

Machine Learning Based Drag Coefficient Prediction For Two-Dimensional Geometries

M A S Manuranga

Department of Mechanical engineering

Faculty of engineering

University of Sri Jayewardenepura

somanthamanuranga@gmail.com

Abstract— With many potentials in the field of computational fluid dynamics, machine learning is rapidly rising to the top of the list of technologies for scientific computing. In present accurate prediction of aerodynamic properties is an important role for the design of applications that involve fluid flows. Mainly it is essential for the aerospace industry. In this paper it investigates the working of the convolutional neural network to predict the drag coefficient in laminar, low Reynolds number flow for generated two dimensional geometries. After the training aim is to optimize the efficiency of the trained convolutional neural network and increase the accuracy, decrease the validation loss of data set. Here we are mainly focusing on the random two-dimensional shape drag coefficient. To find the drag capabilities from using machine learning we use convolutional neural network for laminar low Reynolds number flows which are associated with random two-dimensional shapes. Here our main goal is to train the machine learning model with convolutional neural network and testing the results with real life shapes.

Keywords—Drag coefficient, convolutional neural network, machine learning, Neural network.

I. INTRODUCTION

Machine learning, and neural networks, have seen a new trained in popularity in recent years due to their ability to solve complex problems that were previously thought to be impossible for computers to handle. This has led to their adoption in diverse research fields, including computer vision, natural language processing, speech recognition, robotics, finance, healthcare, and many others. When using machine learning techniques less manual calculation is required while designing models. The primary benefit of computation is its ability to visualize what occurs in a model under various flow conditions. More complex models will require more processing power, which will speed up output.

When comparing 3D model simulations to 2D simulations, high performance processors are needed to handle the data and it takes longer. The major drawback of CFD in the present situation is this. The recent rapid development of machine learning techniques has created new opportunities for addressing challenging computational fluid dynamic issues. Computational fluid dynamics (CFD) is a field that involves simulating the behavior of fluids and gases using numerical methods. Neural networks have been used in various ways to assist or improve CFD computations in recent years.

The growing of the neural networks in the computational fluid dynamics area enhances different aspects of CFD simulation and analysis. When considering the neural networks of this area it can be trained to simplify the complex fluid flow problems. Instead of solving complex

equations directly machine learning based neural networks can learn. The mapping between inputs and outputs of a particular problem. This advantage allows faster evaluation of the flow behavior without the use of computational simulations. When considering the turbulent flows, it is challenging to simulate in an accurate way because of its dynamic nature. When using neural networks that can develop data driven models that can execute the turbulent behavior in a more efficient way. It can also help to reduce the computational cost also. Not only that machine learning based neural networks also can be used to optimized flow control strategies and by connecting neural networks with optimized algorithms and it is easy to find the better shape and control variables that minimize the drag, and it helps to enhance the efficiency to reach the desired objectives.

For estimating the uncertainties of computational fluid dynamic simulations neural networks can be used. By self-learning from data that known uncertainties neural networks can do the predictions of flow properties which is a advantage of making decisions and can be perform the sensitive analysis also. It is very important to identify the application of neural networks to computational fluid dynamics is still an active key area of research and there are so many improvements that are happening to explore their capabilities and their limitations. The combination of computational fluid dynamics and neural networks is giving accelerating to improve accuracy and enabling the new pathways to fluid analysis and design improvements.

exciting area of research and has the potential to significantly improve the accuracy and efficiency of CFD simulations in a wide range of applications. Predicting the drag coefficient of two-dimensional shapes is an important problem in fluid mechanics, as it can help optimize the design of various objects, such as aircraft, automobiles, and ships. Machine learning can be used to develop predictive models that can accurately estimate the drag coefficient of arbitrary two-dimensional shapes. Here the random shapes were generated from using softr.io online shape generator.



Fig 1: Randomly generated two dimensional shapes.

One approach to developing a machine learning-based drag coefficient prediction model is to use a dataset of shapes with known drag coefficients to train a neural network. The

dataset was got by research that was done from the Jonathan viquerat and Elie hachemen.[3] The neural network can then be trained to learn the relationship between the shape geometry and the drag coefficient. The novelty of this research was increasing the predicted drag coefficient values accuracy and decrease the validation loss of the data set using convolutional neural network.

II. OBJECTIVES

In this research the main objective is to make a model from using machine learning model to predict the drag coefficient of random two-dimensional geometry. This is the main objective of this research. The sub objectives are as follows.

1. Develop a convolutional neural network (CNN) for drag prediction.
2. Train the model with high accuracy and lower validation loss.
3. Compare tabulated drag coefficients with predicted drag coefficients.
4. Predict drag coefficient for a given custom image.

There are some major applications based on two-dimensional drag prediction in engineering and physics. Predicting generated drag forces on airfoils and wings is important in aircraft design. From considering the flow characteristics and properties around two dimensional airfoils experts can understand the drag coefficient and optimize the shape for improved aerodynamic performance. When considering the automobile industry, drag prediction is valuable in designing shapes of the cars to reduce the aerodynamic drag.

To improve fuel efficiency engineers can optimize the shape of automobiles by using this kind of neural network drag prediction method. There are many fields that the drag will crucially effect. In sports such as swimming and cycling reducing drag will affect to increase the performance. From considering flow around 2D shapes and models of sports equipment like swim kits, sky suits can help designers optimize their shape with minimizing the drag. Not only architects and civil engineers can utilize 2D drag predictions to give better performance of design buildings and structures to decrease the wind resistance. This will majorly affect the tall buildings and bridges where high drag forces can lead to structural failures of the buildings and bridges. Designing creative shapes and AI based design will be improved from using this kind of drag prediction method.

III. METHODOLOGY

When an two dimensional object is placed inside a free stream flow, two forces act on that object. Here we only consider about the drag force and its coefficient only. The magnitude of the force varies with the geometry as well as the flow conditions such as velocity (v), Reynold's number (Re).

To represent those two forces dimensionless parameters which is known as drag coefficient (C_D) are used.

$$C_D = \frac{2F}{\rho U^2 A}$$

C_D = Coefficient of drag

A = Reference area / Projected area

ρ = Density of fluid

U = Fluid speed

F = Drag force

Reynold's number is defined as follows,

$$R_e = \frac{\rho U L}{\mu}$$

U = Freestream velocity.

L = Characteristic length.

μ = Dynamic viscosity of fluid.

ρ = Density of fluid.

In this research, the first goal is to create the data set before the training of the machine learning model. Here convolutional neural network was trained from Reynolds number constant at 10. When considering the predicting drag coefficient for two dimensional images for low Reynolds number is a task. In the low Reynold number region flow is more laminar and drag coefficients are predictable and can be solved from analytical formulas. Most of the cases neural networks are used for study about the complex flow patterns when the relationship between images features and drag coefficient values is not easily captured by analytical formulas.

A. Convolutional Neural Network Architecture

Convolutional neural networks (CNNs) are a type of neural network commonly used in computer vision tasks such as image recognition. They are particularly well-suited for these tasks because they can learn hierarchical representations of images. In a CNN, the convolutional layers use local receptive fields to learn features from small patches of an image, and the weights for these features are shared across the entire image. This allows the network to learn spatially invariant features that can be used to recognize objects regardless of their position in the image. By stacking multiple convolutional layers, the network can learn more abstract and complex features that are able to distinguish between different classes of objects.

When working with random images as our input we must use convolutional layers instead of fully connected layers. Because the fully connected layers most of the times use with less common image recognition tasks. Here each

convolutional kernel which is inside the convolutional neural network will be used for the purpose of extracting special features from the input random shape image. During the forward processing each kernel involve with successfully match with input image to generate an activation map which shows the response of the kernel at every spatial position in the given input. This flow of the process will allow the network to learn spatially invariant features that can be used to identify the suitable and correct object and patterns that comes up with the input image. Most of the times convolutional layers are followed by pooling layers. Pooling layer major advantage is to reduce the spatial size of the feature maps generated by the convolutional layers while retaining the important spatial information. And it can decrease the number of degrees of freedom in the neural network and make efficient procedure. Not only that pooling layers can spread the initial data throughout the successive convolutional layers. When it comes to CNN it gives some of the major advantages such as shared weights, efficient detection of spatial features of the given image. For the selected data they were split into three categories. Those are training set, validation set, test set. In training data set is used to train the model and update its parameters. In validation set is used to monitor the accuracy of the model during the training and make some decisions about the hyperparameters and the network architecture. Finally, the test set is used to evaluate the final performance of the model that one did not train data after the training is complete. Here we need to remove the dataset from overfitting problem. If overfitting happens, the model performs well on the training data and poorly on unseen data. Another factor is the validation and test sets should not overlap with each other. If it happens this could bias the evaluation of the machine learning model performance. For the first 100 shapes were put into following neural network and those are the basic layers of CNN.

B. Activation Function

In this convolutional neural network, it is an essential component, and it gives nonlinear into the model and allowing the model to learn complex patterns and relationships that already in the dataset. Each neuron in a neural network applies an activation function to its input image and output is then passed to the next layer of neurons.

In recent machine learning techniques use Rectified linear unit (ReLU) use for hidden layers in neural networks for classification tasks due to its complexity and computational efficiency. In this training process we must reduce the loss function also. The loss function measures the difference between the predicted output of the neural network and the true output. The learning process objective is to minimize the difference between which is achieved by updating the networks parameters. Finally, the process of adjusting the parameters is repeated over many training inputs or they are also called epochs. Here we do this process until the loss function is minimized to an acceptable level. At this point we can decide to the neural networks can be used to make predictions on new unseen data as well.

As the first stage of training section, we used 100 input data. The low amount of data in our training section can lead to overfitting the neural network model. A small network with few parameters may not have enough strength to learn the underlying patterns in the data and its features accurately and may perform poorly on both training and validation set. As a implementation of neural network we use this amount data and after the neural network making the input data will increase up to more than 12000 inputs. In further model training stages, we should limit the overfitting. From gathering more data, reducing the size of the network can lead network to lower overfitting. For the training section find out the optimum number of epochs for this network is 15 from repeatedly training.

For the neural network implementation stage, we use inbuild libraries which is already with TensorFlow. Here used architecture consist of a pattern of repeated convolutional and pooling layers with varying number of filters in each layer. (These filters or kernels are used to extract features from the input image. They are small matrices that slightly varies over the input image and perform element wise multiplication and summation to produce feature map). Here includes tow convolutional layers with initial layer including 8 filters. And the network is terminated with two dense layer of size 16 and last layer used for outputting predicted drag coefficient using linear activation function.

C. Data Preprocessing

For the machine learning image processing task there are several important libraries that were used here. For the data manipulation panda's library was used. TensorFlow was used which is a popular open-source library for machine learning and NumPy helps to large multidimensional arrays and mathematical functions. In the data processing section 12000 images and drag coefficient values were obtained in the previous research paper. [9]



Fig 2: Generated two dimensional shapes.

The data set was split into two categories which are trained set and the test set. For the trained set 10200 images were trained using convolution neural networks and 1800 data were used as the test set. Then the 20% of trained data set was split to create the validation dataset. The purpose of this validation set is to test the convolutional architecture which was used in training is correct or wrong. After that every image was rescaled 0 – 1 range to increase the processing speed of the data set. Then the images were converted into greyscale to reduce the complexity of the images Because greyscale images have a single set channel which they have low memory requirements and computational power compared to the RGB images that consist with three

channels. From converting RGB images into greyscale ones the task can focus on the overall brightness of the image which easily extract the features of a particular image.

For the images loading and training, validation and testing were done by using special functions. For efficient data loading data generators were used and those are load image data from specified data frames. These data generators are fetch the images in batch wise and allow the model to process and reducing memory requirements and allow for the efficient training. When processing the images as batches model can vary parallel processing and GPU acceleration causes to more faster training and predicting times.

For the architecture for this 2D image drag prediction task was plays a major role here. For the regression task convolutional neural network architecture was on grayscale images. For the input layer model performs grayscale images with dimensions of 256*256 pixels. After the feature extraction multiple convolutional layers were applied from the input images. Each layer of these layers applies set of filters (16,32,64) to the input by providing ReLU activation function. To reduce the spatial dimensions of the output Max pooling layers were used in this architecture. For the output layer was flattened into a 1D vector to prepare the fully connected layers. To learn the complex relationships between extracted features dense layers were added to the architecture. Finally, the fully connected layer only has a single unit and here uses linear activation function for predicting the regression target value. For the final step of this architecture the model was combined with the Aam optimizer and mean squad error loss function.

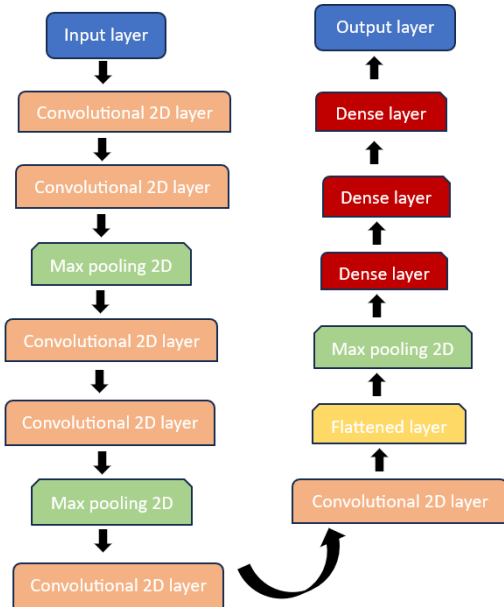


Fig 3: Tested finalized CNN architecture with high accuracy.

IV. RESULTS AND DISCUSSION

After generating random shapes all the drag coefficient values were set up in the drive folder. After that the CNN model training was done using TensorFlow libraries. The following shows the tested convolutional neural network architecture. Here we used 3×3 conv,16 layer, it means that

convolutional layer with 16 filters and each filter is a small matrix of size 3×3 . When we use multiple convolutional layers, it helps to enable the network to learn hierarchical representation of input image. Here each filter will make a separate feature map. Here we use a 2×2 pooling layer and it refers to layer that reduces spatial dimensions of the feature map.

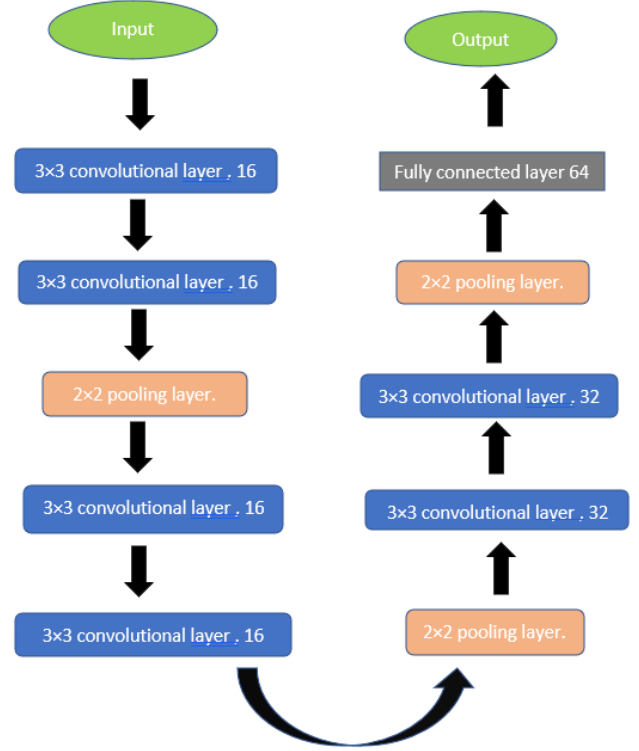


Fig 4: Initially used CNN architecture for drag coefficient prediction.

For the data preprocessing part, the data frame was read and tabulates them as shape id value. After that the image path was assigned to a address. The whole data frame was split into 80 for the training and 20 for the testing. For validation 20% was selected from the training dataset. From validation the main purpose is to tune the accuracy of the network. Then rescale the image pixel values between 0-1 to improve the processing speed of the network and reduce the RGB color image to a grayscale image also to improve the processing speed of the neural network. Then the training was done according to the above figure CNN architecture.

Table I : Relative error table for randomly selected shapes

Shape	Exact drag coefficient	Predicted Drag coefficient	Relative error
Shape 66	2.54185	2.27895	10.34 %
Shape 89	2.18827	2.30818	5.4%
Shape 62	2.1859	2.06021	5.7%
Shape 55	2.11403	2.0505	3%

The main drawback of this network was, we only used only 100 data and those are not enough to train the CNN network in better accuracy. Here our relative error results are between 3% - 10%. These values are somewhat higher and in the remaining works we need to increase data amount to get

better accuracy of the predicted drag coefficients. After first training process the CNN training process was done with 12000 images and its drag coefficient values with the same architecture. After training all the images consider shape_11825 for testing the architecture. The actual drag coefficient value for the shape is 1.8433. Here considers the predicted value with actual value of the shape and calculated its relative error. The goal of this comparison is to find out the optimum number epochs with lowest validation loss.

Table II: Shape_11825 tested with various number of epochs.

Number of epochs	Predicted Drag coefficient	Validation loss	Relative error (%)
6	2.0350	0.3117	10.39
7	2.12320	0.3090	15.18
8	2.071	0.3080	12.35
9	2.0258	0.3129	9.9
10	2.0496	0.3101	11.19
11	2.1088	0.3084	14.4
15	2.0731	0.3085	12.46
20	2.0964	0.3081	13.73
30	2.1400	0.3100	16.09

This shape was included in the test set of the data frame. According to the table when number epochs is 9 the predicted drag coefficient value was given lowest relative error. But this value was high. As the result of this comparison got to know that the CNN architecture need to develop to get more accurate results.

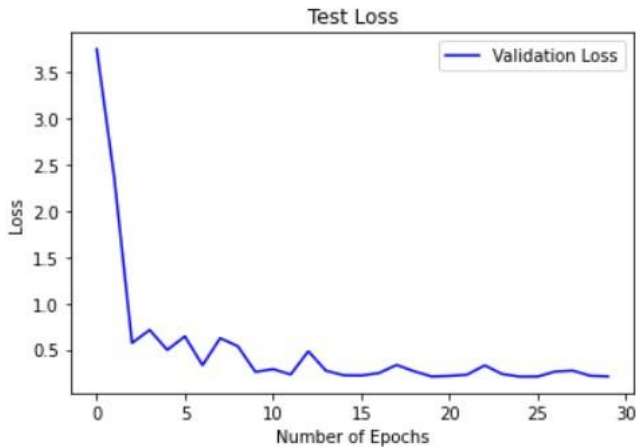


Fig 5: Test loss for initial CNN architecture

According to the above graph, 28 number of epochs were given the best results. But the predicted value not much closer to actual value with the trained architecture. After that CNN architecture was improved to the above figure X. Here included more hidden layers and Max pooling layers to the architecture to obtain a better accuracy of the results. The objective of adding more Max pooling layers is to extract more spatial features from the input images.

Table III: Shape_10441 With optimized CNN architecture

Number of epochs	Predicted Drag coefficient	Validation loss	Relative error (%)
6	2.65	0.0025	1.5
7	2.7763	0.0032	2.32
8	2.7416	0.0022	1.04
9	2.6716	0.0026	1.5
10	2.2.65	0.0038	2.33
15	2.7062	0.0017	0.26

Shape_10441 actual drag value is 2.71326 with the Reynolds number at 10. For the optimized CNN architecture, the predicted values were given the best results. When the number of epochs was near to 15 this architecture was given 0.26% relative error and it was well predicted result with the actual value. Validation loss also decreases largely when compared to the initial architecture.

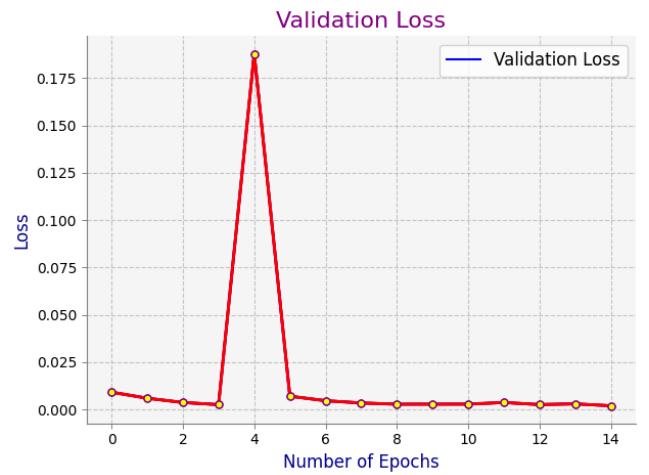


Fig 6: Validation loss graph with number of epochs

According to the above graph when the training is set near to the 15 number of epochs it gives the best predicted values. Because of that we can prove that the data set was not overfit with the training set. As the final output the data set was greatly behave with the 15 number of epochs with lowest validation loss. When considering the average relative error vs batch size graph for a CNN task with 12000 images, we can compare the performance from average relative error vs batch size graph. By analyzing batch size and relative errors we can identify which batch size performs the lowest error and it helps obtain better performance. Optimal batch size can be found out from this graph. Not only that, the hyperparameter tuning and understanding of the model behavior is another advantage of relative error and batch size graph.


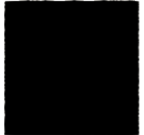

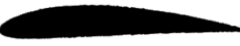

For the figure 4 architecture Average relative error and batch size graph was drawn in figure 7. According to the above graph average relative error was reduced when the batch size range was 200 – 250.



Fig 7: Average relative error vs batch size graph

For the result comparison custom images were evaluated which was based on real life shapes. The optimized CNN network was performed accurate results as follows.

Table IV: Exact and predicted drag values for real life shapes.

Shape	Actual drag coefficient	Predicted drag coefficient	Relative error (%)
	1.589	1.61	1.32
	1.764	1.765	0.056
 NACA 4424	1.263	1.298	2.77
 NACA 4412	1.121	1.125	0.35
 NACA 0018	1.186	1.190	0.33

V. CONCLUSION

Machine learning techniques are widely used present in computational fluid dynamics to improve accuracy and performances better than the traditional computations. Here

this research was done to predict the drag coefficients for two dimensional geometries using convolutional neural networks. The developed CNN architecture was used to extract the features of the images. Finally, the predicted values form the neural network and simulation values were compared and relative error also compared. To develop the accuracy of the neural network it is necessary to increase the number of data points and tune the hyperparameters of the neural network. At the first stage basic CNN architecture was developed and compare the predicted and actual results along with the relative error. Then CNN architecture was developed, and prediction was done in an accurate way.

VI. ACKNOWLEDGEMENT

“This research was supported by the Science and Technology Human Resources Development Project, Ministry of Education, Sri Lanka, funded by the Asian Development Bank (Grant No . CRG/R3 18 and CRG/R3 17).

REFERENCES

- [1] S. F. Y. Z. P. W. J. Z. Tianyi Liu, "Implementation of training convolutional neural network," 2015.
- [2] w. z. ., j. k. ., y. l. linyang zhu, "Machine learning methods for tubulence modeling in subsonic flows around airfoild," 2018.
- [3] E. H. Jonathan Viquerat, "A supervised neural network for drag prediction of arbitray 2D shapes in laminar flows at low reynolds number," 2021.
- [4] j. k. ., a. ., v. ., i. g. josef musil, "Towards sustainable architecture 3D convolutional neural networks for computational fluid dynamics simulation and reverse design workflow," 2016.
- [5] k. f. ., k. z. ., a. G. .. N. ., k. f. masaki morimoto, "Convolutional neural networks for fluid flow anlysis toward effective metamodeling and low dimensionalization," 2021.
- [6] a. h. ., a. k. Matthias eichinger, "surrogate convolutional neural network models for steady computational fluid dynamics simulations," 2020.
- [7] Y. w. ., x. y. y. ., z. h. c. ., w. t. w. ., n. a. Ming yu wu, "fast prediction of flow field around airfoils based on deep convolutional neural network," 2015.
- [8] s. j. j. ., v. o. s. ., a. A. .. a. ., j. v. g. t. ., b. g. Paulo alexandre costa rocha, "Deep neural network modeling for CFD simulations banchmarking the fourier neural operator on the lid driven cavity case," 2022.
- [9] y. w. ., a. m. ., j. l. ., x. c. xiaohui yan, "CFD-CNN modeling of the concentration field of multiport buoyant jets," 2022.