Learning Aerodynamics Through Data to Improve Optimization Algorithms

Mario Leupolt* and Jospeh Morlier†

The calculation of the aerodynamic coefficients of an airfoil with established methods is a time consuming and computationally expensive process. This work takes a deep learning approach to determine the aerodynamic coefficients of a given airfoil much faster. It uses an already existing data base and tries three different platforms (Keras and SMT package in python and Monolith AI) to predict the coefficients as accurately as possible. The newly obtained surrogate model can then be used to give scalar and graph predictions for certain Mach numbers and angles of attack or in optimization algorithms to calculate the aerodynamic coefficients in the optimization steps.

I. Nomenclature

$\boldsymbol{A}$	=	coordinate matrix
$\boldsymbol{a}$	=	mode shape vector
b	=	bias
c	=	camber index
i	=	waypoint index
j	=	neuron index
t	=	thickness index
$\boldsymbol{U}$	=	mode shape matrix
$\boldsymbol{V}$	=	right singular values matrix
w	=	weight
x, y	=	scalar value
$\boldsymbol{x}$, $\boldsymbol{y}$	=	scalar vector
$\boldsymbol{\Sigma}$	=	singular value matrix
φ	=	generic function
$\boldsymbol{\phi}$	=	mode shape matrix

II. Introduction

The selection of the airfoil geometry can be seen as one of the most crucial parts in the aircraft design process. It influences the cruise speed, take-off and landing distances and and determines the aircraft overall aerodynamic performance [1]. In the beginning of aircraft history the aerodynamic parameters were estimated using experimental methods. With the rise of available computational power, simulation methods were applied for their calculation, for example using the Reynolds- averaged Navier-Stokes (RANS) equations. Despite the accuracy of these calculations there are also major disadvantages. They cost a lot of time and computational power [2].

To reduce the time and needed computational power a deep learning approach will be taken in this work using an already existing database of ten-thousands of airfoils with their respective aerodynamic coefficients. In deep learning artificial neural networks (ANN) try to imitate the working processes in the human brain. Inputs are fed into the network which consists of numerous layers of so-called neurons. They process the data and issue an output. This output is then compared to the output that should have been issued. On the base of the difference between the predicted value and the true value the network can learn. The learning phase is finished when the network reaches a certain accuracy measured through different metrics. After that it can predict outputs from not-learned input values. [3]

*Student, ISAE SUPAERO, mario.leupolt@student.isae-supaero.fr

†Professor, ISAE SUPAERO, joseph.morlier@isae.fr

The neural networks build in this paper will be trained with an already existing database developed by Bouhlel et al. [4]. To build the database the researchers first performed a singular value decomposition (SVD) of the camber and thickness lines of 1172 airfoils in the University of Illinois at Urbana–Champaign (UIUC) airfoil database to obtain their mode shapes and to have a big design space. Following, a database for the subsonic and transonic regimes was built with 81000 and 32400 airfoils respectively that were controlled by 14 and eight mode shapes, respectively. The subsonic regime is in a range between Mach numbers of 0.3 to 0.6 and the transonic regime ranges from Mach numbers of 0.65 to 0.85. Their aerodynamic coefficients were calculated using a RANS method for alphas in the range of -2 to 6 degrees. [5]

It followed the construction of a surrogate model to predict the coefficients. A gradient enhanced kriging with partial least squares (GE-KPLS) with a mixture of expert has been performed. A GE-KPLS is an improvement of a gradient enhanced kriging (GEK) model. A GEK can reduce the number of evaluations drastically by using the gradient information, but two problems occur with the rise of sample points and independent variables. Firstly, the correlation matrix grows rapidly and secondly more hyperparameters must be estimated. GE-KPLS solves this problem by using the partial least squares method [6]. A mixture of experts was introduced because the amount of data is still difficult to handle by one GE-KPLS surrogate model. The variable space was split and distributed to different surrogate models. Two independent surrogate models were built for the subsonic and transonic regimes, respectively.

In their following work the GE-KPLS was improved to a variation of an artificial neural network. The applied neural network uses the Sobolev training method where the gradient information is passed to the loss function of the training samples during the training phase in addition to the target values [7]. Bouhlel et al. [8] modified the Sobolev training method for their artificial neural network slightly, so the gradient information is passed gradually to the loss function. Their obtained results show an improvement to the first approach with the GE-KPLS surrogate models. Another improvement was the utilisation of a single surrogate model for both flow regimes.

Using two different python packages, Keras and the surrogate modelling toolbox (SMT), and the platform Monolith AI this work aims to learn the mentioned database as well as possible. The concentration lays on the subsonic regime. This will provide a tool to predict the aerodynamic coefficients lift, drag and moment for random subsonic airfoils efficiently. The tool can be applied to accelerate optimization algorithms for example. Keras is a deep Learning application programming interface which uses the platform tensorflow, a well-established platform for deep learning problems [9]. Keras provides an easy user interface with high flexibility. SMT is a newly developed platform which provides a collection of surrogate modelling methods and sampling and benchmark functions. It focuses on the use of derivatives which can be used for gradient enhanced networks [10]. Monolith AI is an online platform which learns from data of previous calculations that are uploaded to the platform. After the learning process the platform can produce results for new structures in parameterising them and solve optimization problems [11].

The rest of the paper is organized as follows. First the problems that can occur during the work are stated and possible solutions are proposed in Section III. Then an introduction to ANN is given and the methodology for parameterising the airfoil is described in Section IV. Following, the construction of the neural networks on each platform is described in Section V. Next, the results are presented in Section VI. The final Section will conclude the results obtained in this paper and give an outlook on future work.

III. Problem statement

Despite having a big database and therefore no need to build our own database a few problems can occur. To build the best neural network possible the hyperparameters and architecture for the networks must be chosen carefully. The determination of the hyperparameters can be realized carrying out a hyperparameter study. Since there are three outputs to predict it is possible to build one network that predicts all three aerodynamic coefficients or separate networks for each coefficient, so the predictions for both methods must be compared.

To test and use the obtained networks, the input airfoils must be parameterised using the exact same methods as Li et al. [5] in their work. If the parametrisation is done incorrectly the networks receive a faulty input and will compute incorrect outputs, making the networks deficient despite having a good architecture and well selected hyperparameters.

IV. Background

In this section an introduction into ANN is given and the parameterisation to get from the raw airfoil geometry to the mode shapes is described.

A. Artificial Neural Networks

Artificial neural networks are computational models that try to imitate the working process in the human brain. The development of an ANN is composed of the learning phase, the validation phase, and a testing phase. ANN use supervised learning which means it learns using previous collected examples where the output was already determined.

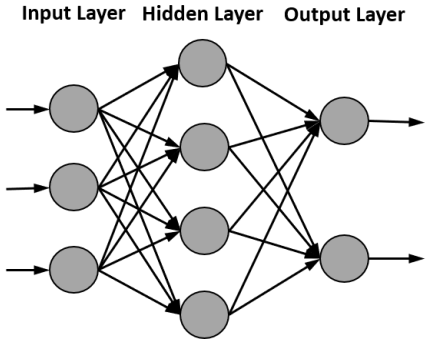

Fig. 1 simple architecture for an ANN.

The networks consist of several layers (the input layer, the hidden layers, and the output layer) that each consist of several so-called neurons. An example for a simple architecture of an ANN can be seen in Fig. 1. If a neuron receives an input, it processes it and produces a state of activation as an output. The calculations inside the neuron are often done using:

$$y_j = \varphi\left(\sum_{i=1}^{n} w_{ij}x_i + b_j\right). \tag{1}$$

Layers that use this formula inside their neurons are called Dense Layers and are used in multi-layer perceptrons. The formula states that all the inputs x_i it receives from the neurons are multiplied by the corresponding weight w_{ij} and added up. A bias b_j is added to the sum to shift the function. n represents the number of neurons of the previous layer. The result from this calculation is then passed into the activation function φ that performs a non-linear transformation. The selection of the activation function can be crucial for the success of the neural network because activation functions perform differently depending on the use-case. In this work the ReLU, the Leaky-ReLU, the SeLU, the tanh and the sigmoid activation functions are used.

The training phase of a neural network aims to alter the weights and the biases after each learning step to improve the predictions of the network in the next step. The predicted output of the network is compared to the true output with the loss function. With the help of an optimizer the network can update its weights and biases on the base of the gradient between the true and predicted output using backpropagation. In this work the Adams optimizer is used. [12]

B. Parameterisation

Mode shapes of the examined airfoils must be input to get predictions from the later presented neural networks. This section describes how to get the correct mode shapes from the original airfoil coordinates with a random number of points.

The mode shapes that control the parametrised airfoils are seven camber mode shapes and seven thickness mode shapes. They derived from the thickness and camber lines of 1172 airfoils from the UIUC airfoil database to cover a huge design space, which is particularly useful for optimization problems. Using the camber and thickness lines instead of full airfoil coordinates as the base for the mode shapes also brings the advantage of having better control over the optimization bounds. For example, thickness constraints can be applied more directly which means that more airfoil shapes are excluded before the optimization starts which leads to a more efficient computing process. The researchers performed an SVD on the camber and thickness lines: [5]

$$\boldsymbol{A_t} = \boldsymbol{U_t}\boldsymbol{\Sigma_t}\boldsymbol{V_t^T}. \tag{2}$$

As an example, the SVD of the thickness lines is given in Eq. 2. m y-coordinates of n thickness lines are arranged in Matrix $\boldsymbol{A_t}$ with the same x-coordinates distribution. The SVD gives three separate matrices. $\boldsymbol{U_t}$ represents the mode

shapes and is a m x m matrix. $\boldsymbol{\Sigma_t}$ is a m x n matrix with the singular values of $\boldsymbol{A_t}$ in the diagonal of the matrix. The columns of $\boldsymbol{V_t}$ are right right singular vectors. The distribution of the first camber and thickness modes of the airfoils in the UIUC database can be seen in Fig. 2.

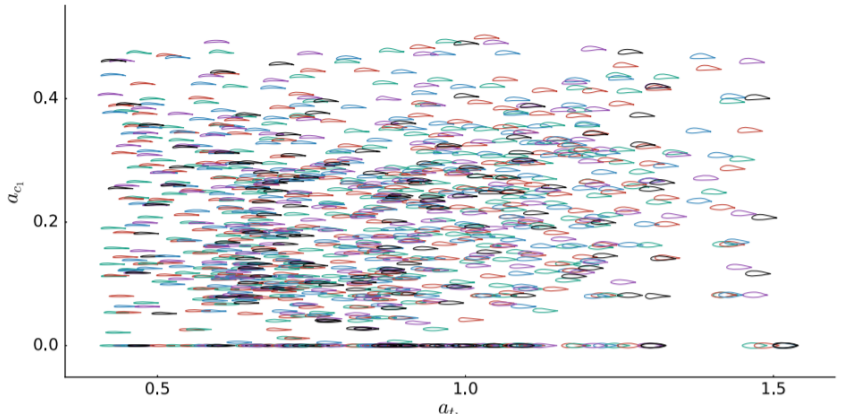

Fig. 2 The distribution of the UIUC airfoils on the base of their first camber and thickness mode shapes. [5]

To arrive at the mode shapes for a specific random airfoil, the airfoil must be pre-processed, which means that a uniform distribution scheme of the points will describe the geometry for the later following parameterisation. First, the airfoil is interpolated with a cubic B-Spline. The airfoil can then be represented by a total of 249 points which are distributed as follows:

$$x_i = \frac{1}{2}\left(\cos\left(\frac{2\pi(i-1)}{250}\right)+1\right) \qquad i = 1...251 \tag{3}$$

where the first and last value in the $\boldsymbol{x}$-vector are not used for the point distribution. The leading edge is set to zero and with the help of a Laplacian smoothing algorithm the curve is smoothed. After the smoothing process, the newly obtained coordinates are used to compute the camber and thickness lines of the airfoil. The results for the camber and thickness lines return a vector for their respective y-values of length 125. Since the airfoil possesses a blunt trailing edge, the first value of the thickness line is not equal to zero. To achieve a zero-thickness 16 more points are added whose x-values are equal to one and whose y-values are distributed linear from zero to the first value of the original thickness line. To make the camber line an equal length, 16 points (0,0) are added.

To determine the camber a_c and thickness a_t mode shapes of the airfoil, the following formulas must be used:

$$\boldsymbol{a_t^T} = 2\boldsymbol{y_t^T}\,\boldsymbol{\phi_t} \qquad\qquad \boldsymbol{a_c^T} = \boldsymbol{y_c^T}\,\boldsymbol{\phi_c} \tag{4}$$

The y-coordinates of the camber $\boldsymbol{y_c}$ and thickness $\boldsymbol{y_t}$ lines must be multiplied with their respective modes ($\boldsymbol{\phi_c}$ and $\boldsymbol{\phi_t}$). Additionally to the matrix multiplication between the thickness line and thickness modes, the obtained thickness mode shapes must be multiplied by two because of the following equation:

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{\phi_c} & \boldsymbol{\phi_t} \\ \boldsymbol{\phi_c} & -\boldsymbol{\phi_t} \end{pmatrix} \begin{pmatrix} \boldsymbol{a_c} \\ \frac{1}{2}\boldsymbol{a_t} \end{pmatrix}. \tag{5}$$

The equation describes how to get to the airfoil geometry with the mode shapes. A matrix multiplication between a camber and thickness mode matrix with the camber and thickness mode shapes must be carried out. The thickness mode shapes must be divided by two for the calculation to work. For the predictions exactly the vector containing the half thickness and camber mode shapes is used. In Fig. 3 the NACA4412 airfoil is shown, once in its original shape and once after the parameterisation process. It is observable that the geometry calculated from the mode shapes varies slightly from the original geometry.

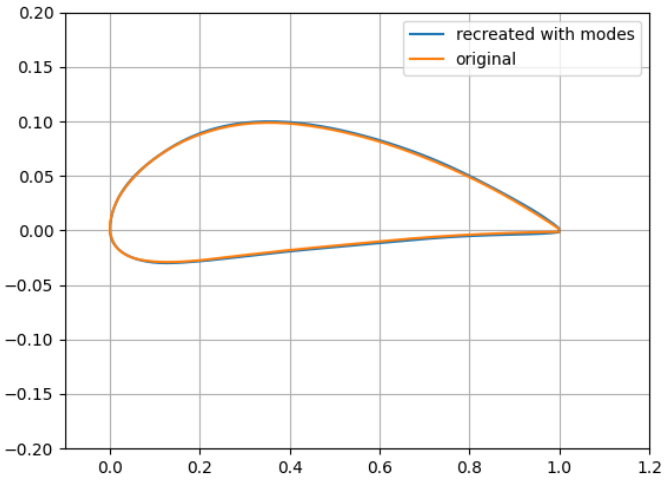

Fig. 3 The airfoil geometry of a NACA4412 profile obtained with the computed mode shapes in comparison to the original airfoil geometry

V. Methodology

After explaining the background behind the networks and describing the parameterisation process the workflow for each platform is presented in this section.

A. Keras

Keras provides an easy user interface. Neural networks are built fast and efficient. Different methods were used in this work to get to an optimal network that predicts the coefficients as precisely as possible.

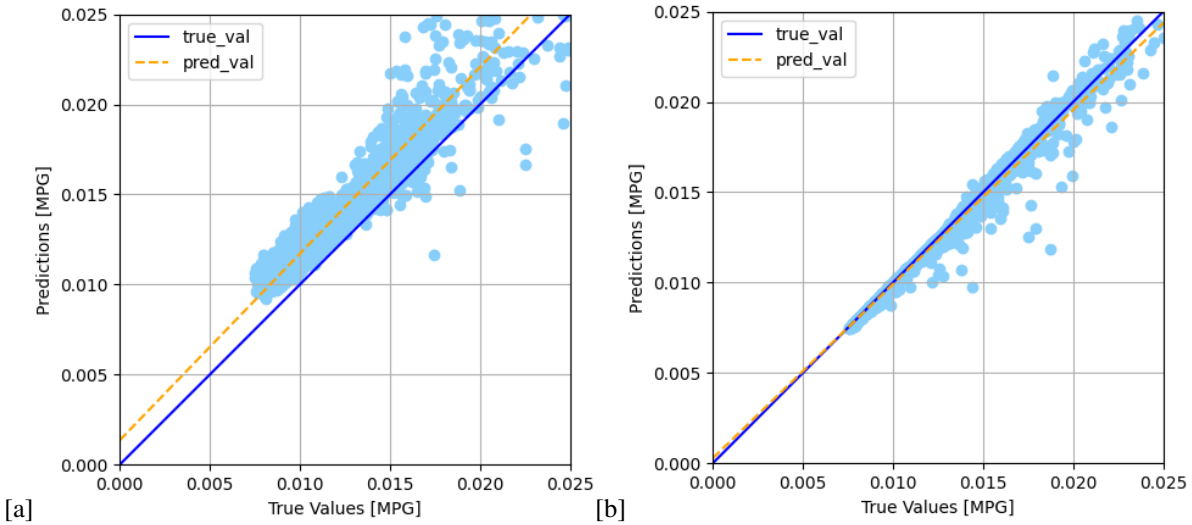

Fig. 4 The results for the predictions of the drag coefficient in Keras for the first built models. Figure [a] displays the results for the model that learned all coefficients, Fig. [b] the results for the model that only learned the drag coefficient. The orange line represents a regression through the points.

First, a model with a very simple architecture was built for each coefficient. They consisted of the input layer, one hidden layer with 1000 neurons and the output layer. The activation function was the ReLU function, and the Adams optimizer was used. As the loss function the mean absolute error (MAE) was chosen, also being the loss function for all the following networks built in Keras. Chosen metrics were the coefficient of determination (R2) and the Root Mean Square Error (RMSE). A good R2 value would be in the range between 0.95 to 1 for this regression problem and the

nearer the RMSE value is to zero, the better is the result. No specific sampling methods were selected for this network.

To decide if the tool for the predictions should be based on separate neural networks for each aerodynamic coefficient or one neural network for all coefficients, a model with a slightly deeper architecture for all coefficients was built. It consisted of three hidden layers with 1000 neurons each; all being activated by the ReLU function. The obtained results were then compared to the results from the models for separate predictions. The quality of the predicted values for the drag coefficient is shown in Fig. 4. On the left side the results for the model that learnt all coefficients is shown. On the right side the results for the model that only learned the drag coefficient is displayed. As it can be seen, the quality of the predicted values from the model for all coefficients is worse despite having a more profound network architecture. Thus, the decision was made to build separate neural networks for each coefficient. This correlates with the findings in the works of Bouhlel et al. [8] who also built separate networks.

Since the quality of the predictions from the models with the simple architecture was not sufficient two further architecture approaches were taken. One approach includes using several Long short-term memory (LSTM) layers instead of Dense layers (Eq. 1) and the other approach takes the architecture from Du et al. [13] with several Dense layers as base for a hyperparameter study. LSTM layers consist of recurrently connected blocks, so-called memory blocks. Each block possesses one or more recurrently connected memory cells. Additionally, it contains three multiplicative units: the input gate, the output gate and the forget gate. This inner architecture allows the write, read, and reset operations for the cells in several continuous time steps [14].

Usually, the LSTM layers are used in time dependent problems, but in tests from Du et al. it was shown, that simple regression problems can also be solved using LSTM layers in feeding every input mode shape as input for one time step. In a hyperparameter study the model is trained in a loop. For every loop one hyperparameter is changed, to see the effect it has on the model performance. After the training loop, the hyperparameter pairing that ended up delivering the best performance is chosen for the final network.

Table 1 RMSE and R2 values for the best and worst performing models in the hyperparameter study

Model	Drag Coefficient	Lift Coefficient	Moment Coefficient
Best RMSE	0.0002	0.0035	0.0005
Best R2	0.9855	0.9997	0.9997
Worst RMSE	7.532	0.102	0.033
Worst R2	-4.9e8	-0.292	-0.256

To validate both, the hyperparameter study for Dense layers and the LSTM layer approach the models will be first evaluated using k-cross validation. In k-cross validation the dataset is split in k equally sized parts. The network is trained using $k-1$ sets of the data as the training set and one set as the validation set. It is possible to see how the network reacts on different inputs it did not learn already using this method. Thus, it can be estimated how well the network will perform in the end. For example, if a network architecture provides good results after each of the training cycles, the network will perform well after the final training. In this work k was set to five. [15]

For the LSTM layer approach, a hyperparameter study could have been possible in theory but due to long training times this option was not considered. Different numbers of LSTM layers were tried and additionally the unit size of the layers was altered and several learning rates for the Adam optimizer were applied. The epoch number, that means the number of training steps, was always at 250. The R2 scores were between 0.90 to 0.97 and the RMSE was found to be in the range between 0.0010 and 0.0011 for the drag coefficient. This leads to the conclusion that the predictions of the model are reasonably well.

36 possible network architectures were examined in the hyperparameter study. Since each network was validated five times because of cross validation, the total amount of training phases was 180. The number of neurons was an examined hyperparameter. Either 20 neurons were added or subtracted from the number of neurons of the base structure or the number of neurons of the base structure was kept. Additionally, the learning rate of the Adam optimizer was varied [0.001, 0.005, 0.0001, 0.0005, 0.00001, 0.00005] and finally the scaling of the data was either set to True or False. The number of epochs was always set to 500.

Every loop brought different results, sometimes varying drastically. To give an overview of the margins between the worst and best performing model for each aerodynamic coefficient the RMSE and R2 values of both are shown in Table 1. After evaluating the results from the hyperparameter study and the LSTM layer approach, the decision for the final architecture for each neural network was made. In Table 2 every layer is described with its activation function

Table 2 architecture of the neural networks built in Keras

Model	Drag Coefficient		Lift Coefficient		Moment Coefficient	
Learning Rate	0.001		0.0005		0.0005	
R2-Value	0.9998		0.9995		0.9995	
RMSE-Value	0.0007		0.0074		0.0012	
Layers	tanh	180	tanh	100	tanh	180
	sigmoid	160	sigmoid	120	sigmoid	160
	selu	140	selu	140	selu	140
	selu	140	selu	120	selu	120
	selu	140	Leaky ReLU	180	Leaky ReLU	80
	Leaky ReLU	120	Leaky ReLU	180	Leaky ReLU	40
	Leaky ReLU	120				
	Leaky ReLU	120				

and number of neurons. Additionally, the information about the learning rate, the final R2 and RMSE value is given, respectively. The evaluation of the hyperparameter study also led to the conclusion that the data must not be scaled to arrive at a model that gives good predictions. The graphs for the predicted against the true values are shown in Fig. 5 for every aerodynamic coefficient.

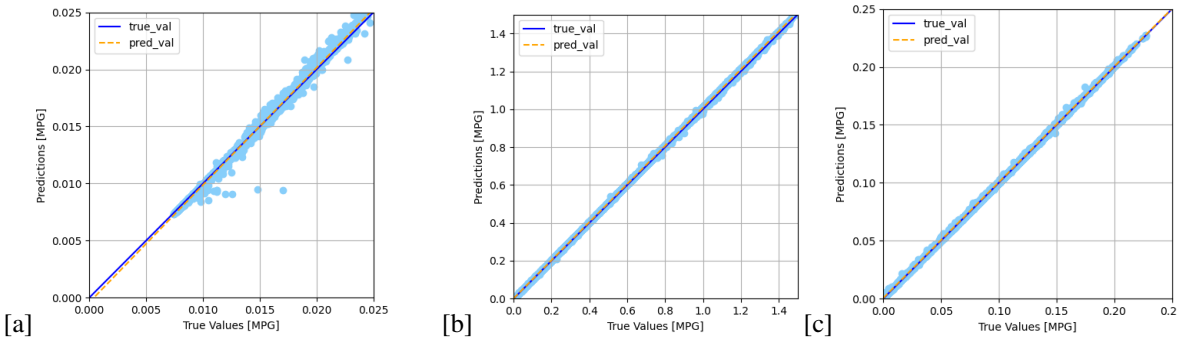

Fig. 5 The results for the predictions of the aerodynamic coefficient in Keras. Figure [a] displays the results for the drag, Fig. [b] the results for the lift and Fig. [c] the results for the moment coefficient. The orange line represents a regression through the points.

B. SMT

The python package SMT provides a selection of surrogate modelling functions, sampling techniques and benchmarking functions. It is a newly developed toolbox that provides surrogate models that are not available in other toolboxes for example the kriging by partial least squares. It also simplifies the application of the gradient information. The application of the gradient information can improve surrogate model results by a lot but also lead to higher computational cost in the training phase. [10] In the database of Bouhlel et al. the gradient information from their calculation for the aerodynamic coefficients was included therefore it can be used when building models with SMT. Since this work focuses more on neural networks, none of the other surrogate models will be applied directly on the database in this work.

Nevertheless, a mixture of experts (MOE) was applied on the whole data to see how clustering the data could improve the predictions for the coefficients. The MOE function in SMT clusters the database in a predefined number of clusters using the Gaussian Mixture function from *"sklearn"*. During the execution of the MOE function it applies different surrogate modelling methods on each cluster. Since SMT is still in its early stages the customization of the clustering in the MOE function was not really possible and just surrogate models not using the gradient information

were examined. Therefore, it was assumed that the results from the MOE approach will not yield better results than the neural networks built in SMT that use the gradient information during training phase. The predicted against the true values for the drag coefficient from the MOE for 20 clusters is shown in Fig. 6 As can be seen there are multiple outliers which could lead to a bad prediction quality. Another disadvantage of the MOE function were the long calculation times. Due to the huge amount of data some of the surrogate modelling methods take very long in their computation phase. Therefore, just the MOE for 20 clusters was examined and not different amounts of clusters which could have potentially brought better results.

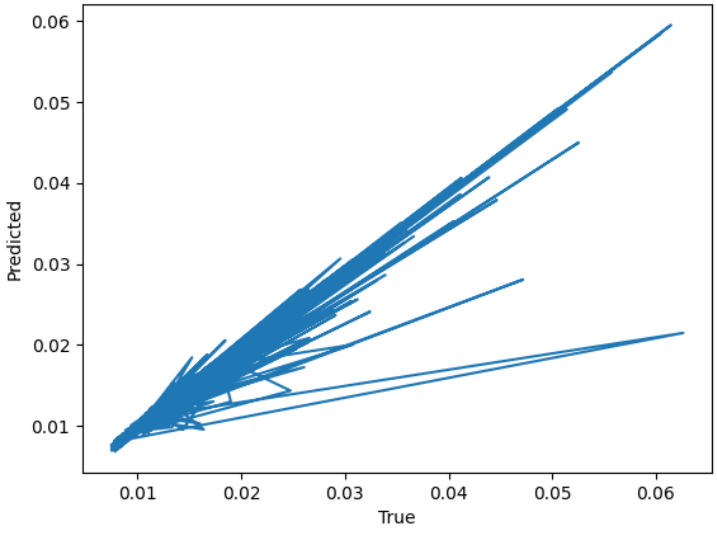

Fig. 6 The results for the predicted against the actual values using the MOE function for 20 clusters.

Since SMT provides the option to include the gradient information in its surrogate models, a gradient enhanced neural network (GENN) was built using the gradient information Bouhlel et al. provide in their database. The gradients were computed using an adjoint method.

The GENN is a multilayer perceptron that includes the gradient information in the training process. Normally a neural network tries to minimize the prediction error from the response. Additionally, the GENN try to minimize the prediction error of the partial derivatives (gradients). This leads to a higher accuracy with fewer training steps and smaller network sizes. Nevertheless, they are only applicable to regression problems used for predictions on the base of computer aided design data because there the computation of the gradient information is possible. [10]

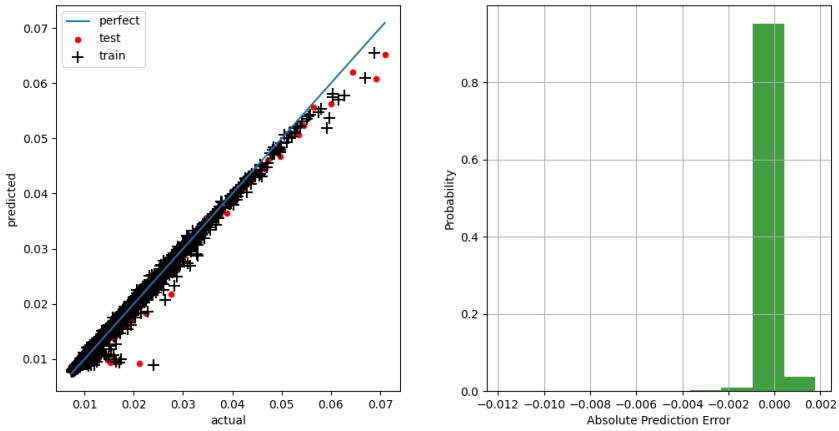

Fig. 7 The results for the training of the GENN in SMT for the drag coefficient. On the left, the predicted against the actual values are displayed. The right graph gives information about the distribution of the absolute prediction error.

It was again decided to build separate GENN for each aerodynamic coefficient. The architecture for all GENN is the same. In SMT several hyperparameters are customizable. The Adams optimizer functions as the optimizer. Its learning rate can be set, as well as two further tuning parameters. They were all left on default settings. The network consisted of two hidden layers, each with six nodes. The size for the batch that was passed through the network in each training step was set to 256 and in each training step the optimizer was set to iterate ten times. After setting up the network architecture and the settings for the training phase the training was carried out in 25 training epochs. In Fig. 7 the results for the drag coefficient are shown as an example. The results for the lift coefficient and the moment coefficient were of equal nature. On the left the predicted against the actual values is presented while on the right the distribution of the absolute prediction error is shown.

Both graphs indicate that the trained GENN can predict the drag coefficient on the base of mode shapes very well. In comparison to the mixture of experts approach that was tried in this work, the GENN is clearly superior.

C. Monolith AI

The last platform that will help building neural networks is Monolith AI. It is an online platform where data from previously carried out calculations can be uploaded. After uploading the data, the data can be processed and learnt using different model types. On the base of the obtained model, curve and tabular predictions can be computed, and optimization problems can be solved for example. [11]

In this work a neural network that includes all the aerodynamic coefficients was built, as well as separate networks for each coefficient. In addition to the aerodynamic coefficients a new column was added in the data of the neural network for all coefficients. This column is the result from the ratio between the lift and drag coefficient. On the base of the ratio an optimization was carried out which optimizes the mode shapes such that the ratio is the highest possible at a certain Mach number and alpha.

The built network for all the coefficients had six hidden layers with 100 neurons. The optimizer that is used in Monolith AI is the Adams optimizer and the activation function for the neural network is the ReLU function. The network was trained in 300 epochs. In comparison to the network for all the coefficients the architecture of the separate networks for each coefficient was smaller. They were composed of three hidden layers with 100 neurons. The training phase was done in 250 epochs. In Fig. 8 the results for the predicted against the actual value for the drag coefficient are shown. It can be observed that the predictions from the neural network for all coefficients is slightly worse. That could lead supposedly to more inaccurate predictions when the model will be tested against a test case over a range of alpha at a specific Mach number.

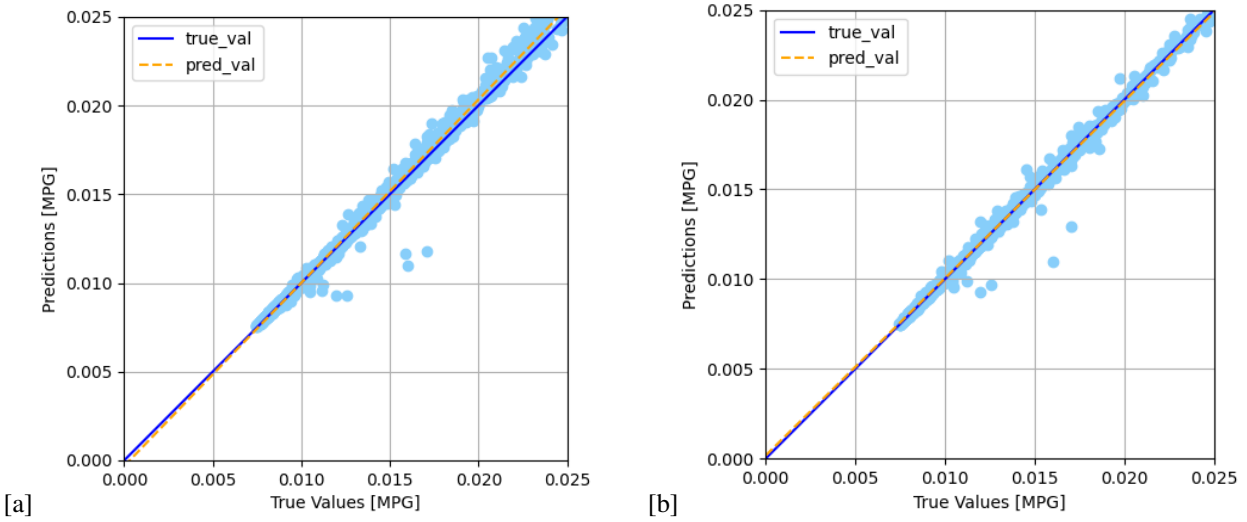

Fig. 8 The results for the predictions of the drag coefficient in Monolith AI. Figure [a] displays the results for the model where all coefficients were trained. Figure [b] shows the results for the model that was only trained for the drag coefficient. The orange line represents a regression through the points.

VI. Results

This Section presents the results obtained using the built neural networks. First the optimization from Monolith AI is evaluated. Following the obtained neural networks are tested against the predictions from the surrogate model from Webfoil and the ADFlow calculations for the NACA0012 and the NACA 4412 profile. Webfoil [16] is the GUI that uses the surrogate models that were built by Bouhlel et al. The aerodynamic coefficients in the database were calculated using ADFlow. [8]

A. Optimization with Monolith

This optimization approach shall show, how well the results from optimization processes using mode shapes to control the geometry perform. The optimization should give the airfoil geometry with the best possible lift to drag ratio at a specific Mach number and angle of attack. The first optimization that was carried out was with no restrictions for the mode shapes. A second optimization was carried out afterwards, limiting all mode shapes except for the first thickness and camber mode shapes. The results for the airfoil geometry obtained in both cases is shown in Fig. 9.

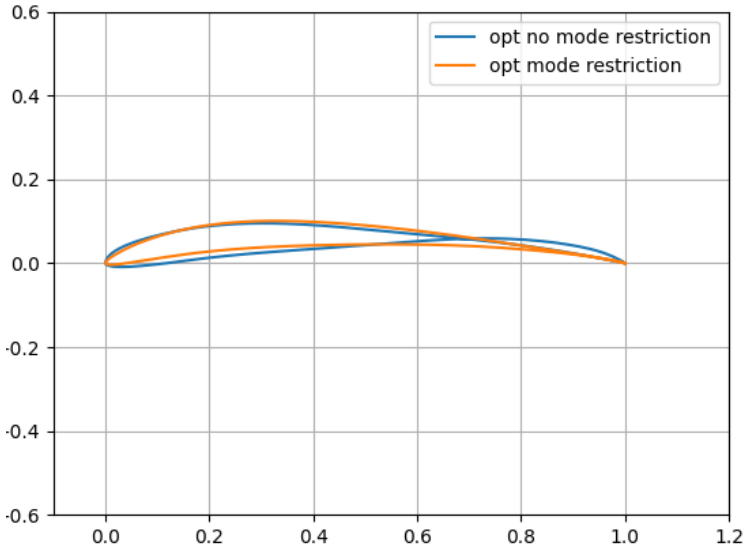

Fig. 9 The optimized airfoil shapes from Monolith AI.

It is observable, that the optimization with no limitation for the mode shapes produces a non-realistic airfoil geometry. Restricting the mode shapes to a certain range already produces a more realistic airfoil geometry. Nevertheless, the trailing edge is still very sharp. Determining optimal bounds for the mode shapes could not be done in this work because of time issues but should be a goal in future work.

B. Predictions

This subsection presents the predictions obtained from each platform for the NACA0012 and the NACA4412 profile and compares it to the predictions from the surrogate model from Webfoil and the ADFlow results which is the CFD tool, that calculated the aerodynamic coefficients for the database.

The mode shapes used as inputs for the neural networks were computed as described in Section IV.B. The drag, lift and moment coefficients over the alpha range between -2 and 6 degrees from the NACA0012 and the NACA4412 profiles can be seen in Fig. 10 and Fig. 11, respectively.

In both cases the SMT predictions fare the closest to the ADFlow results. The biggest discrepancies between the predictions and the ADFlow calculations are observed in the prediction for the moment coefficient and the best quality of prediction gives the lift coefficient. The good quality of the prediction for the lift coefficient results from the linearity of the lift over alpha. This is a relation relatively easy to learn for the neural networks. What is also interesting to see, is that the predictions from the SMT and Keras models surpass the predictions from the surrogate model used in Webfoil. This is especially visible in the prediction of the moment coefficient for the NACA4412 airfoil. Nevertheless, the KERAS model can still be improved because it not always produces a smooth curve, which is observable in the

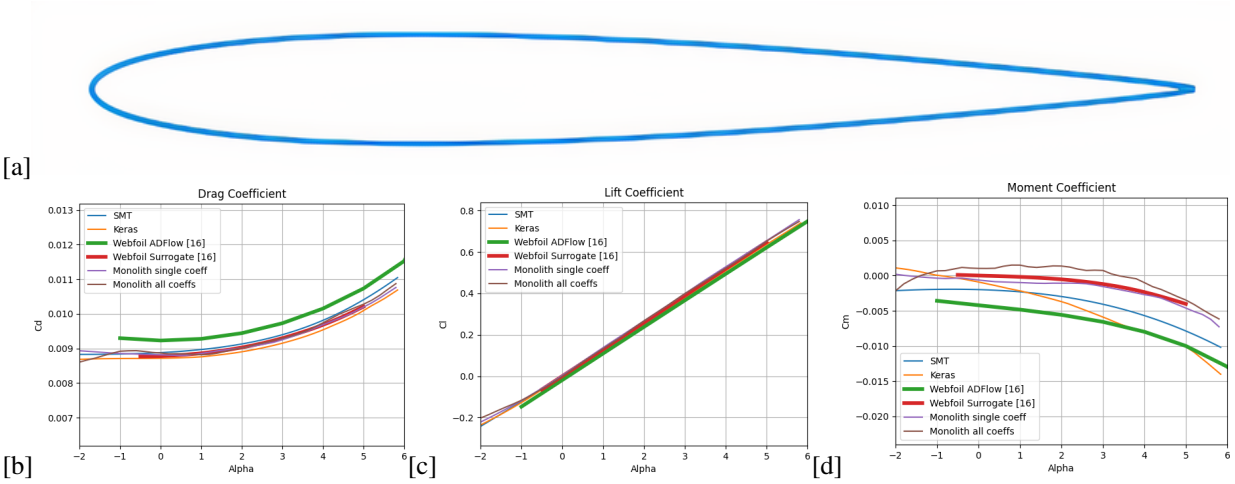

Fig. 10 The results for the predictions of all aerodynamic coefficients from the NACA0012 profile over an alpha range in comparison to the CFD results from ADFlow and the Webfoil predictions.Figure [a] represents the airfoil geometry obtained through the mode shapes.

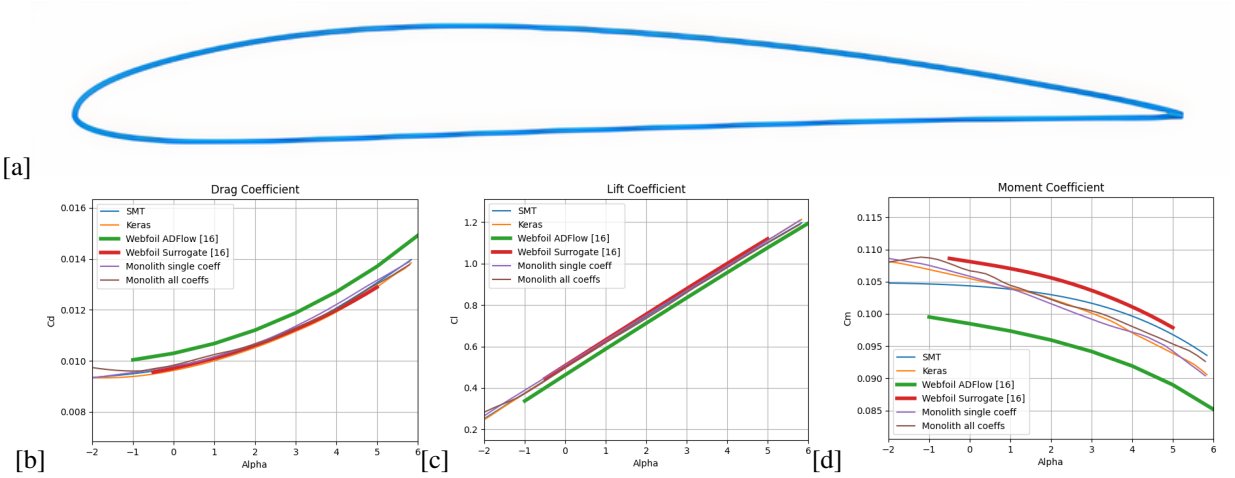

Fig. 11 The results for the predictions of all aerodynamic coefficients from the NACA4412 profile over an alpha range in comparison to the CFD results from ADFlow and the Webfoil predictions. Figure [a] represents the airfoil geometry obtained through the mode shapes.

prediction for the moment coefficients for the NACA0012 profile for example.

The Monolith AI predictions differentiate from each other as expected. The model that learnt all the coefficients at once yields worse results than the separate models. Furthermore, the prediction quality from Monolith AI is the worst as it produces the least smooth curves and has the biggest difference to the calculations from ADFlow. This can be a result of using just the ReLU function in the layers of its models and too few layers in the model architecture.

VII. Conclusion

In this work several neural networks on three different platforms (Keras, SMT, Monolith AI) were built using an already existing data base. The database includes ten-thousands of airfoils with their respective aerodynamic coefficients at a specific Mach number and alpha. The aim was to develop models that can predict the aerodynamic coefficients as accurately as possible. They could then be included in optimization algorithms, to accelerate the computation time, since no more CFD calculations must be carried out during the optimization. For this work only the subsonic part of the database was used. The airfoils in the database are controlled by camber and thickness mode shapes.

To get predictions from the networks, these mode shapes must be input. The parameterisation that was described in this paper produces the necessary mode shapes using a B-Spline interpolation to get from the raw airfoil geometry to a geometry with a uniform distribution scheme. Out of the geometry the camber and thickness lines are computed to then help calculating the camber and thickness mode shapes from the airfoil.

Optimizing an airfoil geometry to have the best possible lift to drag ratio in Monolith AI showed that the mode shapes must be restricted during optimization processes because with no mode shape bounds the optimization produces non-realistic airfoil geometries. Finding the optimal mode shape bounds is as well a goal for future work.

The prediction from the different neural networks that were obtained during this work showed results differing in quality. While the models on Monolith produced not very smooth curves for the coefficient over alpha predictions, the SMT models using the gradient information showed very good results. The SMT models also correlated the best with the results from the CFD simulation carried out on ADFlow which are the basis for the database. The models developed in Keras also showed good results but were not as precise as the SMT models.

To conclude it can be said, that the SMT models are the best models developed in this work. This also proves that using the gradient information in neural networks produces certainly better models for regression problems. The SMT model could then be applied in an optimization algorithm with the optimal mode shape bounds. This can be done in future works.

References

[1] Raymer, D. P., "Aircraft Design: A Conceptual Approach," AIAA, 1992.

[2] Forrester, A. I. J., Sóbester, A., and Keane, A. J., "Engineering Design via Surrogate Modelling," Wiley, 2008.

[3] Schmidhuber, J., "Deep Learning in Neural Networks: An Overview," Elsevir, 2014.

[4] Bouhlel, M. A., He, S., and Martins, J. R. R. A., "mSANN Model Benchmarks," Mendeley Data, 2019. https://doi.org/10.17632/ngpd634smf.1.

[5] Li, J., Amine Bouhlel, M., and Martins, J. R. R. A., "Data-Based Approach for Fast Airfoil Analysis and Optimization," AIAA Journal, 2019.

[6] Bouhlel, M. A., and Martins, J. R. R. A., "Gradient-Enhanced Kriging for High-Dimensional Problems," Springer-Verlag, 2018.

[7] Czarnecki, W. M., and et al., "Sobolev Training for Neural Network," Cornell University, 2017.

[8] Amine Bouhlel, M., He, S., and Martins, J. R. R. A., "Scalable Gradient-Enhanced Artificial Neural Networks for Airfoil Shape Design in the Subsonic and Transonic Regimes," ResearchGate, 2020.

[9] Keras, "Keras Documentary," , 2021. URL https://keras.io/about/, online accessed on 05/05/2021.

[10] Bouhlel, M. A., Hwang, J. T., Bartoli, N., Lafage, R., Morlier, J., and Martins, J. R. R. A., "A Python surrogate modeling framework with derivatives," 2019, p. 102662. https://doi.org/https://doi.org/10.1016/j.advengsoft.2019.03.005.

[11] MonolithAI, "Monolith," 2021. URL https://www.monolithai.com/industry/reduce-testing, online accessed on 05/05/2021.

[12] Michelucci, U., "Applied Deep Learning," Springer Science, 2018.

[13] Du, X., He, P., and Martins, J., "Rapid airfoil design optimization via neural networks-based parameterization and surrogate modeling," Elsevier, 2021.

[14] Hochreiter, S., and Schmidhuber, J., "Long Short-Term Memory," Neural Computation, 1997.

[15] Refaeilzadeh, P., Tang, L., and Liu, H., "Cross-Validation," Springer US, 2009.

[16] University of Michigan, "Webfoil," 2021. URL http://webfoil.engin.umich.edu/, online accessed on 16/06/2021.