually much larger in size than the input to Datcom+ Pro. Both techniques employ meshes to solve the respective problems which can have millions of nodes and elements depending on the size and required accuracy of the model. In contrast, Datcom+ Pro typically involves the use of less than 150 variables to fully describe the aircraft and the flight conditions for the analysis, an example of these variables being shown in ***Fig.2.***

Input Cases		Output Cases	
A	Variable-Only	I	Ten "Low Granularity" Outputs
B	Coordinates-Only	II	Single Output Only
C	2D Binary Image-Only	III	Two Outputs Only

Table 3 *- Summary of Input and Output cases to be evaluated.*

The standard Datcom+ Pro input format will be represented by the variable-only Case A. Similar to how an FEA mesh describes fully the outline of the model of interest, Case B will consist of a large number of {X, Y} coordinates which mark out the outline of the aircraft when looking from above. The number of points that make up the outline will be varied and the effect of changing this number understood. Although FEA meshes include nodes inside the boundary of the model, only coordinates that represent the outline will be used to differentiate from Case C. ***Fig.4*** illustrates the difference between the coordinate and 2D binary input approaches, where Case C is seen to consist of a 999-pixel high by 999-pixel wide binary image of the aircraft from above. This creates an input size on the scale of techniques involving meshes, as a total of 998,001 pixels are used. Case B uses predefined spacing, but usually has an input size of magnitude 10^3.

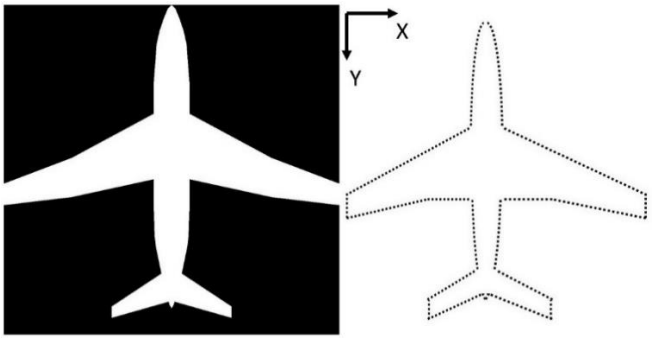

Fig. 4 *- Illustration of 2D binary input (left) where the white area corresponds to a value of 1 and black is zero, and the coordinates input (right) where each dot has [x,y] coordinates corresponding to the outline of the aircraft.*

Due to the nature of Cases B and C, certain inputs such as the flight conditions cannot be represented. However, ANNs can handle different data types if they are all scaled approximately between 0 and 1. To demonstrate this, a combination of Case

A and Case B will be evaluated. This case will approximately correspond to how FEA and similar techniques have input values such as material characteristics (e.g. Young's Modulus) which do not refer to the model or mesh shape.

4.2 Object on Mesh and Mesh on Object

There are two methods of applying a mesh: positioning an object on a predefined mesh (OOM), or fitting a mesh on to an object (MOO). An illustration explaining both methods is provided in ***Fig.5.***

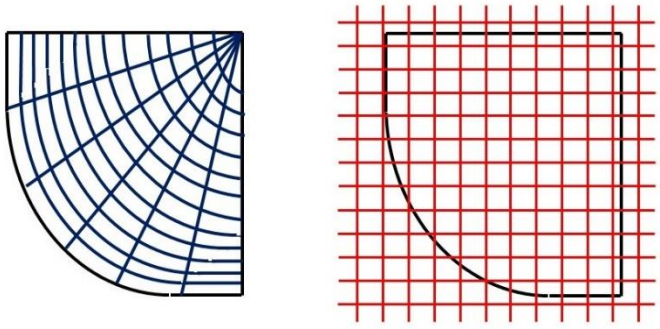

Fig.5 *– Illustration of a sample problem meshed using Mesh on Object (left) and Object on Mesh (right). Coarse meshing is used for illustration purposes.*

In general, it is easier to solve a problem with MOO than OOM, as fitting the mesh to the object can describe the object more accurately, and it provides the option to refine the mesh in areas of interest. Case C is intended to represent an OOM approach as an image is a uniform arrangement of pixels. Case B is the closest analogue to MOO as the coordinates are shaped to the aircraft just as a mesh is shaped to the object in MOO. Of interest is the potential of OOM to form the basis of a general solver, as the mesh is uniform and hence the input-output for an ANN would always be the same size.

5 Neural Network Modelling

The simplified code behind how the ANN method trains is described in ***Fig.6***.

```
Begin
Import Training and Validation Data
Initialise Artificial Neural Network
while(terminating condition is not satisfied)
    Fit training set in batches by adjusting weights
    Find best scoring model on validation set after each batch
    Apply learning rate decay
    Move to next epoch
end while
Export best model to hard disk
Assess accuracy of best model for each output on validation set
End
```

Fig. 6*. Pseudocode describing simple operation of the ANN.*

For an ANN to train from data, the data needs to be stored in an easily accessible form. Utilising DeepLearning4J's ability to read from comma-delimited files (using file extension .csv), all data for each sample was compressed into a single line of comma-separated values. Optimal performance requires for all data to be scaled between 0 and 1, although -1 to +1 is also recommended [8-10]. DeepLearning4J provides this scaling service. To provide a consistent analysis (due to the potential for different minimum and maximum values between datasets) the data was scaled using fixed boundaries prior to storing the results in the .csv file. If output values were outside of the acceptable range then the entire line was omitted to prevent errors. While scaling to the absolute maximum and minimum values for all outputs would have presented an easier analysis (as the scale would be the same for all outputs), this would have resulted in outputs clustered around small sections of their range and give a false impression of accuracy. In terms of the hardware used, all ANNs were trained with combined CPU and GPU usage (Intel i5-7600K and NVIDIA GTX 1060-6GB respectively). "Early stopping" methods, where the ANN trains for a predefined amount of time whilst saving the best scoring model, were employed.

To prevent "overfitting" i.e. where the ANN simply learns answers rather than attempting to create a general solution, and to achieve the highest accuracy possible, several techniques documented in Benardos et al. [32] were employed. In general:

1. Start with one layer of 6 nodes.
2. Add or remove nodes until there is a decrease in accuracy.
3. Sample activation functions to find the best accuracy.
4. Add an extra layer and repeat.

The process is repeated until no further accuracy is gained by the ANN. The learning rate is decayed over time to allow for smaller corrections as the ANN reaches full accuracy, the decay rate being dependent upon the size of the dataset. See Appendix H for learning rate decay parameters.

6 Modelling Results

It has been shown by Fernandez et al. [26] that for application to CFD problems, ANNs can be an accurate prediction tool for a highly limited or simplified input and output. ***Table 4*** illustrates the performance of the ANN approach over a sample of the Cases described, with strong correlation shown by the values of coefficient of determination.

	Maximum Possible Accuracy	**20,000 Data**		**2,000 Data**	
		Case I MAE	**Case I R^2**	**Case II MAE**	**Case II R^2**
C_L	6e-5	9.8e-3	0.850	3.8e-3	0.961
C_d	1e-4	1.8e-3	0.781	1.4e-3	0.913
C_m	3e-5	8.2e-3	0.978	7.7e-3	0.978
$C_{L_{\dot{\alpha}}}$	5e-4	9.8e-3	0.961	8.3e-3	0.963
$C_{m_{\dot{\alpha}}}$	5e-5	5.4e-3	0.965	5.3e-3	0.954
C_{l_p}	5e-3	9.9e-3	0.916	5.3e-3	0.972
C_{y_p}	6e-4	1.5e-3	0.792	3.5e-4	0.985
C_{n_p}	1.4e-3	7.3e-4	0.822	5.4e-4	0.966
C_{L_q}	1e-4	9.6e-3	0.981	8.3e-3	0.974
C_{m_q}	3e-5	9.5e-3	0.974	8.2e-3	0.978

Table 4 *- Comparison of maximum possible accuracy limited by granularity to accuracy achieved from learning for Case A.*

The maximum achievable accuracy is caused by the "granularity" of the output, as the comma-delimited output file from Datcom+ Pro is limited to 4 decimal places of precision. As a result, there is a limit to how precise the predictions of the ANN can be, as the output from an ANN is usually to the precision of 15 decimal places (known as double precision). ANNs learn from penalising incorrect answers, and therefore a prediction which is accurate to four decimal places but not five (i.e. everything after that is truncated) will be penalised for being incorrect. For outputs such as the yaw moment derivative w.r.t. yaw velocity, this "granularity" can provide a hard limit to the maximum accuracy of the ANN as the range

of values for a 737-like aircraft are usually of the magnitude of 10^{-2} and close to zero. In comparison, the coefficient of lift typically varies from -0.3 up to $+1.7$ and hence the granularity of the scale is much finer. The limit imposed by the granularity can prevent the ANN from correctly interpolating past four decimal places as it will be penalised for doing so. See Appendix K for an illustration of how granularity occurs. ***Table 5*** provides an example of how the ANN performed at predicting the output for the stock model, although the stock model is within the bounds of the stochastic dataset.

	True Value ± MAE 20k Case II	**Predicted Value**
C_L	0.5577 ±0.0057	0.55625
C_d	0.0291 ±0.0012	0.02888
C_m	0.0235 ±0.0025	0.02274
$C_{L_{\dot{\alpha}}}$	0.0787 ±0.0017	0.07892
$C_{m_{\dot{\alpha}}}$	-0.3331 ±0.0095	-0.33530
C_{l_p}	-0.0081 ±0.0001	-0.00811
C_{y_p}	0.0016 ±0.00005	0.00160
C_{n_p}	-0.0010 ±0.00004	-0.00104
C_{L_q}	0.2497 ±0.0074	0.25155
C_{m_q}	-0.7515 ±0.0264	-0.7384

Table 5 *- Comparison between range and predicted output for the stock 737-300 model at an angle of attack of 4° for Case A.*

Predicting the stability characteristics of an aircraft outside of the stochastic range will result in low-confidence predictions due to input values being outside of the range 0 to 1. In contrast to typical research on ANNs, the MAEs have significance in assessing the ANNs in this paper as they can be compared to the maximum achievable accuracy for each output [8].

6.1.1 Input and Output Cases

For input cases A and B, the performance was observed to be relatively unaffected by input format, although a larger input size results in a longer time to reach fully trained. Increasing the number of inputs by a factor of 10 requires approximately 50% more time to train. ***Fig.7*** illustrates a comparison of training progress. This trend follows through to the much larger 2D input for case C.

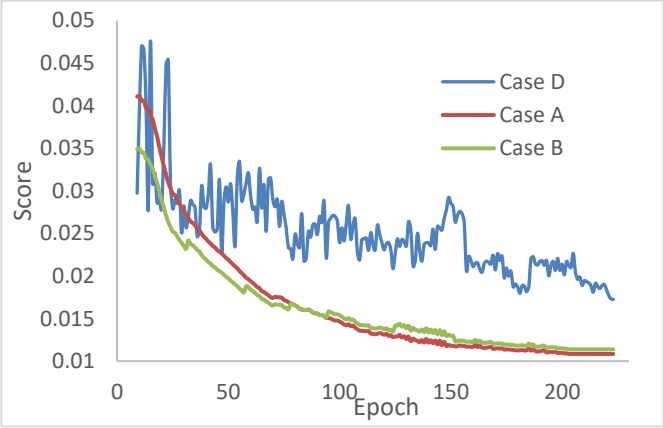

Fig.7 *- Illustration of best scoring model at each epoch with 200 training samples for Cases A, B and D.*

A larger input provides a better initial accuracy, although the percentage accuracy gained per epoch afterwards is reduced in comparison to the initial epoch. This phenomenon appears to be related to the low ratio of initial learning rate to input size. No 2D case was fully trained due to the much slower training progress with the 2D input. For comparison, it took 10 hours to reach 250 epochs with Case C, versus 50 and 180 seconds for Cases A and B respectively. The comparison of score versus epoch in *Fig.7* indicates that the ANN is gaining accuracy during training at a rate lower than a simpler input, and there are greater fluctuations. This is to be expected as there are almost 1 million inputs for the 2D image versus 29 and 1135 for the equivalent variable-only and coordinates-only input case. Cases B and C show similarities such as the initial greater performance, and more large corrections than Case A.

The nature of choosing the best scoring model (the sum of MSE for all outputs) causes the outputs with bigger granularity to have a bigger effect on the score. Therefore, the best model appears to sacrifice accuracy in some of the other outputs to achieve the lowest possible score. It is highly likely that some of the other models deemed non-optimal may in fact have much better accuracy for the smaller granularity outputs. This issue can be resolved by having a consistent range for all outputs, which should be the case for any application of the ANN approach elsewhere. The difference in accuracy between Cases A and B in ***Fig.7*** is negligible for fully-trained models. The main difference between multiple and single outputs was that different ANN configurations were required to achieve maximum accuracy with single outputs. These differences are documented in Appendix J. Different ANN configurations are required as each derivative and coefficient are differently sized problems mathematically inside Datcom+ Pro. Errors for Case I are poor in comparison to Case II in ***Table 4*** except for the two largest-error derivatives because the lowest overall score is key. For a large uniform problem where the calculations for each output are the same, there would likely be no benefit in creating multiple ANNs to solve individual outputs versus one ANN to solve for all outputs. It is worth noting that the best ANN configuration for small training datasets are not always the best ANNs for larger training sets. For example, it was beneficial to increase the number of nodes in each layer as the dataset size increased for the coefficient of drag. These observations are illustrated and documented in Appendix J. This can be attributed to a more complex understanding of the data as more data is available, i.e. a more complex model in the form of an ANN is created.

6.1.2 Dataset Sizing

There were two notable features regarding the effect of the size of the dataset on the accuracy of the model. Firstly, increasing the size of the training dataset increases the accuracy of the ANN once fully trained. Secondly, the law of diminishing marginal returns provides a limit to the accuracy that can be gained by adding more data, which provides an indication to the maximum achievable accuracy of the ANN. For some outputs, this maximum accuracy requires smaller datasets compared to others. ***Fig.8*** illustrates the general trend of how training set size affects accuracy for a selection of the outputs. The coefficients of lift, drag, and moment experience increases in accuracy past 200,000 samples indicative of the added complexity in calculations, in comparison to derivatives related to the rate of change of angle of attack which trained more quickly.

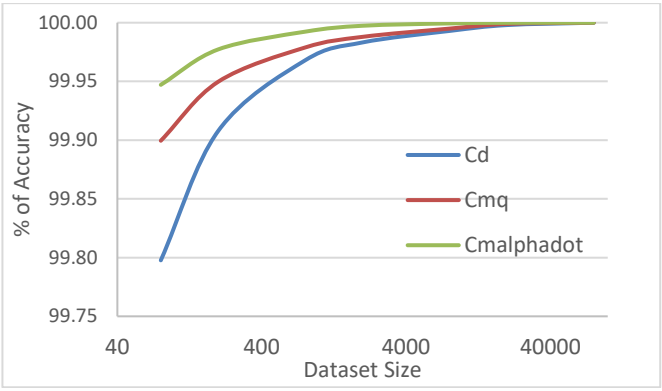

***Fig.8** - Different rates of accuracy gain where 100% corresponds to MAE with 100,000 training samples for Case A.*

While there is evidence in Zur et al. [33] that adding noise in the form of errors to the training data can help with training the ANN, errors in the validation data have a negative impact when saving the best performing ANN configuration. As a result, a sensitivity study is conducted to assess the impact of errors on best ANN configuration and is documented in Appendix D. It was noticed that the coefficient of lift displayed the highest sensitivity when comparing identically sized validation datasets, indicating either a relatively high percentage of inconsistent errors in the data produced by Datcom+ Pro, or the inability of the ANN to fully describe the problem. Since only excessive and obvious errors are removed prior to assembling the datasets, it is probable that subtle errors still make their way into the datasets. As a result, it is likely that errors contribute a larger portion of the inaccuracy compared to the actual ability of the system. This was observed for all input cases, including those without any variation of angle of attack or airfoil shape which were the most likely causes. In general, the biggest validation dataset possible provides the best general result possible, although the issue of granularity differences between individual outputs remains.

6.2 Optimal ANN Configuration

The optimal ANN configuration is found to be the same regardless of input size. This is due to the ANN solving roughly the same general problem regardless of input format. This observation holds for Case C despite the much larger size of the input. ***Fig.9*** illustrates the optimal ANN configuration for Case I.

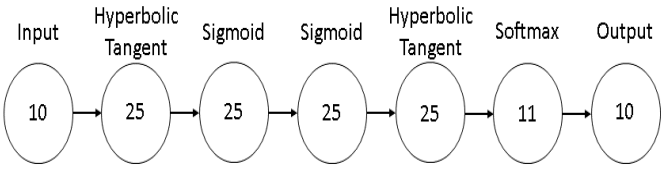

***Fig.9** - Optimal ANN configuration for Case I. Number of nodes is inside the circle, activation function is above circle.*

For reduced output Cases II and III with only one or two outputs, it was found that removing at least two of the hidden (between input and output) layers, and increasing the number of nodes in each layer (apart from the Softmax layer) was the most accurate general configuration. See Appendix J for a catalogue of optimal configurations for some of the other cases.

6.3 Combining Sections

Datcom+ Pro's ability to perform a wing- or body-only analysis makes it possible to investigate using an ANN to combine the sections. While this appears to be a relatively trivial task, equation (1) illustrates the interference effects between the fuselage and the wing. ***Fig.10*** shows how combining sections might work for a CFD problem.

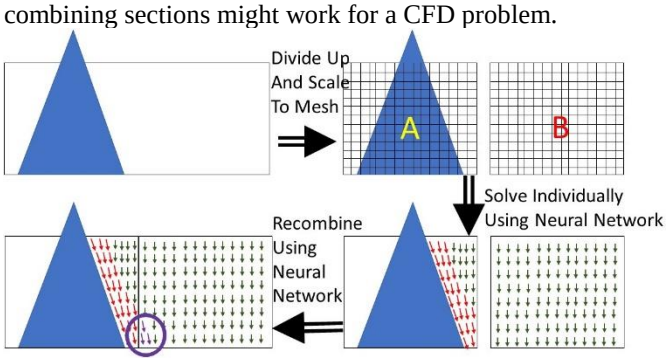

***Fig.10**. Flow-diagram for solving CFD problems with ANN using divide and reassemble technique.*

The top of ***Fig.10*** shows two small sections next to each other where the output is the direction of the flow at each node calculated on the input from the previous section plus the aircraft shape in that section. As section B has no reason to deviate, the flow will remain roughly parallel. However, the perpendicular portion of the flow for section A due to the aircraft shape will affect section B. By using an ANN to process this correction and therefore change the flows in both sections to the correct answer, the need to solve one large problem is removed. ***Table 6*** illustrates the outcome of assembling the wing- and body-only results to form a complete model with interference effects. The full aircraft geometry was also included in the input to the ANN to provide information for assessing the level of interference.

Coefficient	Maximum Accuracy	Combined Sections MAE	Variables Only MAE
C_L	6e-5	1e-4	3e-3
C_d	1e-4	3e-4	1e-3

***Table 6** - Comparison of results between section combination and variables-only, with reference to the maximum achievable values.*

It can be seen in ***Table 6*** that combining the aircraft sections resulted in greater accuracy than analysing the aircraft as a whole. This is due to the inclusion of the Datcom+ Pro results for the wing and body rather than making the problem easier to solve. In theory one ANN could handle all the small sections of a full aircraft CFD analysis, but it would likely be computationally more efficient to have an ANN for a chunk of nearby sections, with an ANN to combine all the chunks, like A and B in ***Figure 10***. This process could be refined, although the time to train each additional ANN would result in exponentially increasing computational costs. See Appendix E for additional discussion on combining sections.

6.4 Loop Learning

The ability to finish training with a dataset, save the best scoring model to date, reload the model, and then continue training with a different dataset is available with the DeepLearning4J library. This is known as "stop-start" learning, and is important for cases where collecting/creating data requires substantial time relative to the time to train. Both FEA and CFD could benefit as they take a relatively long time

to generate large datasets. ***Fig.11*** illustrates the loop created for this paper, whereby datasets were generated and trained on using ANNs periodically.

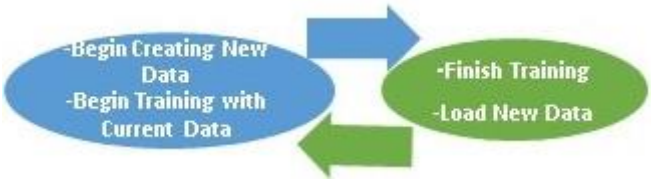

Fig. 11 *– Simplified illustration of the loop learning method.*

Appendix F documents the results in detail, although the results of loop learning were promising, with an average of 5-10% better accuracy for 15 smaller datasets compared to one larger dataset of equivalent size. The net effect of training on smaller datasets concurrently is inconclusive due to the small sample size of 5 runs, although the absence of a large deficit is promising for applications where generating a large dataset is the issue. The downside to training on small concurrent datasets is the additional time required compared to training on one large dataset.

7 Conclusions

The ANN approach showed good agreement with the expected output from Datcom+ Pro. Strong indications of the potential for using ANNs with more accurate aircraft dynamics analysis tools such as CFD are illustrated. The possibility of a single ANN handling a full aircraft CFD analysis would likely be computationally inefficient in comparison to breaking the analysis into small uniform sections and combining them. The ANN approach should be investigated with large CFD and FEA problems with emphasis placed on the potential time advantage of the ANN approach.

References

[1] Finck RD, Air Force Flight Dynamics Laboratory (US), Hoak DE. USAF stability and control DATCOM. Engineering Documents; 1978.

[2] Anderson JD, Wendt J. Computational fluid dynamics. New York: McGraw-Hill; 1995.

[3] Kollmann W. Computational fluid dynamics. Washington, DC, Hemisphere Publishing Corp., 616:A80-37231-A80-37239. 1980.

[4] Williams JE, Vukelich SR. The USAF Stability and Control Digital DATCOM. Volume I. Users manual. McDonnell Douglas Astronautics Co St Louis MO; 1979.

[5] Galbraith, B. Datcom+ Pro User's Manual 3.5 [online]. 2015

[6] Fletcher HS. Comparison of several methods for estimating low-speed stability derivatives for two airplane configurations. National Aeronautics and Space Administration; 1971.

[7] Shafer TC, Lynch CE, Viken SA, Favaregh N, Zeune C, Williams N, Dansie J. Comparison of computational approaches for rapid aerodynamic assessment of small UAVs. 2014.

[8] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015 May 28;521(7553):436-44.

[9] Bengio Y. Learning deep architectures for AI. Foundations and trends in Machine Learning. 2009 Nov 15;2(1):1-27.

[10] Deng L, Yu D. Deep learning: methods and applications. Foundations and Trends in Signal Processing. 2014 Jun 30;7(3–4):197-387.

[11] Mazurowski MA, Habas PA, Zurada JM, Lo JY, Baker JA, Tourassi GD. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. Neural networks. 2008 Apr 30;21(2):427-36.

[12] Maren AJ, Harston CT, Pap RM. Handbook of neural computing applications. Academic Press; 2014.

[13] Torbati N, Ayatollahi A, Kermani A. An efficient neural network based method for medical image segmentation. Computers in biology and medicine. 2014 Jan 1;44:76-87.

[14] Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, Dieleman S. Mastering the game of Go with deep neural networks and tree search. Nature. 2016 Jan;529(7587):484-9. 2016.

[15] Oh J, Guo X, Lee H, Lewis RL, Singh S. Action-conditional video prediction using deep networks in atari games. InAdvances in Neural Information Processing Systems.2863-2871. 2015.

[16] Oshri B, Khandwala N. Predicting moves in chess using convolutional neural networks. 2016.

[17] Eberhart RC, editor. Neural network PC tools: a practical guide. Academic Press; 2014.

[18] Jin Y. A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing-A Fusion of Foundations, Methodologies and Applications. 2005 Jan 1;9(1):3-12.

[19] Alizadehdakhel A, Rahimi M, Sanjari J, Alsairafi AA. CFD and artificial neural network modeling of two-phase flow pressure drop. International Communications in Heat and Mass Transfer. 2009 Oct 31;36(8):850-6.

[20] Bouzid AH, De Technologie Superieure E, Champliaud H. On the use of dual Kriging interpolation for the evaluation of the gasket stress distribution in bolted joints. InProceedings of the ASME pressure vessels and piping conference 1998 457:77-83.

[21] Stefanou G. The stochastic finite element method: past, present and future. Computer Methods in Applied Mechanics and Engineering. 2009 Feb 15;198(9):1031-51.

[22] Stein ML. Interpolation of spatial data: some theory for kriging. Springer Science & Business Media; 2012 Dec 6.

[23] Lucifredi A, Mazzieri C, Rossi M. Application of multiregressive linear models, dynamic kriging models and neural network models to predictive maintenance of hydroelectric power systems. Mechanical systems and signal processing. 2000 May 1;14(3):471-94.

[24] Demyanov V, Kanevski M, Chernov S, Savelieva E, Timonin V. Neural network residual kriging application for climatic data. Journal of Geographic Information and Decision Analysis. 1998;2(2):215-32.

[25] El-Beltagy MA, Nair PB, Keane AJ. Metamodeling techniques for evolutionary optimization of computationally expensive problems: Promises and limitations. InProceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1 1999 Jul 13.196-203. Morgan Kaufmann

[26] Fernández J, Pernía A, Martínez-de-Pisón FJ, Lostado R. Prediction models for calculating bolted connections using data mining techniques and the finite element method. Engineering Structures. 2010 Oct 31;32(10):3018-27.

[27] Fiesler E, Beale R. Handbook of neural computation. Oxford University Press; 1996 Sep 1.

[28] Craven M, Shavlik JW. Extracting tree-structured representations of trained networks. InAdvances in neural information processing systems 1996:24-30.

[29] Larsson G, Maire M, Shakhnarovich G. Fractalnet: Ultra-deep neural networks without residuals. arXiv preprint arXiv:1605.07648. 2016 May 24.

[30] DeepLearning4J: Open-Source Distributed Deep Learning for the JVM [online]. 2012.

[31] Pamadi, BN. Performance, Stability, Dynamics and Control of Airplanes, AIAA. 2004.

[32] Benardos PG, Vosniakos GC. Optimizing feedforward artificial neural network architecture. Engineering Applications of Artificial Intelligence. 2007 Apr 30;20(3):365-82.

[33] Zur RM, Jiang Y, Pesce LL, Drukker K. Noise injection for training artificial neural networks: A comparison with weight decay and early stopping. Medical physics. 2009 Oct 1;36(10):4810-8.

Appendices

Appendix A – Activation Functions

An activation function describes the output based on the input. ***Figure 1*** illustrates three of the activation functions used in this paper. All input nodes used rectified linear units as default.

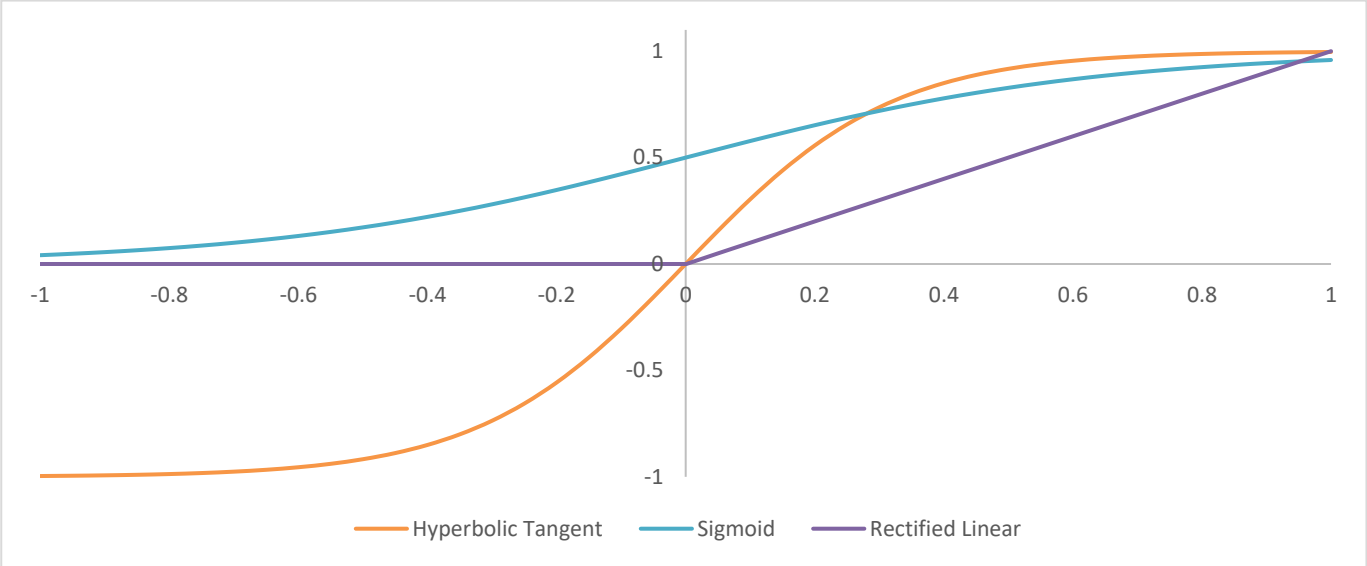

Figure 1 *- An example of three of the most common activation functions.*

Hyperbolic tangent and sigmoid nodes often give similar results based on their similar shape, however hyperbolic tangent nodes are usually preferred as they tend to not "squash" a large range of the outputs as much as sigmoid nodes. Rectified linear units are useful for when a certain threshold must be achieved before an output is sent from the node. This could be useful for switching from a subsonic to supersonic analysis with Datcom+ Pro. The outputs and inputs for nodes do not have to lie between -1 and +1, although for best results it is recommended to scale the full ANN input and output between 0 and 1. ***Figure 2*** illustrates some less commonly used activation functions.

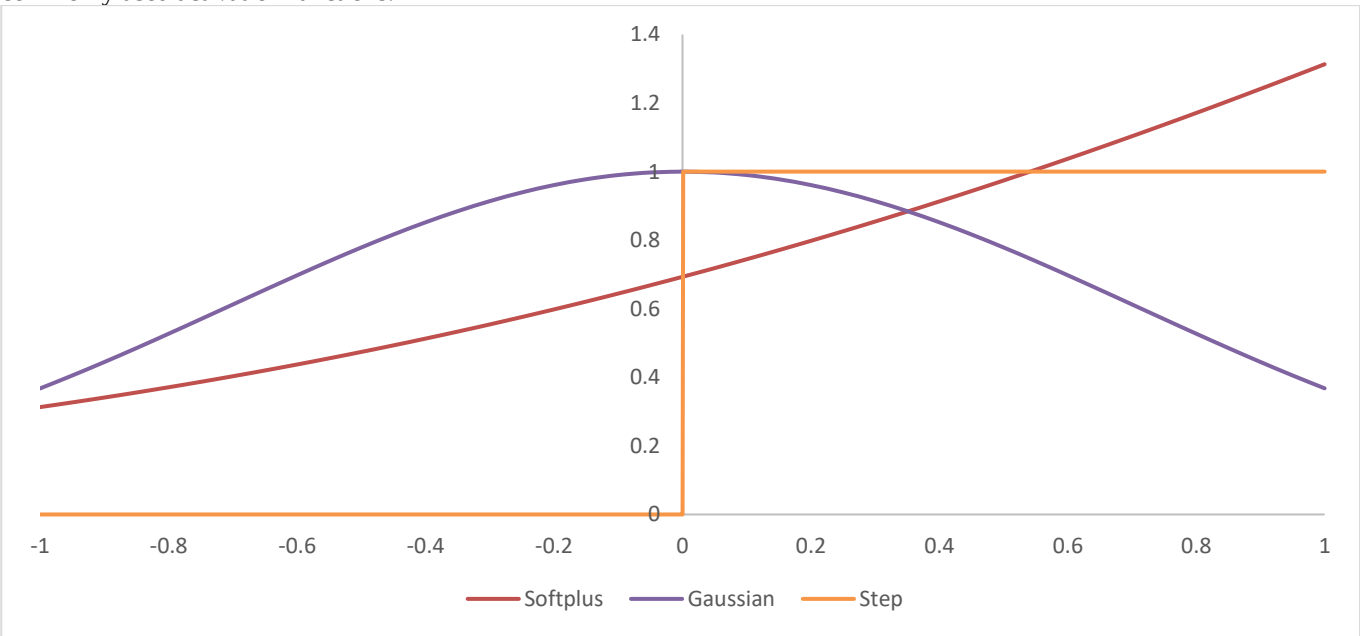

Figure 2 *- Less common activation functions*

However not all activation functions involve the single determining factor of the input from the previous layers. For example, the Softmax function used in this paper is defined as:

$$f_i\left(\underset{x}{\rightarrow}\right) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \text{ for } i = 1, \ldots, J$$

Where J refers to the number of outputs, and x_i and x_j refer to the original and transformed values respectively. The transformed values are derived from a transformation on the sum of the input to make it equal to one. All transformed values are in the range $\{0,1\}$. These functions are useful for tasks such as reducing the effect of errors in the dataset represented by extreme outliers.

Appendix B – Accuracy vs Input Case Results

Figure 3 shows the comparison of Case A relative to Case B in terms of performance, where Case A is used as the performance baseline.

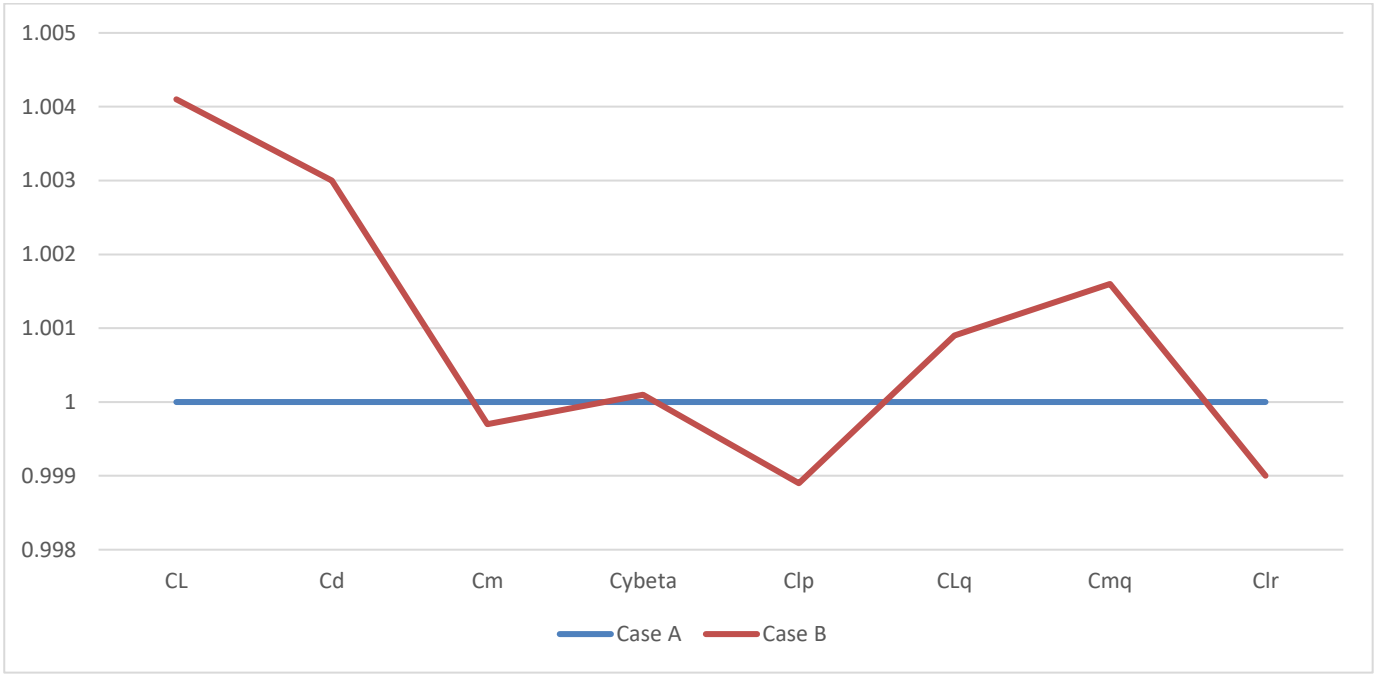

Figure 3 *– Comparison of Case A with Case B for a selection of derivatives and coefficients with planform-only inputs.*

The difference is negligible, with a 0.4% discrepancy more likely to be due to differences in the validation dataset. ***Figure 4*** shows the same results, but instead for when the flight conditions and airfoils are also included in the stochastic model.

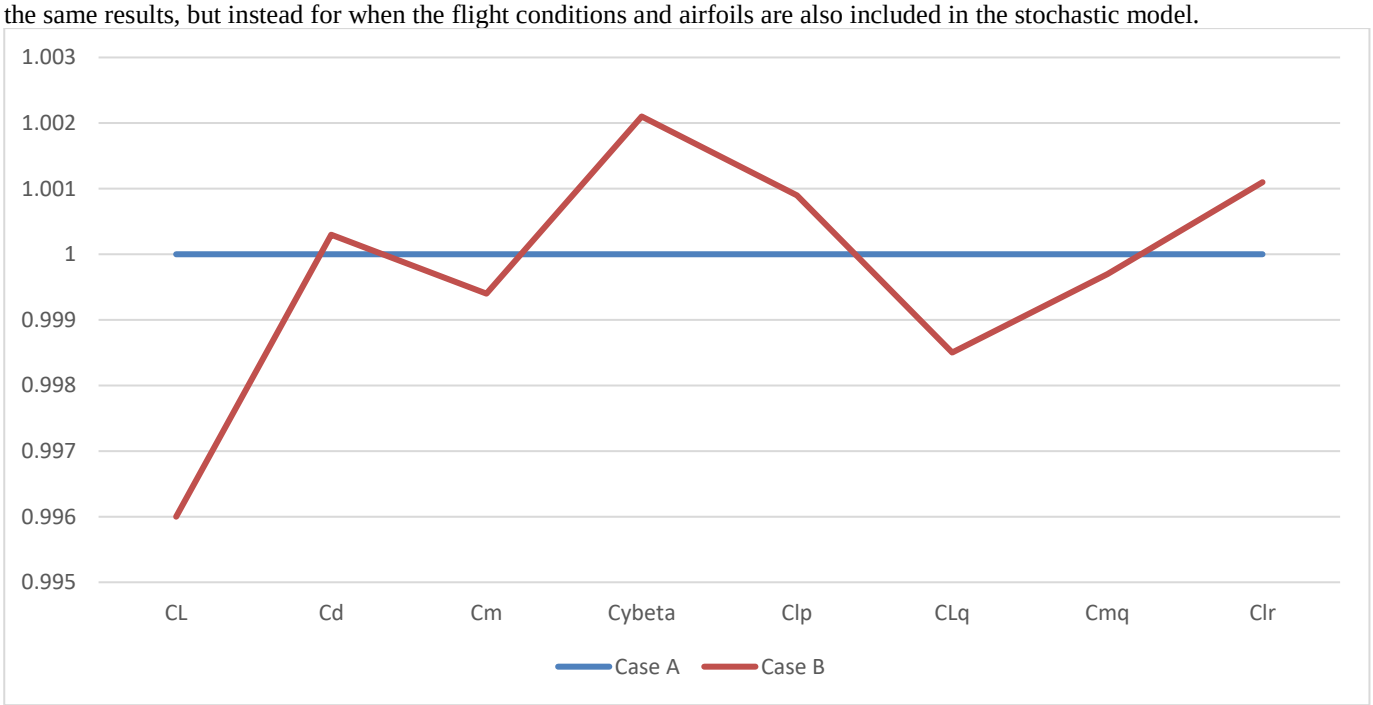

Figure 4 *– Comparison of Case A with Case B for a selection of derivatives and coefficients for all inputs.*

Again, the difference between the two input types is negligible. As no 2D input case was completed due to excessive time requirements for a relatively tiny setup in comparison to current supercomputing centres, no fully trained results can be analysed in comparison. However, as seen in the main body of this paper, the general trend is highly indicative of the ANN being trained similar but slower than Case A or Case B. The large oscillations in the best scoring model for each epoch in ***Fig.7*** in the main section are reflective of the large number of weightings that are being adjusted.

Figure 5 illustrates the same graph, but with only the best model to-date at each epoch for Case C to show the true progress.

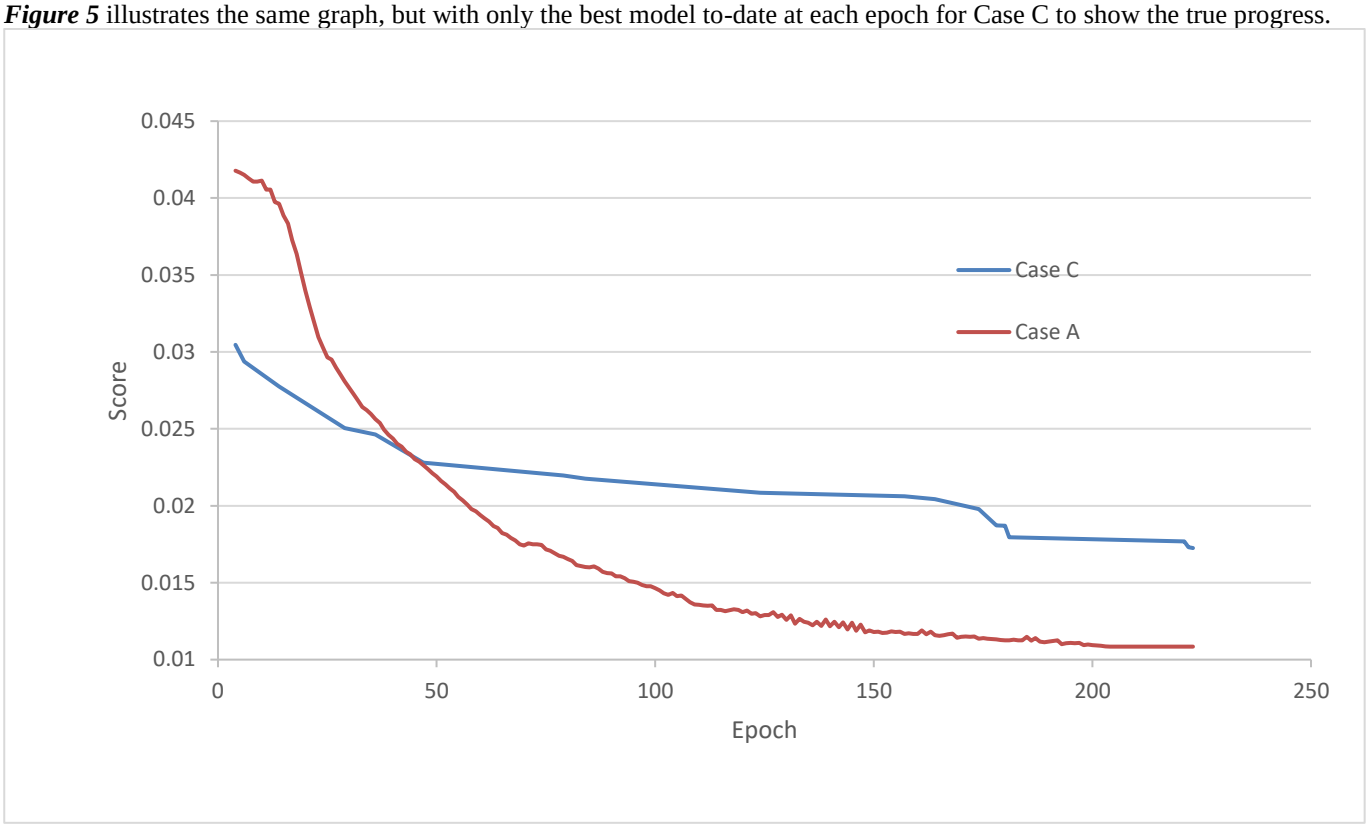

Figure 5 – *Comparison of the learning progress of Case C with Case B.*

Case C shows similar progress to the other cases when only the best model to-date rather than at each epoch is considered. The greater initial accuracy for larger inputs as described in the main section can be seen more clearly here, along with the slower rate of progress after the initial phase of learning.

Appendix C – Accuracy vs Output Case Results

Figure 6 illustrates the effect of different output cases. For Case III, the coefficients were paired with the largest-error derivative C_{L_q} for a dataset size of 2000.

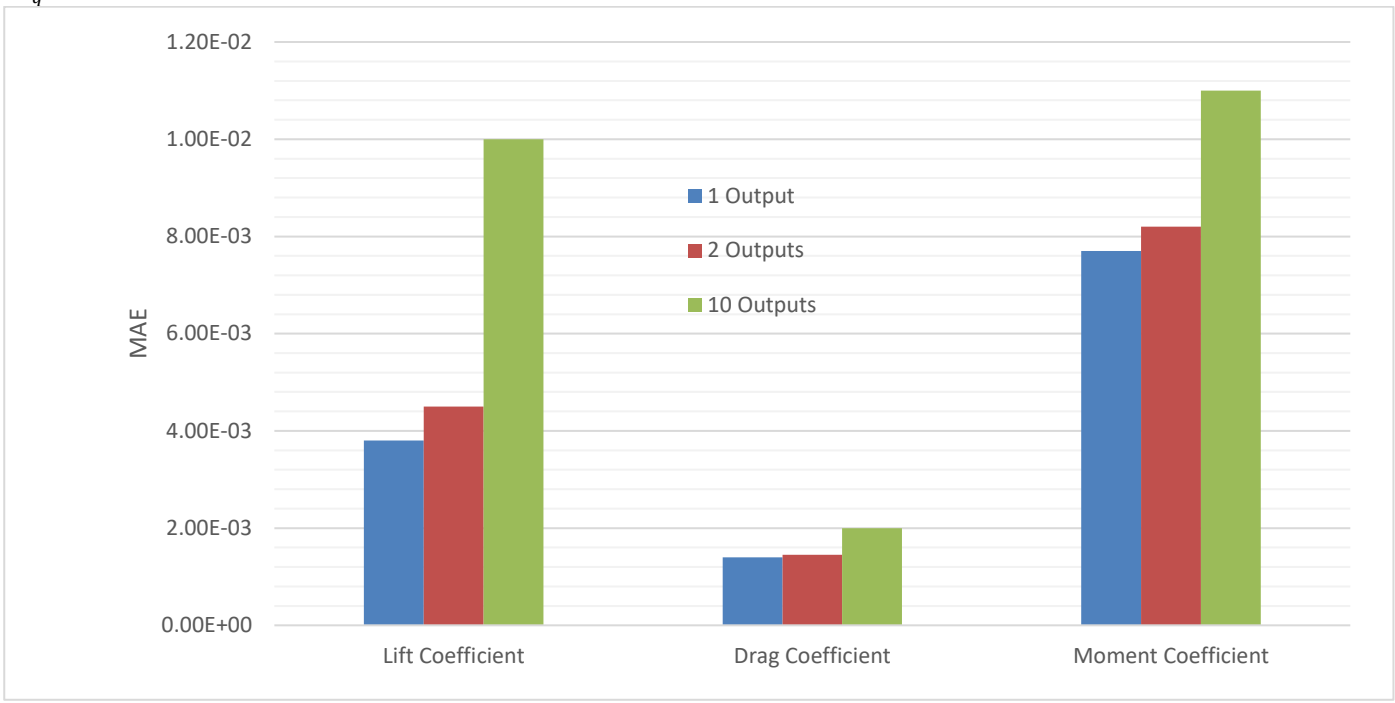

Figure 6 *– Comparison of Cases I, II, and II in terms of MAE for a selection of coefficients.*

It must be mentioned that at larger dataset sizes, the difference becomes more pronounced, as illustrated in ***Figure 7*** where the results for the coefficient of drag at a dataset size of 200,000 are shown, and being paired with the coefficient of lift for the two-output case.

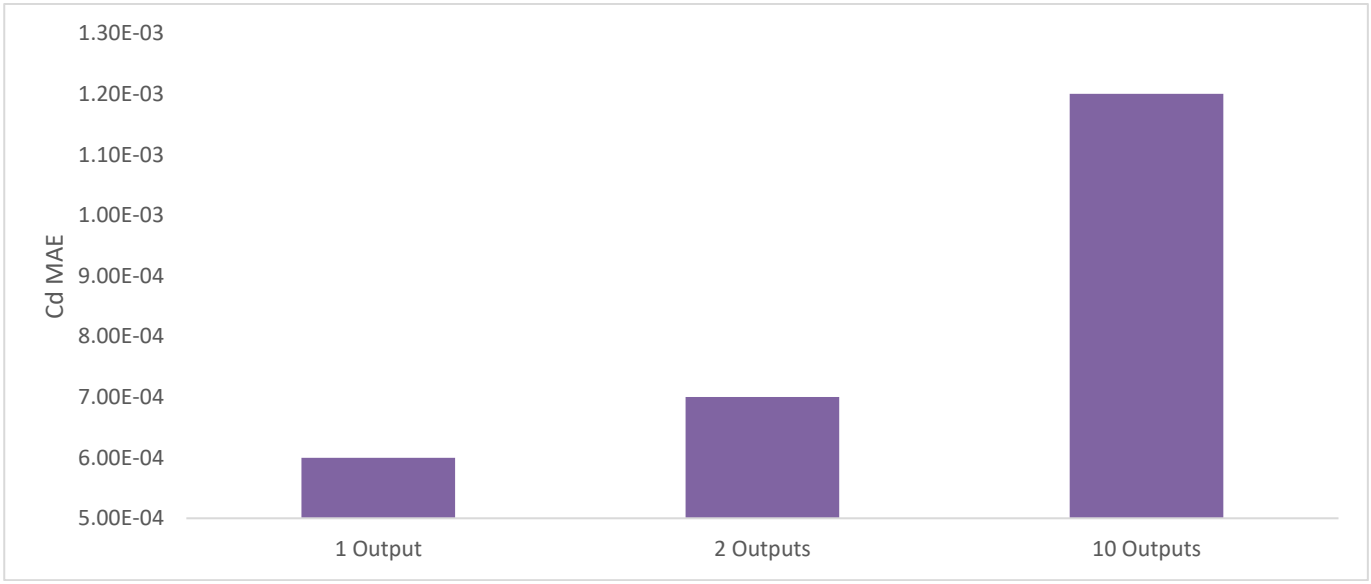

Figure 7 *– Comparison of Cases I, II, and II in terms of MAE for the coefficient of drag.*

Case I with 10 outputs can be clearly seen to be limiting the accuracy for the coefficient of drag, and even when paired with just the coefficient of lift, the accuracy is limited. However, the main reason for the limitation is likely to be the difference in granularity and maximum accuracy for each output rather than a total inability of the ANN to solve for the problem due to the nature of the scoring mechanism.

Appendix D – Accuracy vs Dataset Size Results

An illustration of the effect of training dataset sizing on the error in the coefficient of lift is show in ***Figure 8*** for Case A input and Case II output.

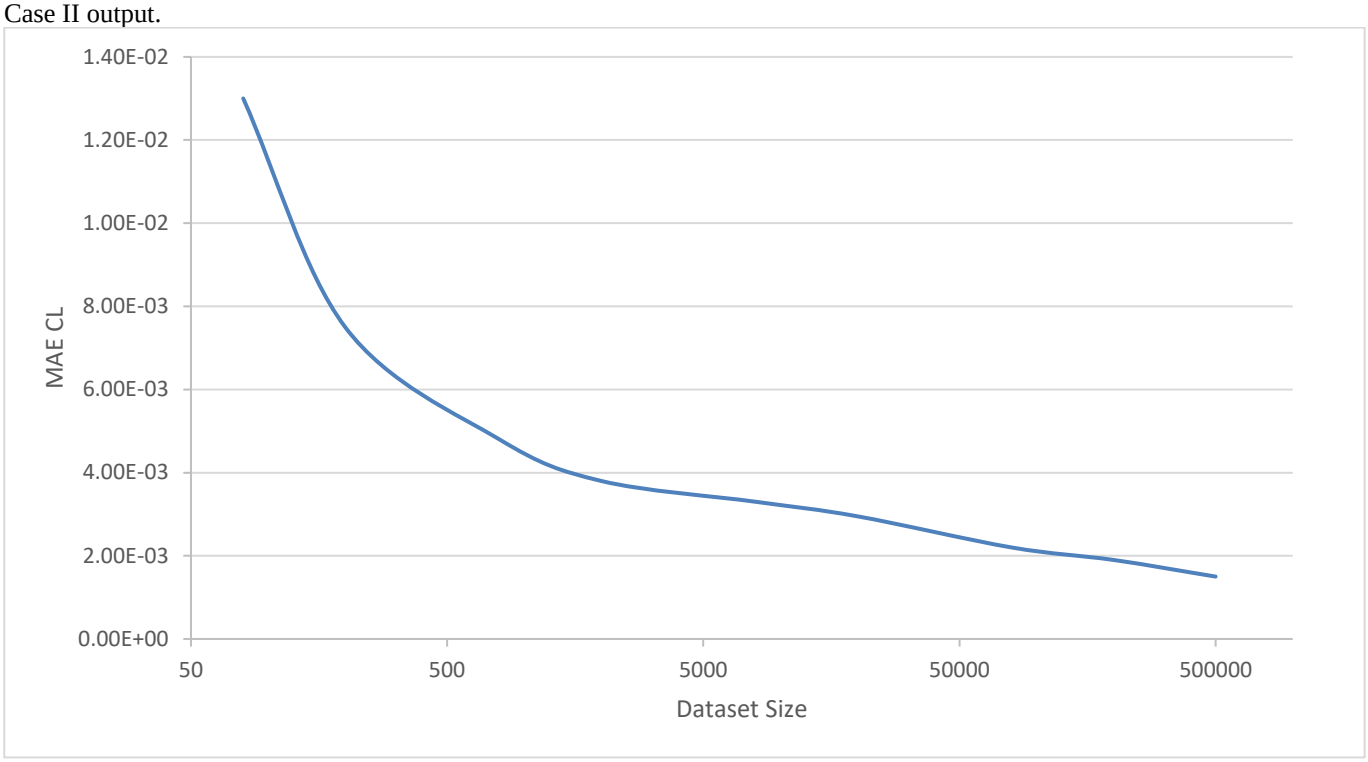

Figure 8 *– Plot of MAE for the coefficient of lift versus dataset size.*

The ANN continues to make progress, and is likely to continue making progress past 500,000 samples. The time taken to fully train must also be considered, as shown in ***Figure 9*** for the same input and output case.

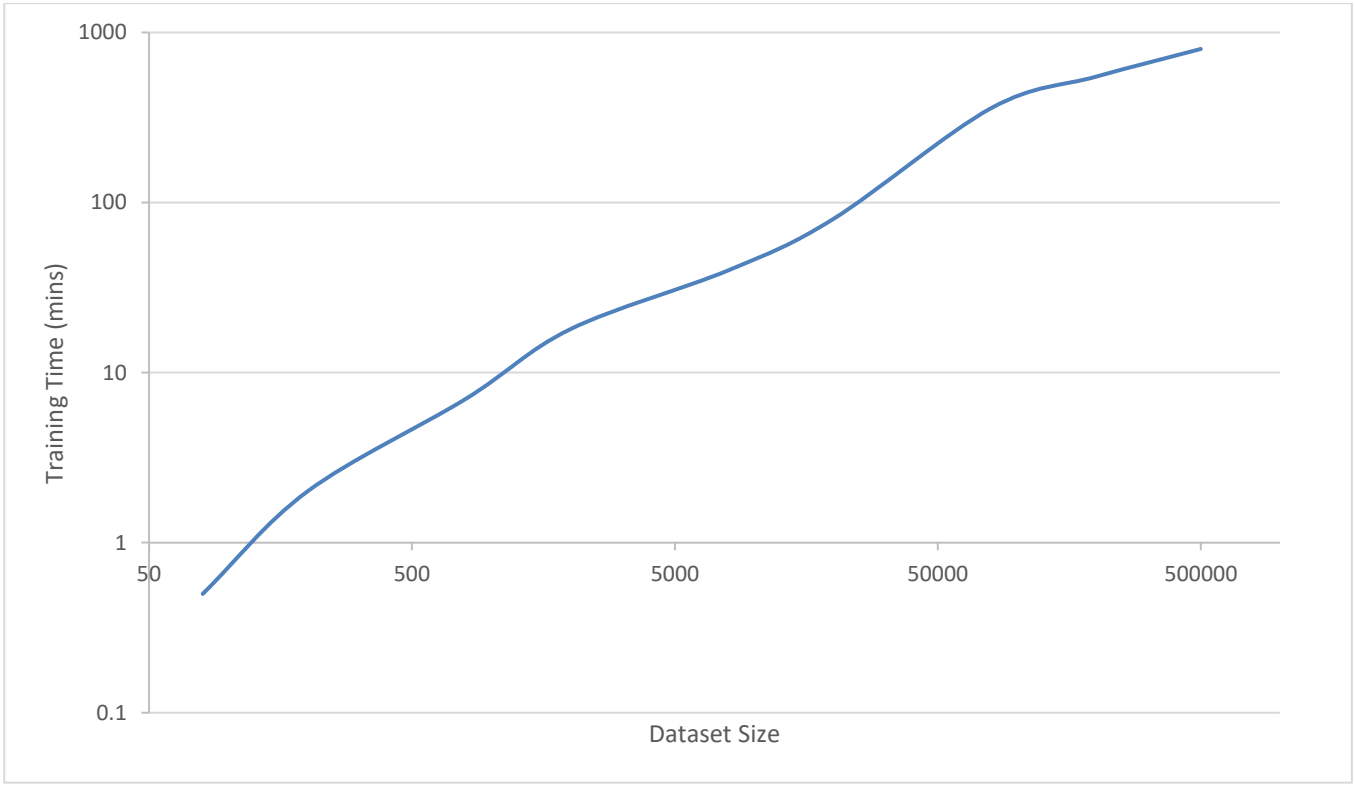

Figure 9 *– Plot of required training time for Case A versus dataset size.*

A roughly linear increase can be seen when viewing with the log-log graph in ***Figure 9***.

There were several surprising trends noticed, including a reduction in the coefficient of determination for larger datasets, in conjunction with the greater accuracy. It was also noticed that the R^2 value could often decrease despite a decrease in MAE for different validation datasets. In certain cases, the R^2 value would turn highly negative despite the MAE being much lower than cases where the R^2 value is very close to the perfect correlation value of 1. ***Figure 10*** documents this phenomenon for the coefficient of lift.

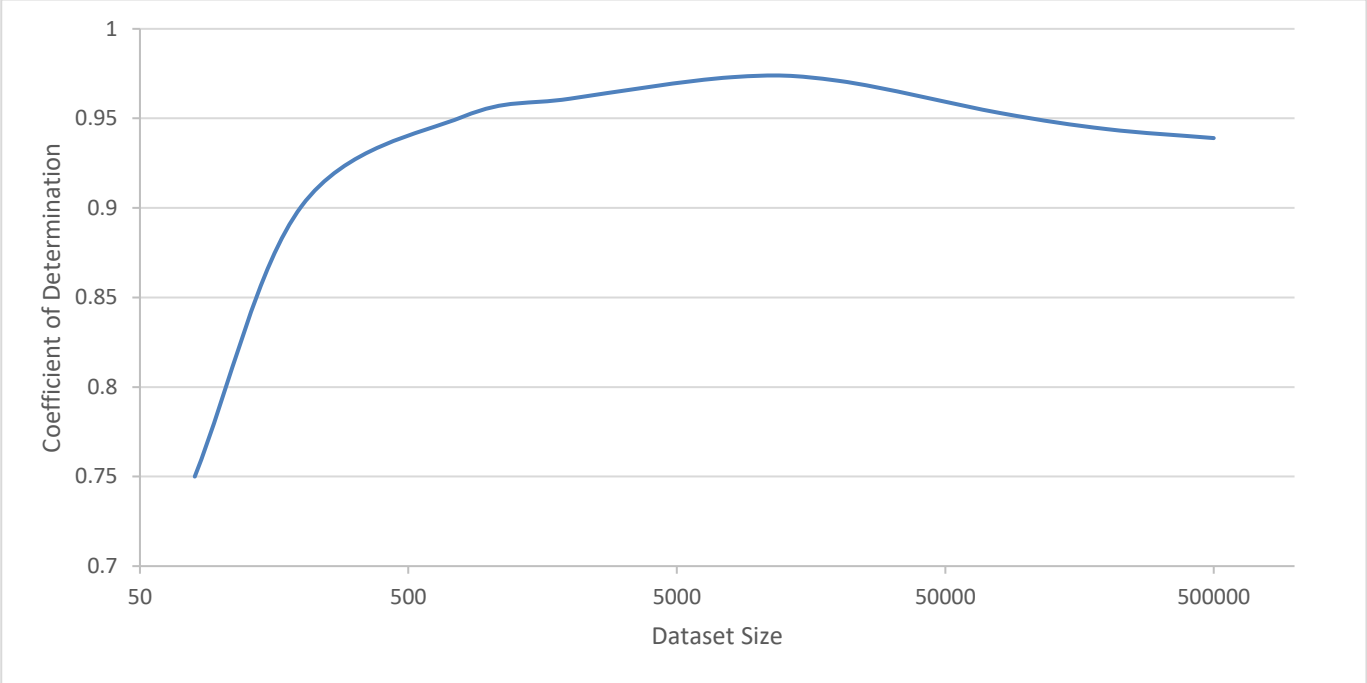

Figure 10 *– Plot of the coefficient determination for the coefficient of lift versus dataset size.*

It appears to occur most often when the MAE is smaller than the corresponding maximum accuracy. This probably indicates that the ANN is compensating for the granularity and predicting values to a degree of accuracy past the limit imposed by the granularity. Regarding the validation dataset sizing, ***Figure 11*** illustrates how smaller validation datasets provide different results to larger datasets, hence why standardised validation set sizes were used, as described in Appendix J.

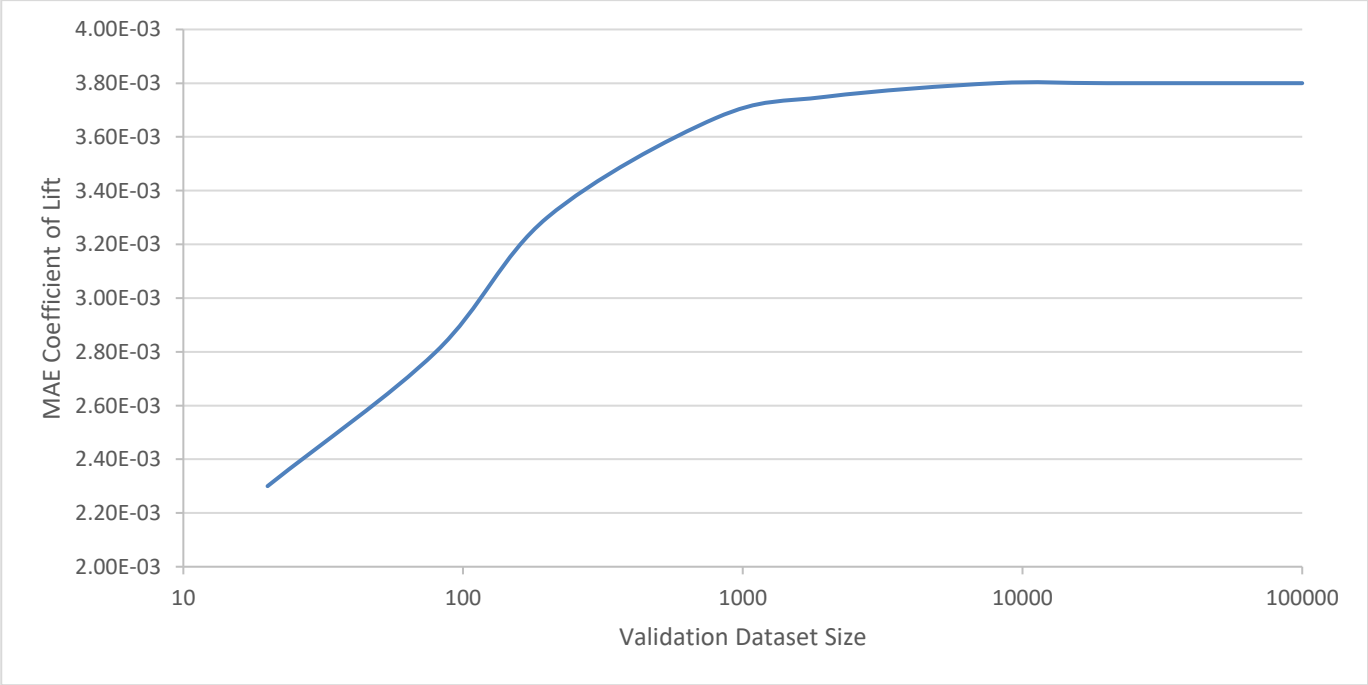

Figure 11 *– Plot of MAE for the coefficient of lift versus validation dataset size.*

To understand discrepancies at smaller validation dataset sizes, several runs of validation data are produced for each dataset size, and their accuracy compared. ***Figure 12*** shows the results of this analysis. For the larger validation datasets little variance was seen, as the number of errors would stay at roughly the same percentage of the dataset. However, for validation datasets with 50 or less samples, large discrepancies could be seen between different batches due to sensitivity to errors. This also gives an indication of the approximate percentage of errors.

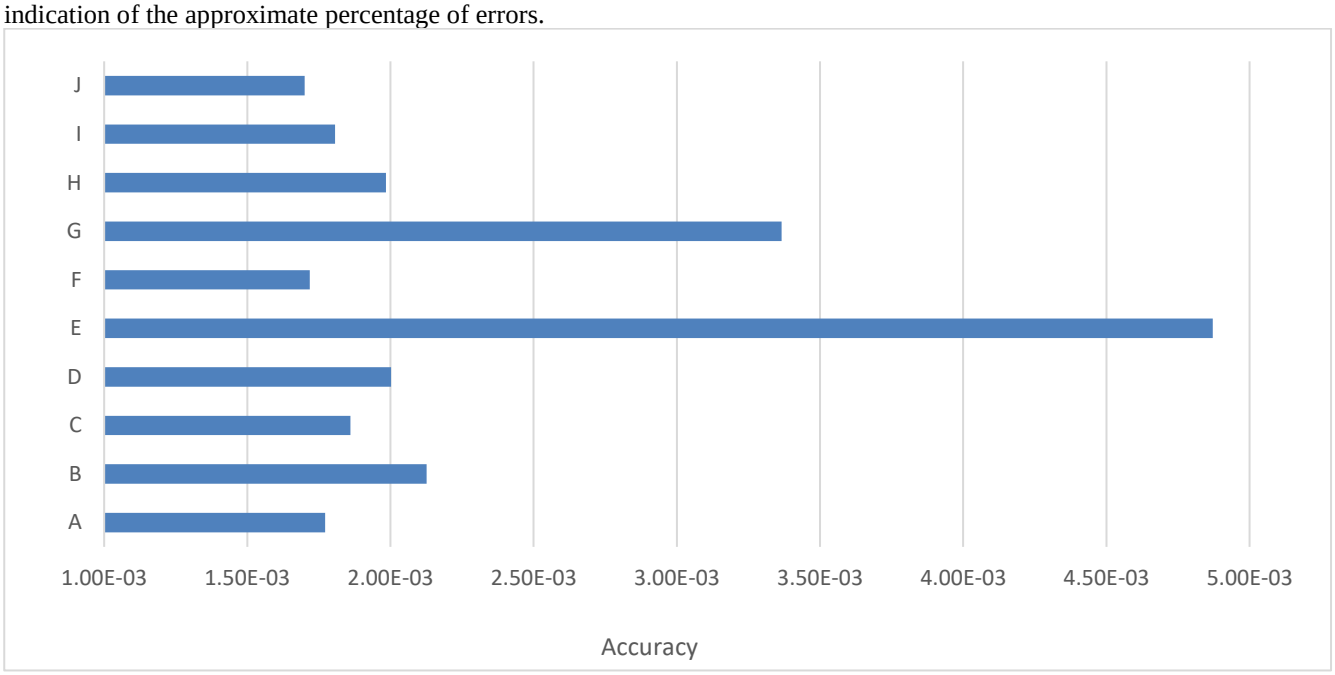

Figure 12 – *Varying levels of error in predicting the coefficient of lift measured between 10 random validation datasets labelled A to J with 20 samples each.*

It must also be noted that with larger validation datasets, the potential to "luck into" a low score that suits the validation data but not necessarily the general solution is less likely. One final point to consider regarding the 999x999 pixel images, is that storing the data as a .csv file results in 4MB per combination. When compressed or stored as a native .jpg image, this reduces to about 200kB which is much easier to transfer. In this same way, if a large CFD or FEA problem is used with an ANN, it would be beneficial to translate or store the data in image form until required in .csv form.

Appendix E – Combining Sections

The theoretical method of utilising section combination in an FEA or CFD analysis is documented here. The overall tree diagram for how combining sections would work is shown in ***Figure 13***.

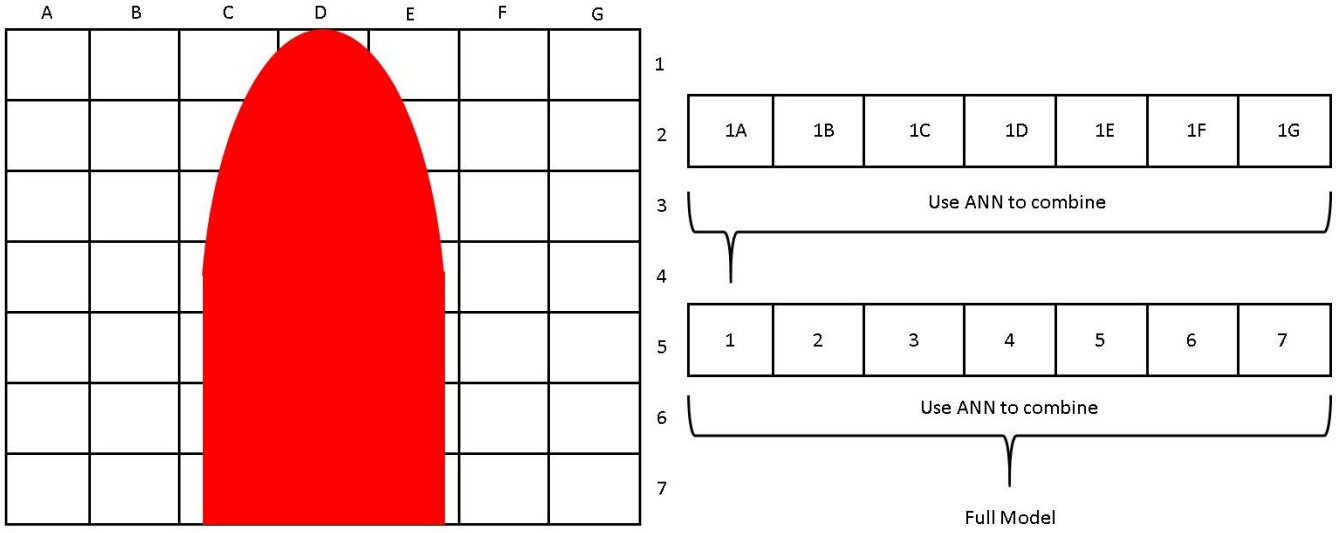

Figure 13 *– Illustration of object on the grid of units and chunks (left), and the structure of the solution (right)..*

Each "unit" (i.e. 1A, 1B, etc.) would have a fixed size mesh, where both the number of nodes and position of nodes is constant. Each unit would require a general unit solver specifically for the size of the standardised unit. The numbers one to seven would correspond to the "modules" which are a collection of units. The modules could be used for a fixed number of units (e.g. 10 units), although different sized modules could exist. From there the model can be assembled into larger "chunks", which are a collection of modules, and there should be sufficient chunks assembled to fully cover the problem area. 2D illustrations are used although there is no reason why 3D problems cannot be solved using the same method. However, getting data from a random MOO mesh into the uniform unit format would require interpolation of the mesh. ***Figure 14*** illustrates how mesh interpolation would work for the uniform mesh of an OOM approach.

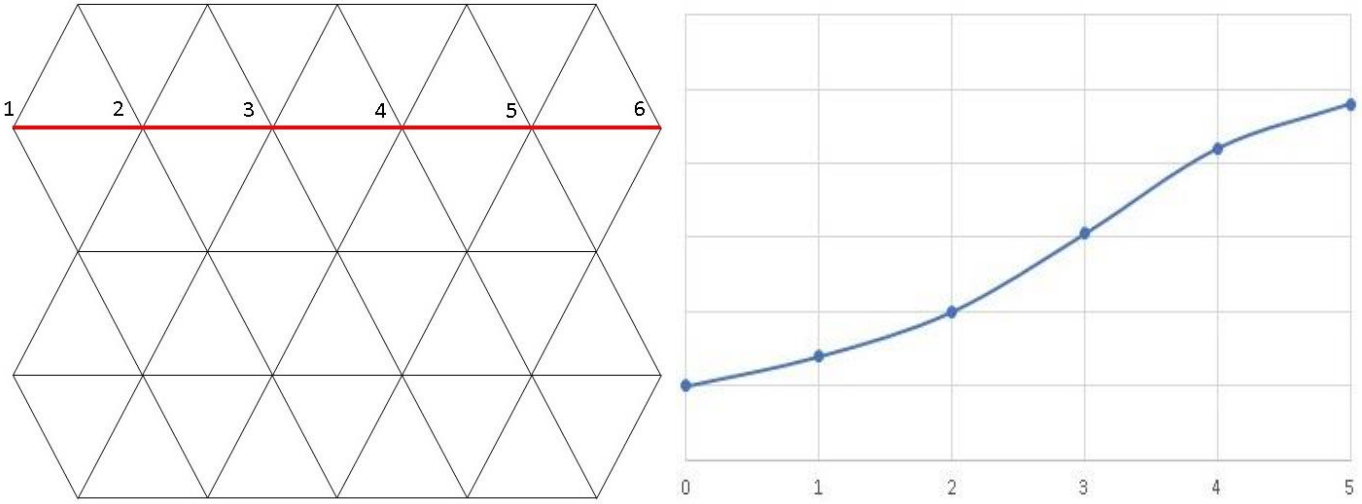

Figure 14 *– Illustration of mesh interpolation technique whereby any unit nodes that cross in between MOO nodes labelled 1-5 (left) will be interpolated from the graph (right).*

The red line corresponds to a path across the mesh, with the corresponding values of interest (i.e. displacement, pressure, stress, etc.) along the path plotted. As the OOM nodes will not line up with the MOO nodes, the values will have to be interpolated between the nodes. However, this allows for an ANN to train with data from CFD or FEA problems with coarse meshes first (allowing quicker creation of less accurate data), before swapping to more refined meshes later when the accuracy increase per dataset begins to level out.

Appendix F – Looped Learning Results

Figure 15 illustrates the results of loop learning, where training on several smaller datasets is compared to training on an equivalently sized single dataset. The coefficient plotted is the coefficient of lift.

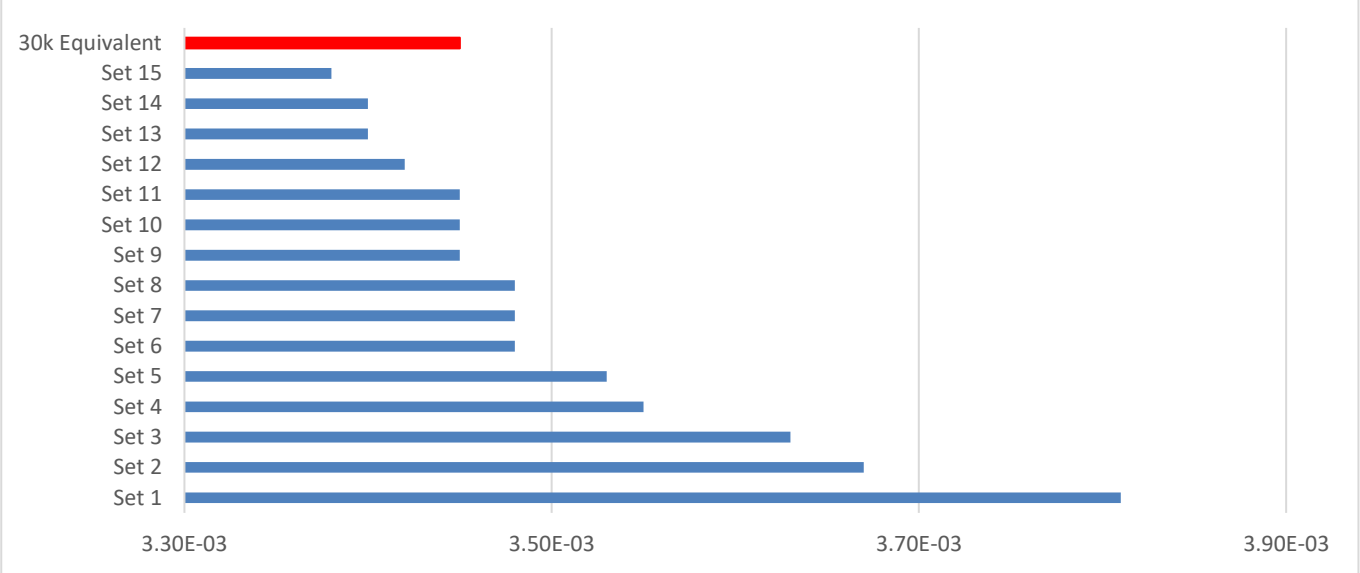

Figure 15 *– Performance of stop-start learning for 15 small 2000-sample datasets versus the equivalent large 30000-sample dataset..*

The average accuracy after each 2k dataset is shown versus a single large 30k dataset. The average was taken from 5 different runs for both the smaller and large datasets. However, for such a small sample size it is difficult to draw a strong conclusion regarding loop learning.

Appendix G – List of Variables

The body is scaled between 75% and 150% in terms of both cross-sectional area and length. To prevent errors due to highly unusual shapes, and to keep track of the exposed wing and tail span, the entire body is scaled rather than individual sections. The position of the tail sections is moved based on the body length. The range for the remaining variables are documented in ***Table A1***. The names are as per DATCOM variable names with the prefixes W, H or V for wing, horizontal stabiliser, and vertical stabiliser respectively. All units are the default DATCOM imperial input. Checks are put in place to prevent errors such as the outer wing panel span being bigger than the entire span.

Name	Min	Max	Name	Min	Max
MACH	0.25	0.7	ZCG	-5	5
ALT	0	30000	XW	30	40
ALSCHD	-4	18	ZW	-5	2
XCG	45	55	ALIW	0	5
ZH	5.5	7	H-CHRDR	9.5	38
W-CHRDR	11.65	46.6	H-CHRDTP	2.4	9.9
W-CHRDTP	2.65	10.62	H-SSPN	10.7	42.8
W-CHRDBP	6.42	25.7	H-SSPNE	8.8	40.9
W-SSPN	23.7	94.8	H-SAVSI	0	45
W-SSPNOP	15.6	62.4	H-DHDADI	0	10
W-SSPNE	17.4	88.6	H-TWISTA	-2	2
W-TWISTA	-4	2	V-CHRDR	5.95	23.8
W-SAVSI	0	45	V-CHRDTP	1.96	7.84
W-SAVSO	0	60	V-SSPN	10.4	41.6
W-DHDADI	0	5	V-SSPNE	9.1	40.3
W-DHDADO	-5	10	V-SAVSI	0	60

Table A1 *– Range for DATCOM variables in the default DATCOM input units.*

The remaining 96 of 135 variables are related to the body and are too numerous to list.

Appendix H – Learning Rate Decay

To allow the ANN to train to its maximum potential, the learning rate must be forced to decay. The rate at which the learning rate decayed was relative to the dataset size in this paper, illustrated in ***Table A1***. A batch size of 5 was used for all ANN configurations as it was found to be ideal for all situations. The training times are with a single Intel i5-7600K and NVIDIA GTX 1060-6GB.

Dataset Size	Decay Factor	Decay Step	Total Training Time
80	0.9	Every 100 batches	0.5mins
200	0.9	Every 1000 batches	2mins
800	0.925	Every 4000 batches	7mins
2000	0.95	Every 16000 batches	18mins
8000	0.975	Every 24000 batches	40mins
20000	0.99	Every 48000 batches	80mins
80000	0.995	Every 64000 batches	360mins
200000	0.9975	Every 96000 batches	550mins

***Table A2** – Learning rate decay parameters for different dataset sizes.*

The effect of decaying the learning rate versus choosing a lower initial rate with no decay is shown in ***Figure 16***. The derivative shown is the lift force derivative w.r.t. rate of change of angle of attack for a dataset size of 200.

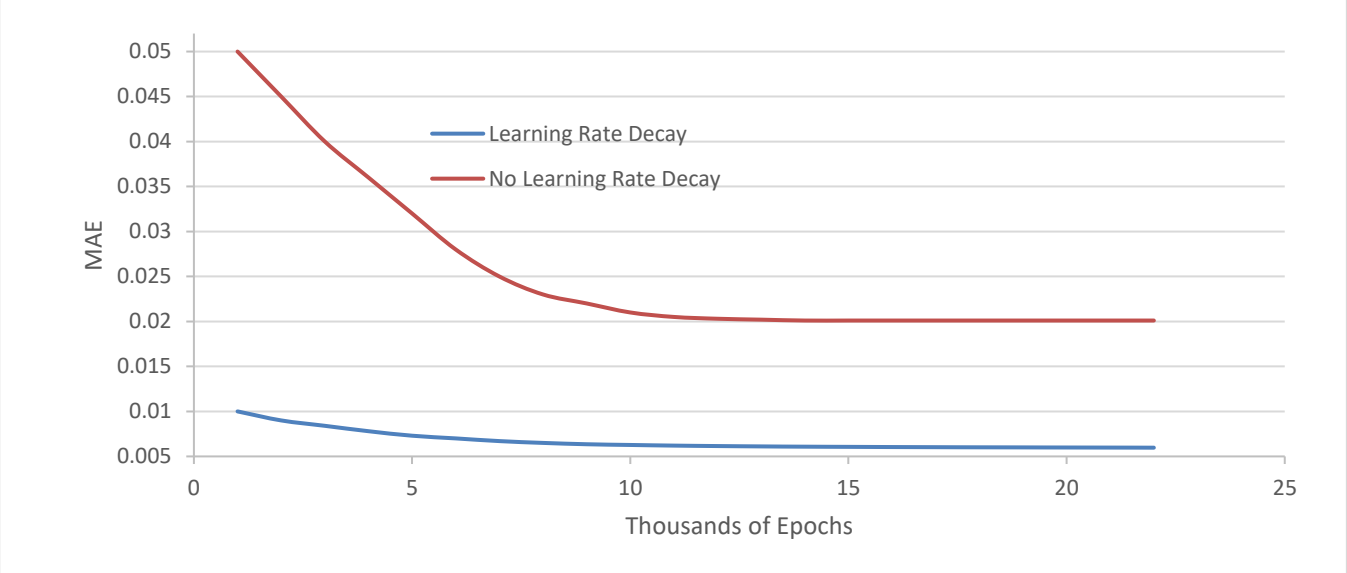

***Figure 16** – Illustration of the effect of a smaller learning rate with no decay versus a larger learning rate with decay for the coefficient lift force derivative w.r.t. rate of change of angle of attack*

It can be seen how allowing the ANN to train with a lower learning rate from the beginning does not result in a continuous rate of learning. The advantage of the higher learning rate for the configuration with decay is shown by the much better initial performance. The training progress with decay is more even, and does not reach the asymptote caused by accuracy limit as quickly as the case without decay does.

Appendix J – Optimal ANN Configurations

The advanced ANN settings to be used for repeating the results achieved are described in ***Table A2***. The optimal ANNs in this paper are not guaranteed to be the absolute best solutions due to the time considerations associated with finding the optimal configuration. As a result, the results documented are potentially not the best results achievable with an ANN. It is not possible to explain or define each setting here due to the extreme volume of equations and principles involved.

Updater	RMSPropogation with 0.95 decay
Layer Types	Dense
Precision	Double
Weight Initiation	Xavier
Optimisation Algorithm	Stochastic Gradient Descent
Batch Size	5
Input Layer Activation	Rectified Linear Unit
Output Layer Activation	Identity
Validation Dataset Size	Case A, B: 80000 / Case C: 2000

***Table A3** – General artificial neural network configuration settings for all cases.*

Table A3 shows a selection of the optimal ANN configurations for a range of cases, illustrating the number of different configurations required for each coefficient/derivative and case.

Case I	**Case II C_L only**	**Case II C_d only**	**Case III C_L and C_d**
1xTanh w/ 25 nodes 2xSigmoid w/ 25nodes 1xTanh w/ 25 nodes 1xSoftmax w/ 11 nodes	1xTanh w/ 30 nodes 1xSoftmax w/ 11 nodes	1xTanh w/ 47 nodes 1xSigmoid w/ 47 nodes 1xSoftmax w/ 5 nodes	1xTanh w/ 30 nodes 1xSigmoid w/ 30 nodes 1xSoftmax w/ 11 nodes

***Table A4** - Ideal artificial neural network configuration for 2,000 training samples.*

The wide variety of ANN configurations is shown by the varying number of layers and nodes per layers. ***Table A4*** shows how dataset size changes the optimal ANN configuration for the coefficient of lift.

Dataset Size	2000	8000	20000
ANN Configuration	1xTanh w/ 30 nodes 1xSoftmax w/ 11 nodes	1xTanh w/ 36 nodes 1xSoftmax w/ 11 nodes	1xTanh w/ 25 nodes 1xSigmoid w/ 12 nodes 1xSoftmax w/ 11 nodes

***Table A5** - Ideal ANN configuration for differently sized datasets.*

Appendix K - Granularity

Granularity is the result of insufficient decimal places in the Datcom+ Pro output. ***Figure 17*** illustrates how granularity occurs.

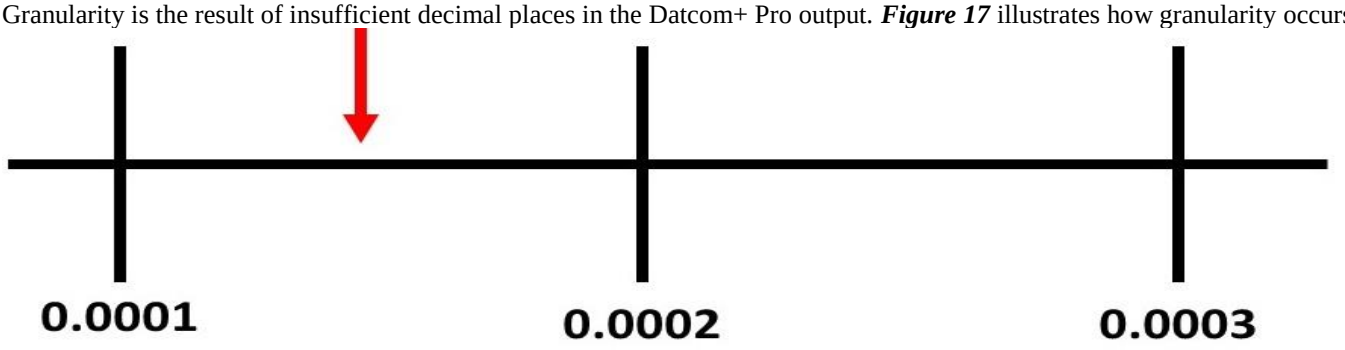

***Figure 17** – An illustration of granularity in action, where the red arrow indicates the ANN's accurate prediction, but due to the limitations of Datcom+ Pro the ANN would be penalised for not predicting to the nearest rounded value.*

There appears to be no literature regarding what method of rounding Datcom+ Pro uses, although always rounding up or down would result in bigger errors than normal rounding. It can be seen how the ANN predicting a value of 0.00015 (even if that would be the correct answer if the Datcom+ Pro output did go to five decimal places) would always be penalised regardless, enforcing a hard limit on the accuracy of the derivatives and coefficients. For these outputs with a large range of possible values this is not as much of an issue, but for values that have very little variation the penalising of technically correct answers is an issue.

Appendix L – Turnitin Report

Appendix M – Declaration of Work

It is hereby declared that this report is entirely my own work, unless otherwise stated, and that all sources of information have been properly acknowledged and referenced. It is also declared that this report has not previously been submitted, in whole or in part, as part fulfilment of any module assessment requirement.

Signed: _____ Date: _____