Predicting pressure distribution of a NACA0012 aerofoil using a machine learning approach

Anudeep Ayinaparthi

November 2020

CFD is used extensively to model fluid flows however the process is computationally expensive. In this paper a novel, machine learning approach to model the pressure distribution over a NACA0012 aerofoil is introduced. Through the use of machine learning, a model is developed that successfully predicts C_p for a range of angles of attack and Reynolds number. The current state of the field of machine learning is discussed as well as the problems it is able to address in fluid mechanics modelling, in particular, CFD.

Contents

1 Introduction 2
1.1 Machine Learning . . . 2
1.2 Modelling in fluid mechanics . . . 2

2 Methodology 3
2.1 Preliminary Models . . . 3
2.1.1 Model 1 . . . 3
2.1.2 Tuning the models . . . 3
2.1.3 Model 1 Results . . . 3
2.1.4 Model 2 . . . 3
2.1.5 Model 2 Results . . . 3
2.1.6 Preliminary Models Discussion . . . 4
2.2 Random Forest and Decision Tree Algorithms . . . 4
2.3 Final Model . . . 5

3 Results and Discussion 5

4 Conclusion 7

1 Introduction

1.1 Machine Learning

Machine learning has become a prominent buzzword during the past decade. As advancements in algorithms have occurred, and implementations of these algorithms in easy-to-use interfaces have been developed, the application of machine learning has been vast. From healthcare [14], image recognition [15] to engineering [24], machine learning has tapped into every major industry. This advancement comes from access to big data, high-performance computing, advanced algorithms, and considerable investment by industry [3]. One of the most significant milestones in machine learning has been the development of the deep neural network [9] which has led to human-like performance in tasks such as image recognition and games [23]. Further to this, there exists a multitude of available algorithms that can be purposed towards the development of models in the engineering domain [4] which can deliver excellent performance on par with traditional engineering methods. One of the most exciting applications of machine learning is in fluid dynamics.

1.2 Modelling in fluid mechanics

Recently there has been a surge in research using machine learning in fluid mechanics. ML has been used for flow feature extraction [13], modelling fluid flows [20] and flow optimisation and control [19]. Modelling fluid flows plays a key role in the development of aviation vehicles. Conservation laws can be used to analytically solve simple fluid problems and model the flows; however, in the regions of high Reynolds numbers, the Navier-Stokes equations must be solved which is currently beyond current computational capacity. An alternative approach is to utilise computer simulations to approximate these equations. However, these simulations can be computationally expensive due to the number of iterations required to complete the process. By utilising the ability of machine learning to work with non-linear, non-convex and high dimensional data [2], it is possible to decrease the computational cost of modeling complex fluid flows. The idea, whilst innovative, is not novel. In fact utilising machine learning in modeling flow has been discussed for decades. *Koumoutsakos et al.* [16] developed a neural network to reconstruct turbulence flow fields and the flow in the near-wall region of a channel flow using wall-only information. Recently there has been a resurgence in ML applications in fluid dynamics [5], and now there is widespread research into using ML to augment or improve existing methods in fluid mechanics.

Traditional CFD methods produce highly accurate results however they are computationally expensive. This is why designers tend to use CFD simulations at the end of the design process for design validation. In the design optimisation process, designers quickly iterate over many design options, each of which needs to be explored, analysed and optimised at the same time. Optimisation methods such as the finite difference method [12] are computationally expensive due to the costly CFD process [27]. They are also slow, taking in the order of hours and days to complete because of the slow-feedback from the CFD solver. By utilising machine learning it is possible to decrease the computational cost of optimisation by using machine learning in the CFD stage. As stated, there are a plethora of available algorithms to choose from, however certain algorithms are more suited for the purposes of this paper than others [7]. They are:

- Neural Network
- Random Forests
- Decision Trees

Fuchi et al. [6] were able to enhance low fidelity simulations by using random forests to determine discrepancies between low and high fidelity models. They were able to predict navier-stokes velocity fields with good accuracy. Neural networks are also a powerful tool that are being used extensively to model fluid flows. *Ye et al.* [25] used a convolutional network to model the pressure distribution around a cylinder by using wake flow velocities as an input. This work showed that it is possible to use machine learning to accurately predict pressure distribution around a body, however the paper did not extend to more complex geometries. *Zhang et al.* [26] utilised a convolutional neural network to predict lift coefficient of aerofoils for a range of angles of attack, Mach numbers, and Reynolds numbers. Whilst this work makes an excellent case for the generalisation of machine learning models, it does not prove that this condition can be extended to a more complex output. To extend on these works, this paper will aim to create a machine learning model that is able to accurately predict the pressure distribution around a NACA0012 aerofoil, at a range of angle of attacks and Reynolds numbers.

2 Methodology

2.1 Preliminary Models

2.1.1 Model 1

The first step in this process is to identify which machine learning algorithm is best suited for the task of predicting pressure distribution around an aerofoil. To begin, simple flow conditions are considered. The operating conditions are $Re = 0$, $M = 0$, and the flow is inviscid and incompressible. Pressure coefficient is the target output and is defined by equation 1. The pressure distributions are obtained for a range of angles of attack, α, for $4° <= \alpha <= 7°$. The data is obtained through xfoil, after which the preprocessing step is completed to prepare the data for use in the ML models.

$$C_p = \frac{P}{\frac{1}{2} * \rho * v^2} \tag{1}$$

The pressure distribution data is split into testing and training sets based on angle of attack. The features used to train the model with are:

1. Angle of Attack
2. X-position along aerofoil
3. Y-position along aerofoil

The target feature is the pressure distribution, C_p.

2.1.2 Tuning the models

Hyperparameter tuning for each algorithm is performed using the *GridSearchCV* tool. Different permutations are iterated through to identify the parameter set that results in the lowest RMSE.

2.1.3 Model 1 Results

The metric to analyse performance is Root Mean Squared Error which is defined by equation (2), where y_i is the true C_p and $\hat{y}_i$ is the predicted value.

$$RMSE = \sqrt{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \cdot \frac{1}{n}} \tag{2}$$

Table 1 shows the results of the preliminary investigation. It is observed that the lowest score is presented by decision trees, closely followed by random forests and deep learning algorithms.

Decision Tree	Random Forest	Deep Learning
0.16	0.189	0.77

Table 1: RMSE scores for ML Models

2.1.4 Model 2

Model 2 is extended to consider a more complex flow. The operating conditions are $R = 5x10^6$, $M = 0$, and the flow is viscous and incompressible. The range of angle of attacks is increased so that $2° \leq \alpha \leq 22°$. Towards the higher angle of attacks, boundary layer separation occurs which leads to more complicated pressure distributions that the machine learning algorithm has to capture. The same steps of hyper-parameter tuning are completed as before.

2.1.5 Model 2 Results

The results of model 2 are shown in Table 2. It is clear to see that the decision tree model and the random forest model outperform the deep learning network by a factor of 10. This indicates that decision trees and random forest algorithms are better suited to this task than deep learning algorithms.

Decision Tree	Random Forest	Deep Learning
0.058	0.060	0.935

Table 2: RMSE scores for ML Models

2.1.6 Preliminary Models Discussion

From the initial models it has been determined that the decision tree and random forest algorithms are the models that will be used going forward. However it is useful to discuss why the deep learning method failed to live up to its hype as a reliable and accurate model. A brief explanation of how neural networks learn and how they predict an output is warranted first. Neural networks take an input, x and produce an output, y. The input is mapped to the output via a function, W. Equation 3 shows this. The training process aims to create the function W by a process called back propagation [18]. The model computes an error between the output and input, from which it updates W to reduce error. By repeating this process and reducing error, the neural network aims to produce a function that is able to map the input to the output.

$$y = W(x) \tag{3}$$

Delving through the current research, it is apparent to see that a type of neural network, the convolutional neural network, has had the most success in fluid mechanical simulations, hence why it is widely used [21] [10] [17]. A convolutional neural network works similarly to neural networks in that it also creates a function to map the input to the output, however it re-configures the input data into kernels which make it a more effective algorithm [8]. These types of networks are used extensively with image processing tasks. Neural networks fail to capture patterns when an insufficient volume of data is provided. After initially performing on par with the DT and RF methods, the deep learning method failed to handle more complex data inputs when viscous effects and a larger range of angle of attacks were introduced. This may be a result of having a wide sample of data but not enough depth to create a weight function that can accurately predict the pressure distribution. For the type of data input that this paper has used, a convolutional neural network would not be an appropriate algorithm to use. Future investigations can look at how the input data can be adapted so that it can be learnt by a convolutional neural network.

2.2 Random Forest and Decision Tree Algorithms

A random forest is an ensemble method that is based on the decision tree algorithm [22]. A decision tree regressor builds a series of if/then/else statements which it then traverses to predict an output, y. Figure 1(a) shows a decision tree regressor. Each green box, known as a node, represents an if/then/else statement. Each blue box, known as a leaf, represents a classification or numeric value. The depth is the number of layers in the tree. A simple example of a decision tree regressor would be determining if a person is going to go play golf today. The first rule is, "There a government lockdown". If it is true, the blue arrow is followed to a leaf node which says, "No". If it is not true, the green arrow is followed. The next nodes rule is "The temperature is greater than 30°". If true, the blue arrow is followed, if untrue the green arrow is followed. For regression, the tree determines the best feature to split on a node by identifying which feature split minimises the variance of each leaf node. Equation 4 shows the formula for variance: μ is the sample average, X is each value in the sample and N is the number of values in a sample.

Decision trees can suffer from overfitting. Overfitting is when a model performs well on training data but performs relatively poorly on data it has not seen before [11]. This is due to the amount of specificity we look at in each level of the tree, leading to smaller sample of events that meet the previous assumptions [1]. As the depth increases, the number of samples in the node decreases so nodes with just one sample are created. This is not enough information to make a reliable prediction on. To counter this effect, the depth of the tree can be reduced. However by doing this, we increase the variance in the nodes which leads to an overall weaker predictor. What would be ideal is to reduce both bias and error due to variance.

To overcome this issue, the random forest regressor is introduced. As shown in Figure 1(b) a random forest consists of multiple decision trees that run in parallel. The result of each individual tree is aggregated and then the final prediction is made by taking an average of all the trees. Each individual decision tree is trained on a random feature set which means that, even if each tree may have a degree of overfitting, by taking an aggregate this effect is mitigated. Their ability to limit overfitting without substantially increasing error due to bias is why random forests are such powerful models.

$$\sum \frac{(X-\mu)^2}{N} \tag{4}$$

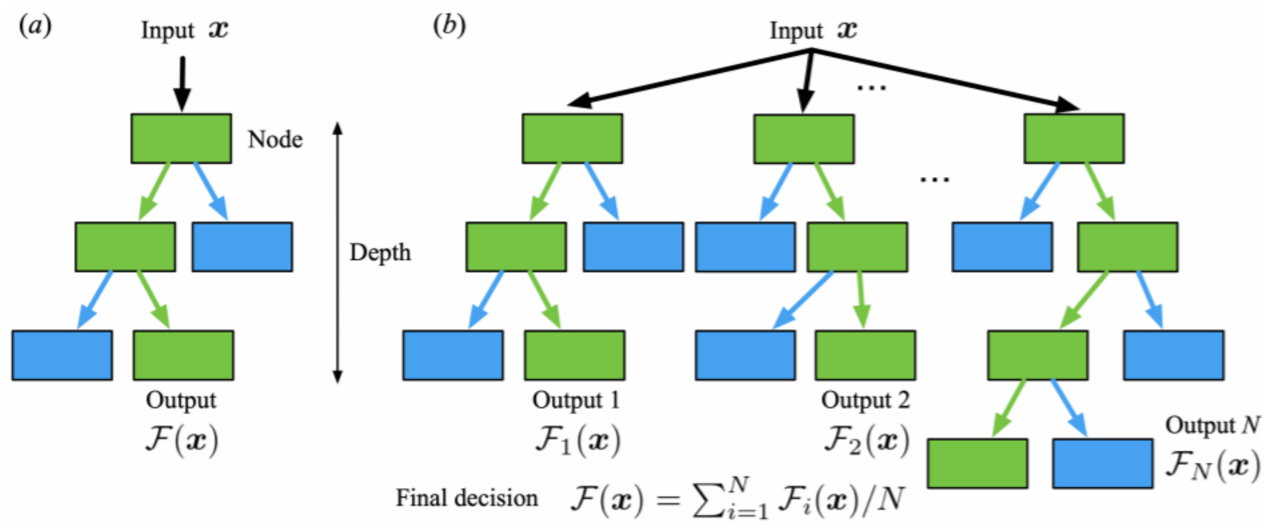


Figure 1: (a): Decision Tree Regressor (b): Random Forest Regressor

2.3 Final Model

The final model brings further complexity to the training data by increasing the feature space to include a range of Reynolds numbers. The operating conditions are $M = 0$, $Re = [5x10^6, 6x10^6]$ and the flow is viscid and incompressible. The input features are:

1. Reynolds Number
2. Angle of Attack
3. X-position along aerofoil
4. Y-position along aerofoil

Using *GridSearchCV* the decision tree and random forest algorithms were again tuned to select the parameters that would reduce RMSE.

3 Results and Discussion

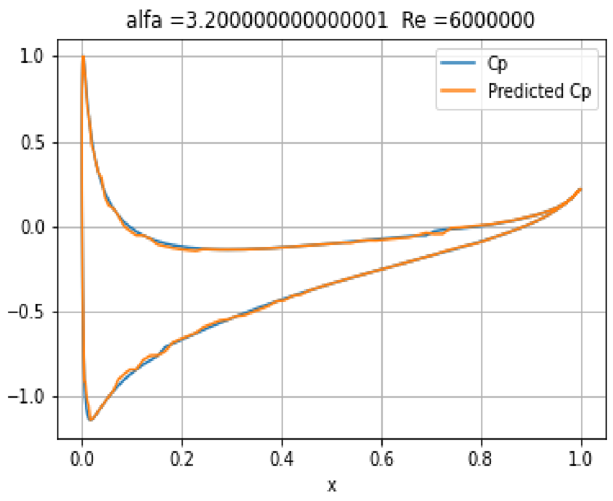
Table 3 shows the RMSE scores for the random forest and decision tree regressors. It is observed that the random forest outperforms the decision tree model by a significant margin. This is not in trend with the preliminary models, however it can be explained by the decision tree model's tendency to overfit data when a large feature space is used.

Figure 2 shows the predicted C_P distribution using the two algorithms. The decision tree regressor demonstrates overfitting at $0.8 \leq x \leq 2.2$. This is shown by the jagged response of the curve. However, the random forest algorithm manages to capture the curve extremely accurately. This result is as expected because when the feature space increases, decision tree algorithms perform poorly compared to random forest models.

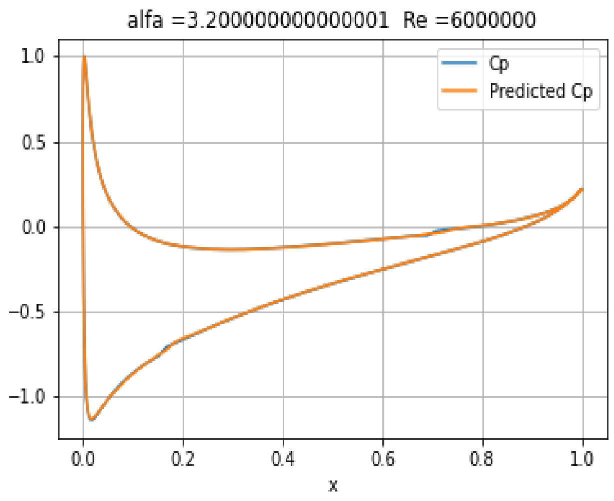
Decision Tree Regressor	Random Forest Regressor
0.046	0.027

Table 3: Final Model RMSE Scores

Figure 3 shows how the RMSE varies as angle of attack increases for each model. The general trend for both models is that error increases as α increases. This is likely due to the fact that at larger angles of attack, boundary layer separation occurs due to stall. The behaviour of the pressure becomes less predictable which is why the models struggle relative to lower angles of attack. It is also interesting to see that both models perform relatively poorly when $\alpha = 2.0°$ for $Re = 6x10^6$. The models were trained for the same angle of attack, however for a lower Reynolds number. They fail to capture the physics of increasing the Reynolds number and so they

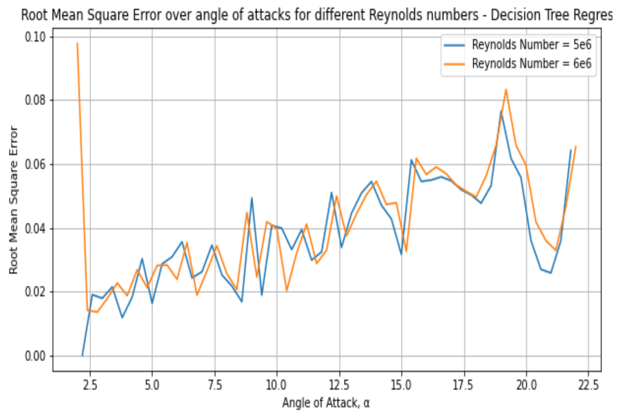


(a) Decision Tree Regressor

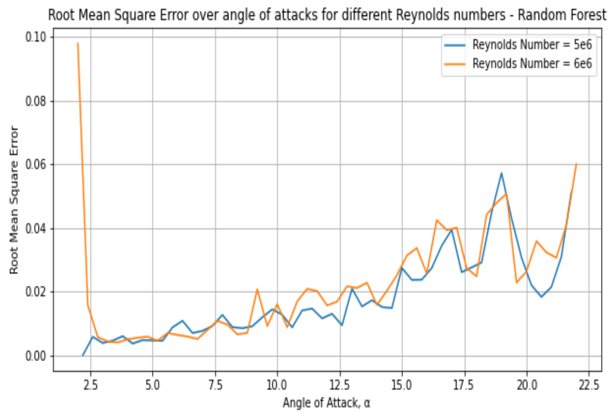


(b) Random Forest Regressor

Figure 2: Cp distribution for $Re = 6x10^6$ at $\alpha = 3.2°$

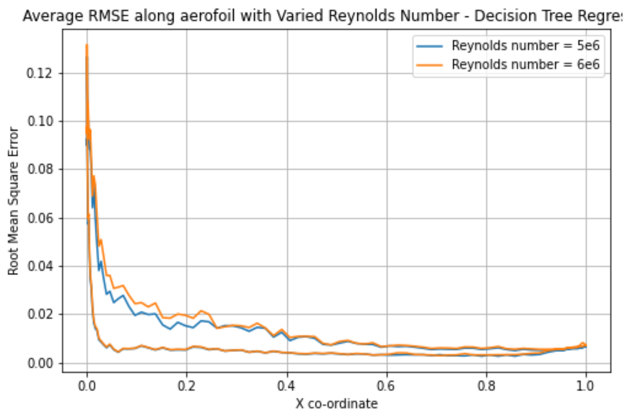


(a) Decision Tree Regressor

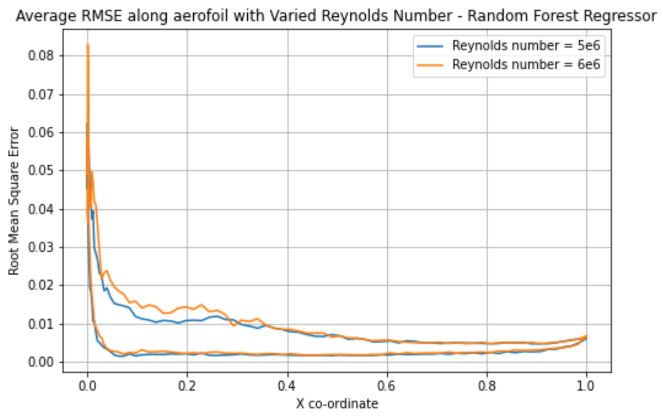


(b) Random Forest Regressor

Figure 3: RMSE over angle of attack for $Re = [5x10^6, 6x10^6]$



(a) Decision Tree Regressor



(b) Random Forest Regressor

Figure 4: RMSE over aerofoil for $Re = [5x10^6, 6x10^6]$

both output the pressure distribution for a lower Reynolds number. This is indicative of ML's limitations of performing predictions on data that the model has not been trained on.

Figure 4 shows how the error varies across the aerofoil. Both models struggle to predict values towards the leading edge of the aerofoil whilst towards the trailing edge the error decreases. This is likely due to a problem of scale. At the leading edge, the C_P values have a range of -15 to 2 across α. As α increases, the shape of the pressure distribution changes more considerably towards the leading edge. Because of this, the error in C_P at the leading edge is exacerbated. For both models, visually, the error seems to be larger for the higher Reynolds number condition. However the error is not likely to be statistically significant.

4 Conclusion

In this paper a method to determine the pressure distribution around an aerofoil using machine learning is demonstrated. The model performs well with an error of less than 0.03. This demonstrates that machine learning is a viable method to decrease the computational cost of fluid mechanics modelling. Whilst the model could interpolate well, where the model fails is when it is exposed to data outside of its training set, showing machine learning's ever present problem with extrapolation of data. This shows that it is important to train the model on a wide data set to ensure that it is able to perform well on a wide range of inputs. In future work, the model can be expanded to custom aerofoil shapes.

References

[1] Max Bramer. *Principles of data mining*, volume 180. Springer, 2007.

[2] Steve Brunton. *Machine Learning for Fluid Mechanics.* 2020.

[3] Steven L Brunton, Maziar S Hemati, and Kunihiko Taira. Special issue on machine learning and data-driven methods in fluid dynamics, 2020.

[4] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control.* Cambridge University Press, 2019.

[5] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.

[6] Kazuko Fuchi, Eric M Wolf, David Makhija, Nathan A Wukie, Christopher R Schrock, and Philip S Beran. Enhancement of low fidelity fluid simulations using machine learning. In *AIAA Scitech 2020 Forum*, page 1409, 2020.

[7] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Assessment of supervised machine learning methods for fluid flows. *Theoretical and Computational Fluid Dynamics*, pages 1–23, 2020.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[9] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[10] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. pages 481–490, 08 2016.

[11] Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

[12] Raymond M Hicks and Preston A Henne. Wing design by numerical optimization. *Journal of Aircraft*, 15(7):407–412, 1978.

[13] Eurika Kaiser, Bernd R Noack, Laurent Cordier, Andreas Spohn, Marc Segond, Markus Abel, Guillaume Daviller, Jan Östh, Siniša Krajnović, and Robert K Niven. Cluster-based reduced-order modelling of a mixing layer. *arXiv preprint arXiv:1309.0524*, 2013.

[14] Konstantina Kourou, Themis P. Exarchos, Konstantinos P. Exarchos, Michalis V. Karamouzis, and Dimitrios I. Fotiadis. Machine learning applications in cancer prognosis and prediction. *Computational and Structural Biotechnology Journal*, 13:8 – 17, 2015.

[15] M krishna, M Neelima, Harshali Mane, and Venu Matcha. Image classification using deep learning. *International Journal of Engineering Technology*, 7:614, 03 2018.

[16] Michele Milano and Petros Koumoutsakos. Neural network modeling for near wall turbulent flow. *Journal of Computational Physics*, 182(1):1–26, 2002.

[17] Josef Musil, Jakub Knir, Athanasios Vitsas, and Irene Gallou. Towards sustainable architecture: 3d convolutional neural networks for computational fluid dynamics simulation and reverse designworkflow. *arXiv preprint arXiv:1912.02125*, 2019.

[18] Michael A Nielson. *Neural Networks and Deep Learning.*

[19] Guido Novati, Lakshminarayanan Mahadevan, and Petros Koumoutsakos. Controlled gliding and perching through deep-reinforcement-learning. *Physical Review Fluids*, 4(9):093902, 2019.

[20] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[21] Mateus Dias Ribeiro, Abdul Rehman, Sheraz Ahmed, and Andreas Dengel. Deepcfd: Efficient steady-state laminar flow approximation with deep convolutional neural networks. *arXiv preprint arXiv:2004.08826*, 2020.

[22] Lior Rokach and Oded Maimon. Top-down induction of decision trees classifiers-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4):476–487, 2005.

[23] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[24] Bradley L. Whitehall and Stephen C-Y. Lu. Machine learning in engineering automation —the present and the future. *Computers in Industry*, 17(2):91 – 100, 1991. Special Issue IMS '91-Learning in IMS.

[25] Shuran Ye, Zhen Zhang, Xudong Song, Yiwei Wang, Yaosong Chen, and Chenguang Huang. A flow feature detection method for modeling pressure distribution around a cylinder in non-uniform flows by using a convolutional neural network. *Scientific Reports*, 10(1):1–10, 2020.

[26] Yao Zhang, Woong-Je Sung, and Dimitri Mavris. Application of convolutional neural network to predict airfoil lift coefficient, 2018.

[27] Beckett Yx Zhou, Nicolas R Gauger, Jeremiah Hauth, Xun Huan, Myles Morelli, and Alberto Guardone. Towards real-time in-flight ice detection systems via computational aeroacoustics and machine learning. In *AIAA Aviation 2019 Forum*, page 3103, 2019.