A DEEP LEARNING METHODOLOGY FOR THE FLOW FIELD PREDICTION AROUND AIRFOILS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

CIHAT DURU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
MECHANICAL ENGINEERING

SEPTEMBER 2021

Approval of the thesis:

A DEEP LEARNING METHODOLOGY FOR THE FLOW FIELD PREDICTION AROUND AIRFOILS

submitted by **CIHAT DURU** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Mehmet Ali Sahir Arıkan
Head of Department, **Mechanical Engineering** _____

Assist. Prof. Dr. Özgür Uğraş Baran
Supervisor, **Mechanical Engineering, METU** _____

Assist. Prof. Dr. Hande Alemdar
Co-supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Mehmet Haluk Aksel
Mechanical Engineering, METU _____

Assist. Prof. Dr. Özgür Uğraş Baran
Mechanical Engineering, METU _____

Assoc. Prof. Dr. Hacer Yalım Keleş
Computer Engineering, Hacettepe University _____

Assoc. Prof. Dr. Sinan Kalkan
Computer Engineering, METU _____

Assist. Prof. Dr. Onur Baş
Mechanical Engineering, TED University _____

Date: 07.09.2021

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Cihat Duru

Signature :

ABSTRACT

A DEEP LEARNING METHODOLOGY FOR THE FLOW FIELD PREDICTION AROUND AIRFOILS

Duru, Cihat
Ph.D., Department of Mechanical Engineering
Supervisor: Assist. Prof. Dr. Özgür Uğraş Baran
Co-Supervisor: Assist. Prof. Dr. Hande Alemdar

September 2021, 96 pages

This study aims to predict flow fields around airfoils using a deep learning methodology based on an encoder-decoder convolutional neural network. Neural network training and evaluation are performed from a set of computational fluid dynamics (CFD) solutions of the 2-D flow field around a group of known airfoils at a wide range of angles of attack. Reynolds averaged Navier-Stokes (RANS)-based CFD simulations are performed at a selected Mach number on the transonic regime on high-quality structured computational grids. The results of these simulations are utilized as training data set. For better shape learning, a distance map is generated from airfoil shape and provided to the algorithm at data locations of the flow quantities, i.e., pressure coefficient, Mach number, relative to the airfoil shape. The predictive ability of the model is scrutinized both qualitatively and quantitatively. The flow features associated with the transonic effects and the angle of attack variation, such as the shock waves and the flow separation, are well predicted. The results indicate that the presented model provides remarkably good flow field predictions at a fraction of the computational time of CFD simulations. The predicted flow field allows the compu-

tation of the aerodynamic coefficients, providing an accurate and fast airfoil selection tool for aircraft designers.

Keywords: Deep learning, airfoil aerodynamics, transonic flow

ÖZ

KANAT ETRAFINDAKİ AKIŞ ALANININ TAHMİNİ ÜZERİNE BİR DERİN ÖĞRENME METODOLOJİSİ

Duru, Cihat
Doktora, Makina Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Özgür Uğraş Baran
Ortak Tez Yöneticisi: Dr. Öğr. Üyesi. Hande Alemdar

Eylül 2021 , 96 sayfa

Bu çalışmada kanat etrafındaki akış alanını öngören kodlayıcı-çözücü evrişimsel sinir ağı tabanlı bir derin öğrenme metodu amaçlanmaktadır. Sinir ağı modelinin eğitimi ve değerlendirmesi bilinen bir grup kanat etrafında, geniş bir hücum açısı aralığında 2-boyutlu hesaplamalı akışkanlar dinamiği (HAD) çözümlerinden elde edilmiş akış alanları ile gerçekleştirilmiştir. Reynolds ortalamalı Navier-Stokes tabanlı HAD benzetimleri, ses geçişi rejiminde seçilen bir Mach sayısında yüksek kalitede blok-yapılı çözüm ağları üzerinden yapılmıştır. Benzetimlerin sonuçları eğitim için veri seti olarak kullanılmıştır. Kanat şekillerinin sinir ağı modeli tarafından daha iyi öğrenilebilmesi için mesafe haritaları elde edilerek algoritmaya verilmiştir. Mesafe haritasında verilerin konumuna göre basınç katsayısı, Mach sayısı gibi akış parametreleri de algoritmaya sağlanmıştır. Modelin öngörü kabiliyeti hem nicel hem nitel olarak detaylı incelenmiştir. Ses geçişi ve hücum açısı değişiminin akış özelliklerine etkileri, şok dalgaları ve akış ayrılması başarılı bir şekilde yakalanmıştır. Ayrıca, öngörülen akış alanı verisi üzerinden aerodinamik katsayılar hesaplanmıştır. Bulgular, sunulan mo-

delin HAD benzetimlerinin hesaplama sürelerinin çok altında, dikkate değer şekilde iyi akış alanı öngörebildiğini göstermektedir. Öngörülen akış alanı, aerodinamik katsayıların hesaplanmasına izin vermekte, hava taşıtı tasarımcıları için hızlı ve doğru bir kanat seçim aracına imkan sağlamaktadır.

Anahtar Kelimeler: Derin öğrenme, kanat aerodinamiği, transonik akış

To my family

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisors, Dr. Özgür Uğraş Baran and Dr. Hande Alemdar, for sharing their knowledge with me and always contributing with positive thinking. I am very thankful for all the valuable discussions during my Ph.D. Also, I would like to thank Prof. Dr. Haluk Aksel and Dr. Onur Baş for their guidance throughout my research.

I am also grateful to TÜBİTAK ULAKBİM for the computing resources.

Last but not least I owe special thanks to my mother, father and sisters for their love and support. My family has always been there for me.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

1D	1 Dimensional
2D	2 Dimensional
3D	3 Dimensional
ANN	Artificial Neural Network
AWT	Accuracy with Threshold
CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network
DF	Distance Field
ELU	Exponential Linear Unit
GAN	Generative Adversarial Network
GP	Gaussian Process
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MSE	Mean Squared Error
NN	Neural Network
RANS	Reynolds Averaged Navier-Stokes
ReLU	Rectified Linear Unit
SDF	Signed Distance Field
UIUC	University of Illinois at Urbana-Champaign

CHAPTER 1

INTRODUCTION

1.1 Motivation of the Study

The design of a wing with suitable aerodynamic characteristics constitutes a crucial step in aircraft design. The initial design stage involves experimenting with many different airfoil sections and often results in a custom airfoil profile at the final design. The design of airfoils is a lengthy process that requires for experimental, computational, and theoretical efforts. Various families of airfoils [1] have been developed for different applications, e.g., aircraft, sailplanes, propellers. The starting point of the airfoil selection for a new application design such as a wing, rotorcraft, wind turbine, and others is usually selecting the most appropriate airfoil from those families. This decision is made by considering the mission requirements based on, but not limited to, flight regime (i.e., supersonic, transonic, subsonic, and flight altitude), lift requirement, stall limits, drag limitations, structural limits and maneuvering [2].

A complete wing design starts with the calculation of the required wing area using the lift coefficient and drag-to-lift ratio of the selected airfoil such that the lifting surface fulfills the lift requirement of the aircraft from the forward flight to high lift conditions [2]. However, a modern aircraft wing is much more complex than a simply swept airfoil. Geometric parameters include taper ratio(s), sweep angle(s), aspect ratio, wing twist, dihedral angle, fuselage connections, and wingtip details. Often, the final wing design involves a transition between multiple airfoil profiles.

The airflow around a wing, therefore, is much more complicated than that is determined around a simple infinite length airfoil. The planform result in a more complex flow to manipulate lift distribution along the span of the wing, the twist distribu-

tion of a wing affects the angle of attack, and wingtip and fuselage connections can also generate additional complexity. As a result, the 3-D flow field becomes notably more complex, and the analysis of the flow features around a wing is unexpectedly demanding. Furthermore, it might be challenging to fulfill all the requirements by selecting an available airfoil since those requirements strongly depend on the airfoil shape, Reynolds number, Mach number and the angle of attack. The airfoil shape has an exhaustive list of geometric parameters such as thickness, camber, location of maximum thickness and maximum camber, the leading edge radius and others. All these parameters [3] must be taken into account since they determine the airfoil performance. Therefore, the design engineer should experiment with many configurations to find an optimum airfoil design that meets all the requirements. As a part of the solution and selection procedure, wind tunnel tests might be costly and often time-consuming, especially in the early design stages. It is preferable to conduct experiments at later stages of the design in order to fully and accurately assess the aerodynamic performance of the aircraft. Quick design tools for aircraft often fall short in predicting performance. There are several reasons for this. These methods often rely on extended 1-D models, simplified analytical and empirical methods, or the utilization of simplified (often irrotational and inviscid) 2-D solutions of airfoils. Recent numerical solution techniques allow very accurate results for both viscous and compressible flows. On the other hand, computational fluid dynamics (CFD) simulations require significant computational power, and the amount of computing power may be tremendous if higher-order methods are employed at this stage. Therefore, they are not feasible for experimenting with thousands of geometric configurations.

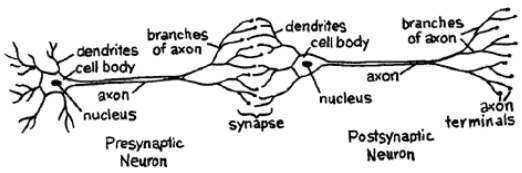
Recent advances in machine learning show promise for a fast alternative approach instead of expensive experiments and time-consuming simulations [4]. Learning from data offers new opportunities for developing computational methods in research fields, such as fluid dynamics, which constantly accumulate a large amount of data. Therefore, a machine learning approach may relieve the massive amount of work from CFD simulations, at least in the initial design stages of airfoils. This thesis is intended to develop a deep learning approach to approximate the flow field around the airfoils and demonstrate the applicability of a machine-learning-based methodology for fluid flow problems. This study is the crucial first step to develop a complete wing

design tool.

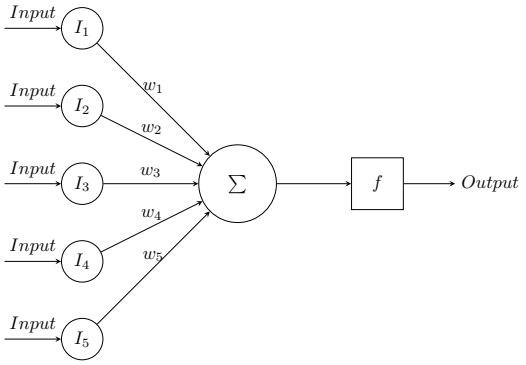
1.2 Artificial Neural Networks

Neural network is a particular field of machine learning, which aims to make feasible the intelligent behavior of the brain in processing data for the use of various applications such as object detection[5], text/image translation[6, 7], image/speech recognition[8, 9], forecasting in finance[10], etc.

Neural networks aim to simulate the learning mechanism of organisms by inspiring the biological neural network of their nervous system. The neurons are connected through synapses which are the regions between axons and dendrites, as illustrated in Figure 1.1a. The incoming signals referred to as spikes from many other neurons are transmitted to one another along those synapses. The synaptic connections in the nervous system describe how learning takes place in organisms and serve as a source of inspiration to an artificial neuron, as depicted in Figure 1.1b. An artificial neuron or node computes a function, f, of all weighted inputs and generates an output. The output then serves as an input to another neuron in the subsequent layer.



(a) Biological neural network[11]



(b) A schematic of an artificial neuron

Figure 1.1: An illustration of a neural network

An artificial neural network is composed of stacked layers, where each layer consists of neurons, as illustrated in Figure 1.2. The first layer is the input layer in which the input data is introduced to the network, while the last layer is the output layer which produces the final result. The layers in between the input and the output layers are

called hidden layers. The number of hidden layers is called the depth of the network. The primary aim of a neural network is to map the input to the output. For instance, in an object detection task, images containing the objects serve as the inputs and the labels, e.g., car, apple, as the output. Learning takes place by changing the weights of the connected neurons. Therefore, the input-output pairs as the training data are fed into the neural network to adjust the weights and provide feedback about how well the neural network makes predictions about the labels. After successively adjusting the weights by feeding the neural network model with the training set of input-output pairs, the model provides accurate predictions. The model will also eventually gain the generalization capability to make predictions for unseen inputs.

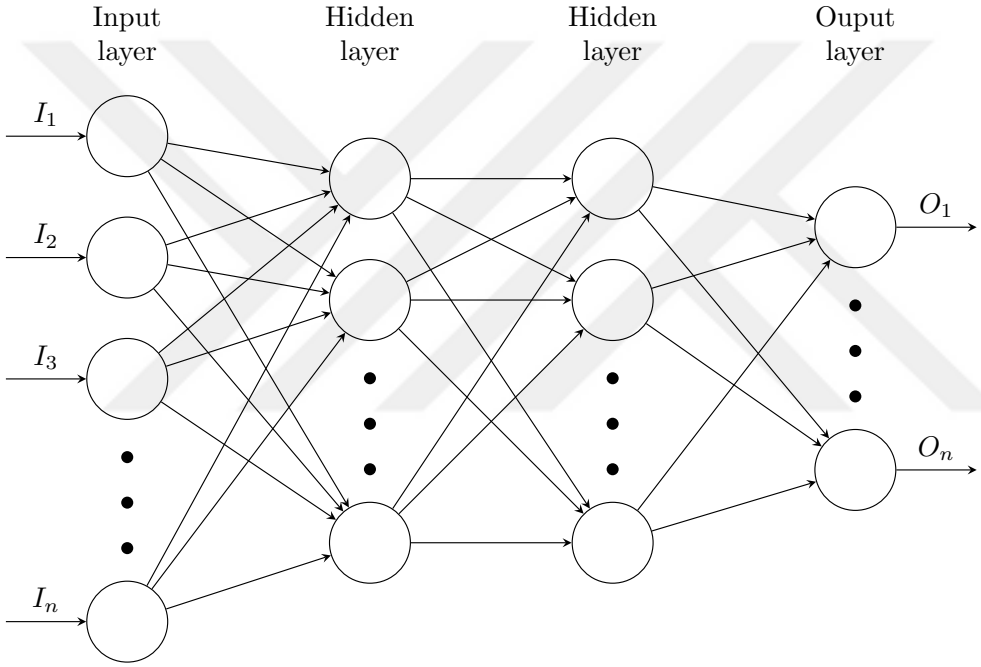


Figure 1.2: An artificial neural network

1.3 Literature Survey

In recent years, deep learning has gained popularity in various research fields due to its promising ability to learn from data. More particularly, neural network applications in fluid mechanics appeared in a growing number of studies. These studies contributed to the field in different directions, such as accelerating the existing CFD solvers, improving turbulence models, aerodynamic coefficient prediction, shape op-

timization, and flow field reconstruction. Earlier, Faller and Schreck [12] have addressed possible aeronautics problems in which neural networks may provide a practical approach. As computing resources have been developed and new neural network algorithms have also been introduced, (such as convolutional neural network (CNN) [13]), more complex network models are generated. Brunton et al. [14] discussed the current developments with emerging opportunities in machine learning techniques for fluid mechanics problems.

Machine learning has assisted turbulence modeling, especially Reynolds Averaged Navier-Stokes (RANS) models. Milano and Koumoutsakos [15] utilized neural networks to reconstruct the near-wall field in a turbulent channel flow using flow fields provided by direct numerical simulations. Zhang and Duraisamy [16] presented a new Gaussian process (GP) regression method for the functional construction of closure modeling. They compared the performance of this method with conventional GP and an artificial neural network (ANN) model. The results showed promise for the data-driven turbulence modeling. Also, several studies have targeted the prediction of the Reynolds stresses and turbulent eddy viscosity. Ling et al. [17] proposed a NN which learns Reynolds stress anisotropy tensor from high-fidelity data. The results showed that the NN model was more accurate than the RANS models on two different test cases. Singh et al. [18] developed a machine learning-based methodology that assists the Spalart-Allmaras turbulence model. Zhu et al. [19] studied to construct turbulent eddy viscosity mapping for subsonic flow around an airfoil by ANN. Liu and Fang [20] proposed a machine-learning-based turbulence modeling strategy. The model is tested in turbulent channel flows at low Reynolds numbers. Maulik et al. [21] proposed a NN model to improve RANS modeling by predicting turbulent eddy viscosities. The proposed model was fed with potential flow solutions as inputs. These studies presented promising results for the machine-learning-based turbulence modeling. Novati et al. [22] presented a multi-agent reinforcement learning methodology to detect spatio-temporal patterns in order to establish sub-grid scale physics.

There is also an increasing interest in utilizing machine learning to accelerate CFD solvers. Yang et al. [23] proposed an ANN to speed up the projection step of the Poisson equation solution. Similarly, a modification in the solution of the Euler equation was employed by Tompson [24] by replacing the pressure projection step with a

learned projection to produce fast and divergence-free flow fields. Wandel et al. [25] proposed a physics-constrained training approach that allows neural network models to learn subsequent time step states from previous time step data.

Predictive models for estimating the surface pressure distribution and aerodynamic coefficients have also been investigated. Thirumalainambi and Bardina [26] built a NN to predict aerodynamic coefficients. They have developed two different architectures for longitudinal (lift, drag and pitching moment) and lateral (side force, rolling moment and yawing moment) aerodynamic coefficients, respectively. A comparative study was also conducted in this study to optimize NN architectures based on training data set size and selection of activation function. Kurtulus [27] developed a NN model to predict the unsteady aerodynamic force coefficients of flapping motion. Ignatyev and Khrabrov [28] studied unsteady aerodynamic characteristics in an extended angle-of-attack range by utilizing an NN, fed with experimental data. Miyanawala and Jaiman [29] presented a convolutional neural network (CNN) based model to predict drag and lift coefficients for unsteady flows over bluff body shapes at low Reynolds number (Re). The predictive capabilities of CNN models for aerodynamic coefficients around airfoils were also studied and presented in the studies of Yilmaz and German [30] and Yuan et al. [31]. Zhang et al. [32] proposed various CNN architectures with different layouts to predict the lift coefficient of the airfoils. In their study, a multilayer perceptron architecture is considered as the baseline model, and the predictions from both multilayer perceptron and CNN models were compared. Viquerat and Hachem [33] presented a CNN model for the drag prediction of arbitrary 2D shapes at $Re = 10$. Hui et al. [34] utilized deep learning in order to predict surface pressure distribution over airfoils.

NNs have also been shown to be an effective tool for inverse design problems, such as shape optimization studies. Sun et al. [35] developed an ANN-based inverse design method for airfoils and wings. Yilmaz and German [36] built deep CNNs to study the relationship between the airfoil geometry and the surface pressure distribution. Their tool predicts the y-coordinates of airfoil geometries at specified x-coordinates based on the pressure coefficient data. Atalay et al. [37] performed an optimization study for the design parameters of the slatted wind turbine blade to maximize the lift-to-drag ratio by utilizing fuzzy linear regression. Renganathan et al. [38] studied

aerodynamic shape optimization by using a gradient-based optimizer coupled with a trained deep neural network model.

Several studies have been conducted to predict flow fields using deep learning. Guo et al. [39] developed several CNN models to predict laminar velocity fields over 2D and 3D domains in one of the first studies. They trained their models by feeding their domains with signed distance fields and simple binary representations separately. The results show that the trained models with signed distance field representations outperform binary representations. They achieved significant speedup with their neural network model compared to CFD simulations. A CNN model was proposed by Jin et al. [40] that predicts velocity fields around a circular cylinder for which the input is the pressure field around the cylinder. The model is trained with the low Reynolds number dataset from 60 to 1100. Fukami et al. [41] developed a machine learning approach to reconstruct a high-resolution flow field from low-resolution input images for laminar and turbulent flows. They applied their model to the 2D cylinder wake at $Re = 100$ to demonstrate of their model. Also, they have constructed the velocity and vorticity fields from low-resolution data for 2D decaying homogeneous turbulence problem. A convolutional encoder-decoder model similar to [39] was developed by Bhatnagar et al. [42] to predict pressure and velocity fields around airfoils. CFD simulations are carried out at various Reynolds numbers to establish the dataset for three airfoil shapes. Chen et al. [43] studied to predict velocity and pressure fields around arbitrary 2D shapes in laminar flow using a CNN-based U-net type architecture. They utilized a dataset consists of random shapes built using Bezier curves. The velocity and pressure fields of each shape are obtained by numerical simulations at $Re = 10$. Lee and You [44] studied the prediction of unsteady flow fields around a circular cylinder using several models based on generative adversarial networks (GANs) and CNNs. Flow fields at future time steps are predicted by utilizing the data from the previous time steps. Physical loss functions based on conservation of mass and momentum are proposed. The effects of physical loss functions are also investigated. Sekar et al. [45] presented an approach to predict laminar steady flow fields around airfoils. In this study, CNN was applied to parameterize airfoil shapes. Then, the extracted geometrical parameters are utilized as input to the multilayer perceptron network for the flow field approximation model. The flow field over an airfoil is ap-

proximated as a function of Reynolds number, airfoil geometry and angle of attack. A U-net-based network architecture was applied to predict velocity and pressure fields around airfoils for incompressible flow by Thuerey et al. [46]. Their model is trained with the flow solutions of different airfoil shapes for a variety of Reynolds numbers.

1.4 Objectives and Contributions

In this thesis, a series of studies are initiated to develop a novel approach to the design of modern aircraft lifting surfaces. We present a deep-learning-based airflow prediction model that can be used in modern airfoil performance prediction tools. A neural network architecture is constructed on a database of high-accuracy solutions of the flow field around different airfoil shapes at a wide range of angle of attack. Using this model, we can produce flow fields over an airfoil in a very short time. Therefore, our method is more accurate and much faster than the simplified-model solutions around airfoils of various shapes.

This thesis is aimed to evaluate the capability of the neural network architecture in various flow conditions at the same flow regime, emphasizing very different gradient distribution of flow-related parameters. Such considerable variation within the flow field database may be achieved by selecting the Mach number. The Mach number, M, is the ratio of the flow velocity to the local speed of sound. The novelty of this thesis relies on the flow regime. We have selected a relatively high subsonic Mach number ($M = 0.7$), but still below the transonic regime ($0.8 < M < 1.2$). This is because at transonic flow regimes, the shocks are observed very often around the airfoils. Therefore, the solutions around such airfoils will result in high gradients at the flow field most of the time. Instead, we have selected a Mach regime where the pressure gradients can change rapidly, and the cases involving shocks do not dominate (but are still represented in) the model database.

Also, the present study serves a proof of concept for the implementation of deep learning methodology into fluid flow problems, including discontinuities such as shock waves and separation.

The computational fluid dynamics (CFD) simulations are performed for the required

database generation. The requirements for this solver are a high-quality compressible flux scheme capable of handling high gradients, and a suitable turbulence model for external flow that would give accurate enough estimations of the separation location and the angle of attack. Moreover, these calculations should be completed in a reasonably short time, considering the immense amount of computational time required for database generation.

1.5 Outline of the Thesis

The remainder of the thesis is as follows.

Chapter 2 introduces the structure of a convolutional neural network with the necessary elements. Then, the details about the training of neural networks are also given.

In Chapter 3, database generation is explained, starting with introducing governing equations. The flow solver and the computational method are presented later.

The details of the encoder-decoder convolutional neural networks are briefly explained in Chapter 4. The design procedure of the proposed network architecture is introduced in this chapter.

Flow solutions obtained with the proposed network model are discussed in Chapter 5. The performance of the model is evaluated with the experiments carried out in this chapter.

Finally, conclusions and recommendations for future studies are given in Chapter 6.

CHAPTER 2

CONVOLUTIONAL NEURAL NETWORKS

This chapter provides an overview of convolutional neural networks and the required elements to build neural network models. The training of neural networks is also discussed in detail later.

2.1 Convolutional Neural Networks

Convolutional neural network (CNN) is a type of neural network designed for analyzing image-like structured data. CNNs have become dominant, especially for computer vision tasks such as image classification and object recognition.

CNN is a multi-layered feedforward neural network composed of an input layer, output layer and stacked middle hidden layers consist of convolutional layers which are the characteristic layer of a CNN. CNNs are very successful in capturing textures, patterns or features (e.g., curves, gradients) in the input image due to the convolutional layers. Each hidden layer extracts features from the input data as the input data is processed through the network. While the low-level features (e.g., color, direction) of the image are captured in the earlier layers, the network captures high-level features (e.g., edges) in the later layers.

2.1.1 Convolutional Layer

The convolutional layer is the key element of a CNN, which performs a special type of linear operation called convolution for feature extraction. The convolution operation is shown schematically in Figure 2.1. A matrix of weights called convolutional fil-

ter/kernel, travels over the input tensor and performs an element-wise product. Then the sum of the element-wise product is assigned to the output tensor, which is called feature map. An arbitrary number of feature maps can be extracted by repeating the same procedure with different filters to obtain different input tensor characteristics. The filter behaves as a feature extractor and highlights specific features. The neural network learns the weights in the filter matrices during the training process.

Several hyperparameters that need to be set before the neural network training: the number of filters, the size of filters, and the stride. The number of filters defines the depth of the feature maps, while the size of filters defines the receptive field. The stride is the number of elements by which the filter moves after each operation. Also, when the filters do not fit the input data, zero-padding becomes necessary to increase the input size by adding zeroes on the each side of the input boundaries. The output size of the feature map can be calculated as

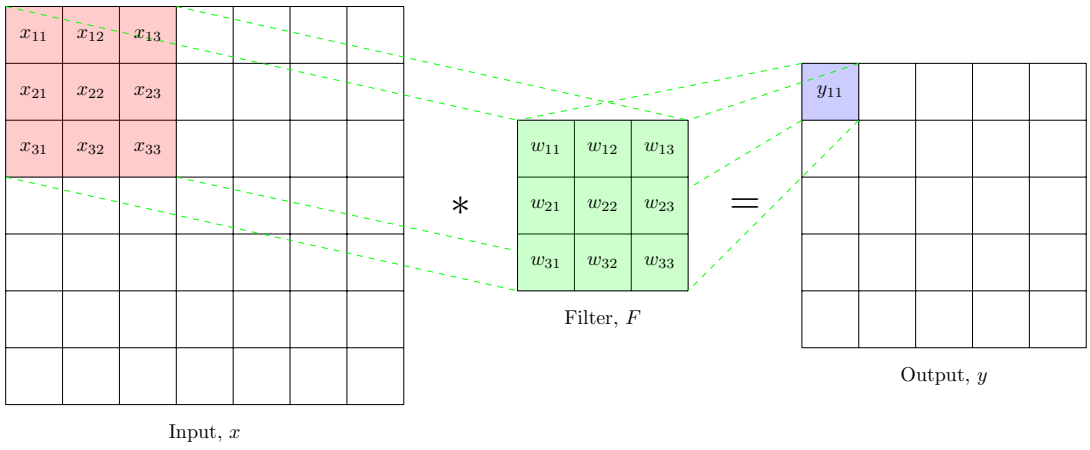
$$O = \frac{I - F + 2P}{S} + 1 \tag{2.1}$$

where O denotes output size, F denotes filter size, P denotes the size of zero-padding and S denotes the stride. For instance, in the Figure 2.1; $I = 7$, $F = 3$, $P = 0$ and $S = 1$, which gives $O = 5$.

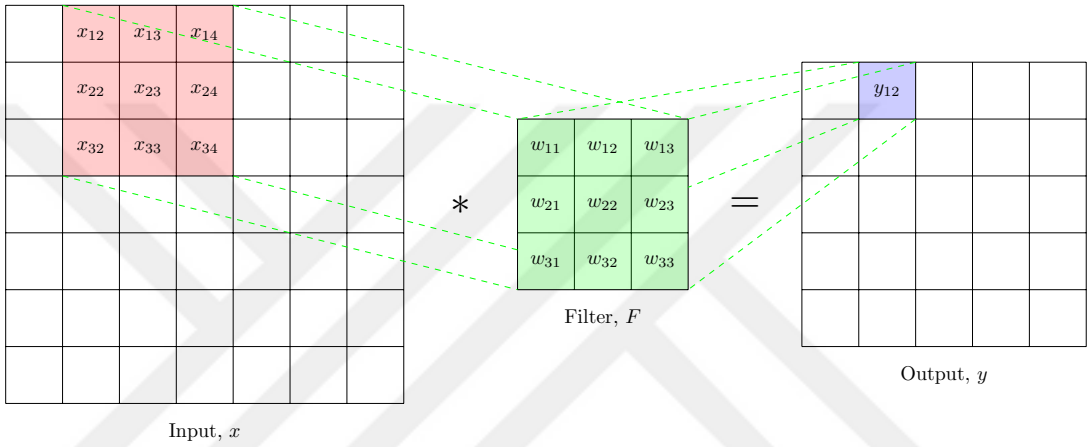
2.1.2 Activation Function

After input data is processed through a convolutional layer, the output is passed through an activation function. The activation function is a mathematical equation that adds non-linearity into the neural network. Otherwise, the neural network may reduce to one linear layer in which consecutive linear operations take place. The most common functions are sigmoid (Eq. 2.2), hyperbolic tangent functions (Eq. 2.3), rectified linear unit (ReLU) (Eq. 2.4), parametric ReLU (Eq. 2.5) and exponential linear unit (ELU) (Eq. 2.6).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$



(a)



(b)

Figure 2.1: The schematic representation of convolution operation

$$\sigma(x) = tanh(x) \tag{2.3}$$

$$\sigma(x) = max(0, x) \tag{2.4}$$

$$\sigma(x) = max(\alpha x, x) \tag{2.5}$$

$$\sigma(x) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases} \tag{2.6}$$

These alternatives have both advantages and disadvantages regarding gradient characteristics, output range or computational cost. An activation function should have some desirable features. It should be differentiable, and its gradient should not tend towards zero, which is called the vanishing gradient problem in the backpropagation process. It should be computationally inexpensive since activation functions are applied after each layer, and it should also provide the neural network to converge quickly. Commonly used activation functions are plotted in Figure 2.2.

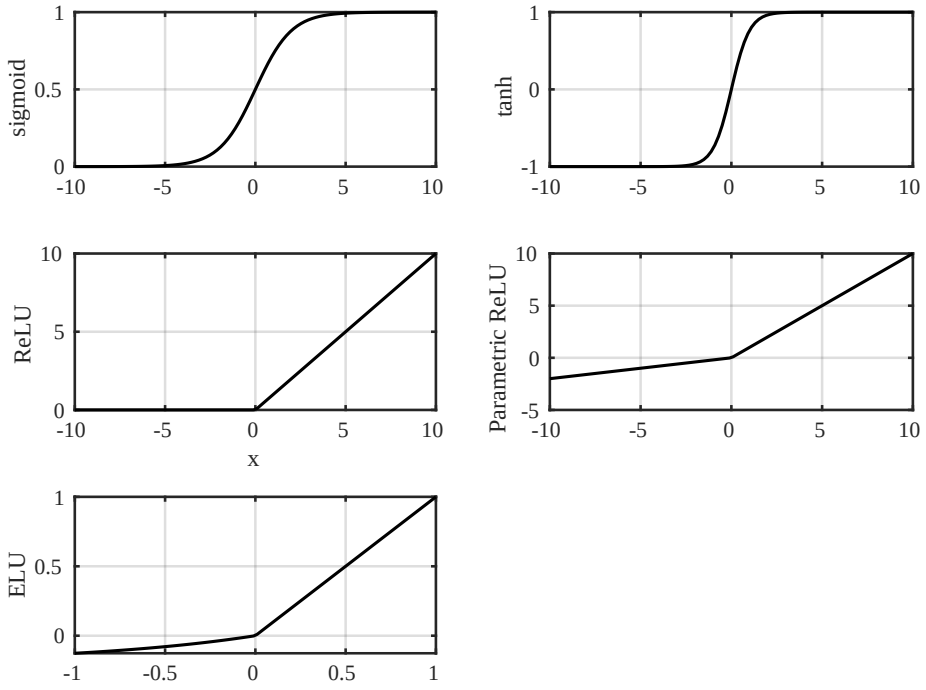


Figure 2.2: Commonly used activation functions

Sigmoid and hyperbolic tangent functions were used frequently in earlier times. However, both activation functions can make the neural network more susceptible to vanishing gradient problems during training. For very small or very large input values, the sigmoid and hyperbolic tangent functions saturate very early and the gradient of these functions are failed through the first layers to propagate the information particularly in deep networks. ReLU [47] and its extensions are the most commonly used activation functions. They are much effective in preventing vanishing gradient problem than sigmoid and hyperbolic tangent functions. Although ReLU is easy

to implement and computationally efficient, it does not activate the backpropagation algorithm when the input values approach to zero or become negative similar to vanishing gradient problem. To overcome this problem, the leaky ReLU ($max(0.1x, x)$) [48] is proposed to serve a slope in the negative region. Then, the slope of the negative region is generalized and studied as parametric ReLU. In Figure 2.2, α is set to 0.2. Similar to parametric ReLU, another variant called ELU [49] as given in Eq. 2.6 also allows negative values. ELU may provide faster learning performance on networks with more than five layers [49].In Figure 2.2, α for ELU is set to 0.1.

2.1.3 Loss Function

The input-output (ground truth) pairs are fed into the neural network for the training process. The loss function (also called cost function, objective function, error function, etc.) is used to measure the difference between the neural network output and the ground truth utilized during the training process. The training process aims to minimize the loss function and adjust the learnable parameters, i.e., weights and biases. There can be various options for the design goal of the network model. For a classification problem, hinge loss or cross-entropy and its variants can be preferred. For a regression problem , mean absolute error, mean squared error loss or smooth mean absolute error functions can be used. The objective in this thesis can be considered as a regression problem. Therefore, the loss functions that are suitable for only regression models are addressed in this section. Mean squared error (MSE) (Eq. 2.7) and mean absolute error (MAE) (Eq. 2.8) are the most commonly used loss functions depending on the problem.

$$MSE = \frac{1}{N}\sum_{k=1}^{N}(y_k - \tilde{y}_k)^2 \tag{2.7}$$

$$MAE = \frac{1}{N}\sum_{k=1}^{N}|y_k - \tilde{y}_k| \tag{2.8}$$

Here, y and $\tilde{y}$ are the ground truth and the model output, respectively. N is the number of data points. MSE estimates the mean of squared distances between the

ground truth and the prediction. The squaring of the difference results in penalizing the larger differences and may provide a quickly converged model. MSE calculates the average of the absolute difference between the ground truth and the prediction. If outliers dominate the ground truth variable, MAE is more robust to outliers than MSE. However, its gradient is the same for both small and large differences, and it results that the model shows slow convergence behavior.

2.1.4 Training a neural network

Training is an iterative process that aims to find the best mapping of inputs to outputs by updating weights progressively. First, the neural network produces a result by feeding the training data through the network using the current weights. The loss function is calculated for the model output based on the ground truth. Then, the derivative of the loss function on each weight and bias are computed in the back-propagation stage to tell us how sensitive the loss function is with respect to those variables.

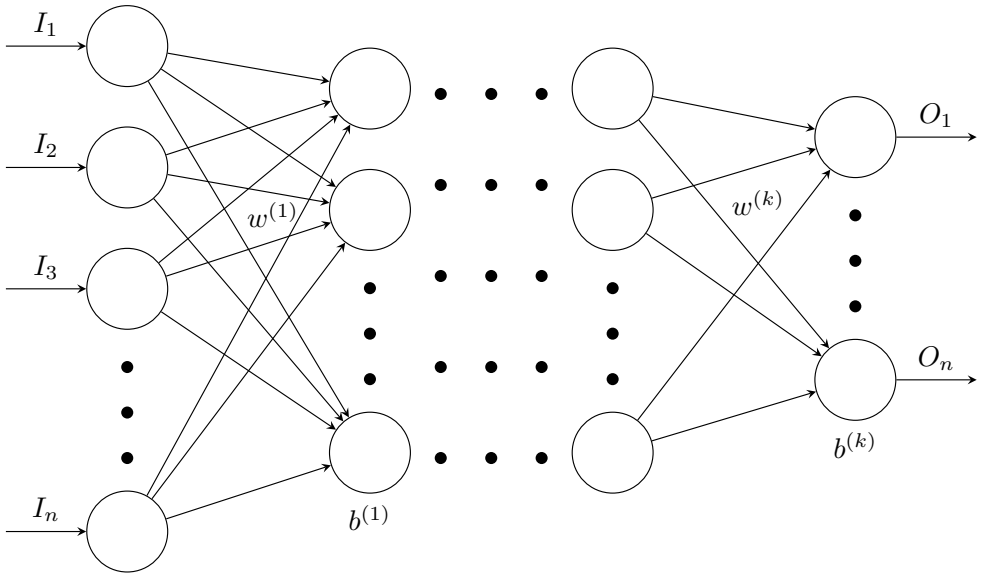


Figure 2.3: An artificial neural network

As an illustration, the output $h^{(1)}$ at each hidden node in the first hidden layer for the

given artificial neural network in Figure 2.3 is computed as

$$z^{(1)} = w^{(1)}I + b^{(1)} \tag{2.9}$$

$$h^{(1)} = \Psi(z^{(1)}) \tag{2.10}$$

where $I = \{I_1, I_2, ..., I_n\}$ is the input data, $w^{(1)}$ and $b^{(1)}$ are the set of weights and biases of each connection between the input layer and the first hidden layer. Ψ is the activation function. Similarly, the output of a hidden layer is computed by using the output of the previous hidden layer as input.

$$z^{(l)} = w^{(l)}h^{(l-1)} + b^{(l)} \tag{2.11}$$

$$h^{(l)} = \Psi(z^{(l)}) \tag{2.12}$$

Finally, the output of the network $O = \{O_1, ..., I_n\}$ is calculated as:

$$O = w^{(k)}h^{(k-1)} \tag{2.13}$$

After the neural network outputs, the loss function is computed. An optimization algorithm is utilized in order to minimize the loss function, which is given as

$$L(f(x;\theta), y) \tag{2.14}$$

where L is the loss function, $f(x;\theta)$ is the model output, O, and f denotes the neural network function. y is the ground truth, x and θ denote the input, I, and the learnable parameters, i.e., weights and biases, respectively. In the backpropagation stage, the gradients of the loss function with respect to each weight and bias are computed. The gradients indicate how much the weights and biases need to be changed during the

training to minimize the loss function. The gradient of a single weight in a layer can be computed by using chain rule

$$\frac{\partial L}{\partial w_{i,j}^{(l)}} = \frac{\partial L}{\partial O}\left[\sum_{\mathcal{P}}\left\{\frac{\partial O}{\partial h^{(k)}}\prod_{n=l}^{k-1}\frac{\partial h^{(n+1)}}{\partial h^{(n)}}\right\}\right]\frac{\partial h^{(l)}}{\partial w_{i,j}^{(l)}} \tag{2.15}$$

where $w_{i,j}^{(l)}$ denotes the weight of the connection between the previous and the current neurons i and j at layer l. P represents all individual paths from $h^{(l)}$ to $h^{(k)}$. The local gradient term in Eq. 2.15 can be written as

$$\frac{\partial L}{\partial O}\sum\left\{\frac{\partial O}{\partial h^{(k)}}\prod_{n=l}^{k-1}\frac{\partial h^{(n+1)}}{\partial h^{(n)}}\right\} = \frac{\partial L}{\partial h^{(l)}} \tag{2.16}$$

We know $h^{(l)} = \Psi(z^{(l)})$ and $z^{(l)} = w^{(l)}h^{(l-1)} + b^{(l)}$. The partial derivative of a neuron's output with respect to the weight can be estimated as follows:

$$\frac{\partial h^{(l)}}{\partial w_{i,j}^{(l)}} = \Psi'(z^{(l)})h^{(l-1)} \tag{2.17}$$

Therefore, the gradient of the loss function with respect to the weights can be written as:

$$\frac{\partial L}{\partial w_{i,j}^{(l)}} = \frac{\partial L}{\partial h^{(l)}}\Psi'(z^{(l)})h^{(l-1)} \tag{2.18}$$

Similarly, by knowing $\partial z^{(l)}/\partial b^{(l)} = 1$, the gradient of the loss with respect to the bias can be written as:

$$\frac{\partial L}{\partial b_{i,j}^{(l)}} = \frac{\partial L}{\partial h^{(l)}}\Psi'(z^{(l)}) \tag{2.19}$$

The gradient descent algorithm is the most popular optimization method in order to minimize the objective functions in machine learning problems, as given in Eq. 2.14. It is an iterative algorithm and uses the first derivative of the objective function to obtain which direction the parameters need to change at each iteration and how much

they respond to the update from the previous iteration. The step size that controls the change of the parameters at each iteration with respect to the loss gradients is determined by defining the learning rate, α. Update rule of parameters can be written as

$$\theta^t = \theta^{t-1} - \alpha \nabla_\theta L(\theta) \tag{2.20}$$

where t denotes the iteration. Then, the momentum method [50] is implemented to the gradient descent algorithm in order to accelerate the convergence and dampen the oscillations in the learning process by remembering the previous update as

$$\nu^t = \gamma \nu^{t-1} + \alpha \nabla_\theta L(\theta) \tag{2.21}$$

$$\theta^t = \theta^{t-1} - \nu^t \tag{2.22}$$

Here, $\nu^0 = 0$, γ is the momentum hyperparameter which is set typically to 0.9 or close to 1 and $\alpha > 0$ is the learning rate, typically 0.01, 0.1.

2.1.5 Adaptive Moment Estimation (Adam) Optimization Algorithm

Different types of optimization algorithms based on gradient descent algorithm have been proposed. These algorithms are particularly focused on the setting of the learning rate. Gradient descent is very sensitive to the learning rate. Whereas low learning rate tends to slower convergence, setting a high learning rate may result in divergence. Improvements to the gradient descent are made mostly on the adaptive learning rate. The most commonly used algorithms are Adagrad [51], Adadelta [52], RmsProp [53] and Adam [54].

Among all the extensions of gradient descent optimization algorithms, Adam combines the benefits of both Adagrad and RmsProp. While Adagrad works well with sparse gradients, RmsProp works well in non-stationary settings. Also, Adam utilizes the average of the second moments of the gradients instead of updating the parameters

based on the average of the first moments as in RmsProp. The update rule is based on an exponential moving average of the gradient, and the average squared gradient with hyperparameters controls the decay rates of these moving averages. The update algorithm is as follows:

$$\theta^t = \theta^{t-1} - \nu^t \tag{2.23}$$

$$\nu^t = \alpha \frac{\hat{r}^t}{\sqrt{\hat{p}^t} + \epsilon} \tag{2.24}$$

where typical settings for α and ϵ are 0.001 and 10^{-8}. The bias corrected first and second moments are

$$\hat{r}^t = \frac{r^t}{1 - \gamma_1^t} \tag{2.25}$$

$$\hat{p}^t = \frac{p^t}{1 - \gamma_2^t} \tag{2.26}$$

where γ_1^t and γ_2^t denotes γ_1 and γ_2 to the power t. Typical settings are $\gamma_1 = 0.9$ and $\gamma_2 = 0.999$. The first and second moments are

$$r^t = \gamma_1 r^{t-1} + (1 - \gamma_1) g^t \tag{2.27}$$

$$p^t = \gamma_2 p^{t-1} + (1 - \gamma_2) g^{t^2} \tag{2.28}$$

where $g^t = \nabla_\theta L^t(\theta^{t-1})$ and g^{t^2} indicates the element-wise square. The initial first and second moments are $r^{t=0} = 0$ and $p^{t=0} = 0$.

In practice, optimization algorithms perform each update using a number of samples rather than the whole data set. The training data set can be divided randomly into batches to speed up the learning process. Then, the objective function can be computed by taking the average over only each batch. However, the use of batch in

training provides a trade-off between the stability, convergence speed and memory requirement. The generalization is usually better by utilizing small batches than large batches. However, the standard deviation of the mean is inversely proportional to the $\sqrt{n}$, where n is the number of samples in the batch. Therefore smaller batch sizes result in a high variance in the gradient estimations. This situation can lead to the need for small learning rates to prevent instabilities in the training and the convergence of the training takes more epochs and memory.

2.1.6 Batch Normalization

The gradients of the objective function with respect to the learnable parameters can sometimes be vanished due to successive multiplications of small numbers during backpropagation and do not provide enough information on how to update the learnable parameters. To address this issue, batch normalization [55] can be applied as a layer before activation functions by replacing each batch, B, with $\gamma\mathbf{B} + \beta$. Here,

$$\mathbf{B} = \frac{B - \mu}{\sigma} \tag{2.29}$$

where

$$\mu = \frac{1}{m}\sum_{i} B_i \tag{2.30}$$

$$\sigma = \sqrt{\frac{1}{m}\sum_{i}(B_i - \mu_i) + \epsilon} \tag{2.31}$$

ϵ is a small positive number, i.e., 10^{-8} added to provide numerical stability. γ and β provide restoring the power of the network by scaling and shifting the normalized value by setting γ to standard deviation and β to mean when needed.

CHAPTER 3

DATABASE GENERATION

This chapter provides the procedure of the data preparation required to train the proposed neural network model. The required data is obtained from computational fluid dynamics (CFD) simulations. This chapter is organized as follows: First, governing equations are introduced. Then, the details of the computational method with the flow solver, boundary and initial conditions are explained. The validation of the computational method is discussed later.

3.1 Governing Equations

Conservation of mass, momentum and energy equations for compressible viscous flows are given by

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u_i)}{\partial x_i} = 0 \tag{3.1}$$

$$\frac{\partial (\rho u_i)}{\partial t} + \frac{\partial (\rho u_i u_j)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} \tag{3.2}$$

$$\frac{\partial (\rho e_t)}{\partial t} + \frac{\partial (\rho h u_j)}{\partial x_j} = \frac{\partial}{x_j}(u_i \tau_{ij} - q_j) \tag{3.3}$$

where ρ, u_i, p, e_t, h and T are density, velocity field, pressure, total energy per unit mass, total enthalpy and temperature, respectively. The equations are closed by using an equation of state, which provides the relation between pressure, density and

temperature. The ideal gas equation is given by:

$$p = \rho RT \tag{3.4}$$

where the temperature is determined from the internal energy, e, which is given by

$$e = \frac{RT}{\gamma - 1} \tag{3.5}$$

In this relation, R is the gas constant, equal to 287.1 J/kgK for air and γ is the specific heat ratio, equal to 1.4 for air. The total energy is determined as

$$e_t = \frac{1}{2}u_i^2 + e \tag{3.6}$$

The total enthalpy is defined as:

$$h = e_t + \frac{p}{\rho} \tag{3.7}$$

The shear stress tensor is given as

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij} \right) \tag{3.8}$$

where the dynamic viscosity, μ is determined by using Sutherland's law in the following form:

$$\mu = \frac{1.458 \times 10^{-6} T^{3/2}}{T + 110.4} \tag{3.9}$$

where T is in Kelvin. The heat flux is given by Fourier's law,

$$q_j = -\mathtt{k}\frac{\partial T}{\partial x_j} \tag{3.10}$$

where $\mathtt{k}$ is the thermal conductivity.

In order to model turbulent flow, considering the high computational cost associated with resolving the smallest scale turbulent eddies, Reynolds-averaged Navier-Stokes (RANS) equations are used. Favre decomposition to the instantaneous Navier-Stokes equations are solved. Favre decomposition principle is applied to the instantaneous quantities as

$$\xi = \widetilde{\xi} + \xi'' \tag{3.11}$$

where $\widetilde{\xi}$ is the mean part of the instantaneous quantity, ξ, and ξ'' is the fluctuating part of ξ. The mean part is obtained by

$$\widetilde{\xi} = \frac{1}{\bar{\rho}} \lim_{T\to\infty} \frac{1}{T} \int_t^{t+T} \rho\xi dt \tag{3.12}$$

where $\bar{\rho}$ denotes the Reynolds-averaged density given as

$$\bar{\rho} = \lim_{T\to\infty} \frac{1}{T} \int_t^{t+T} \rho dt \tag{3.13}$$

The Reynolds averaging is also applied for the pressure, and the final form of the Favre-averaged mass, momentum and energy equations using appropriate assumptions are given as

$$\frac{\partial\bar{\rho}}{\partial t} + \frac{\partial(\bar{\rho}\widetilde{u_i})}{\partial x_i} = 0 \tag{3.14}$$

$$\frac{\partial(\bar{\rho}\widetilde{u_i})}{\partial t} + \frac{\partial(\bar{\rho}\widetilde{u_i}\widetilde{u_j})}{\partial x_j} = -\frac{\partial\bar{p}}{\partial x_i} + \frac{\partial\tau_{ij}}{\partial x_j} \tag{3.15}$$

$$\frac{\partial(\bar{\rho}\widetilde{e_t})}{\partial t} + \frac{\partial(\bar{\rho}\widetilde{h}\widetilde{u_j})}{\partial x_j} = \frac{\partial}{x_j}\left(\widetilde{u_i}\tau_{ij} - q_j\right) \tag{3.16}$$

These equations appear as similar to the instantaneous equations (Eq. 3.1-3.3). However, the turbulence parameters are introduced within these equations. Stress tensor

extended by the Reynolds stresses, $-\bar{\rho}\widetilde{u_i''u_j''}$ is calculated using Boussinesq's eddy viscosity assumption given as

$$\tau_{ij} = (\mu + \mu_T)\left(\frac{\partial \widetilde{u}_i}{\partial x_j} + \frac{\partial \widetilde{u}_j}{\partial x_i} - \frac{2}{3}\frac{\partial \widetilde{u}_k}{\partial x_k}\delta_{ij}\right) - \frac{2}{3}\bar{\rho}k\delta_{ij} \tag{3.17}$$

where μ_T is the eddy viscosity and k is the turbulent kinetic energy. Turbulent kinetic energy is defined by

$$k = \frac{1}{2}\widetilde{u_k''u_k''} \tag{3.18}$$

The total energy is given as

$$\widetilde{e}_t = \widetilde{e} + \frac{1}{2}(\widetilde{u}_k\widetilde{u}_k + \widetilde{u_k''u_k''}) = \widetilde{e} + \frac{1}{2}\widetilde{u}_k\widetilde{u}_k + k \tag{3.19}$$

The heat fluxes extended by the turbulent transport of heat. They are calculated by

$$q_j = -(\mathtt{k} + \mathtt{k}_T)\frac{\partial \widetilde{T}}{\partial x_j} \tag{3.20}$$

where $\mathtt{k}_T$ is the turbulent thermal conductivity given as

$$\mathtt{k}_T = c_p\frac{\mu_T}{Pr_T} \tag{3.21}$$

where c_p is the specific heat coefficient at constant pressure and Pr_T is the turbulent Prandtl number. Pr_T is taken as 0.9 for air. In order to close Reynolds-averaged Navier- Stokes (RANS) equations, a turbulence model is needed. Spalart-Allmaras turbulence model [56] is used. It allows for accurate enough predictions of turbulent flows with adverse pressure gradients while being robust and converging fast. The

Spalart-Allmaras turbulence model is written as follows

$$\begin{aligned}\frac{\partial \rho\tilde{\nu}}{\partial t} + \frac{\partial \rho u_j \tilde{\nu}}{\partial x_j} =& \rho c_{b1}\tilde{S}\tilde{\nu} - \rho c_{w1} f_w \left(\frac{\tilde{\nu}}{y}\right) + \frac{\rho}{\sigma}\frac{\partial}{\partial x_k}\left[(\nu+\tilde{\nu})\frac{\partial \tilde{\nu}}{\partial x_k}\right] \\ &+ \frac{\rho c_{b2}}{\sigma}\frac{\partial \tilde{\nu}}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k}\end{aligned} \tag{3.22}$$

where turbulent eddy viscosity, $\mu_T = f_{v1}\rho\tilde{\nu}$. Coefficients and relations are given as

$$\begin{aligned}&c_{b1} = 0.1355, \qquad c_{b2} = 0.622, \qquad c_{v1} = 7.1, \\ &\sigma = 2/3, \qquad c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{(1+c_{b2})}{\sigma}, \\ &c_{w2} = 0.3, \qquad c_{w3} = 2, \qquad \kappa = 0.41, \\ &f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \qquad f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}}, \\ &f_w = g\left[\frac{1+c_{w3}^6}{g^6 + c_{w3}^6}\right]^{1/6}, \qquad \chi = \frac{\tilde{\nu}}{\nu}, \\ &g = r + c_{w2}(r^6 - r), \qquad r = \frac{\tilde{\nu}}{\tilde{S}\kappa^2 y^2}, \\ &\tilde{S} = S + \frac{\tilde{\nu}}{\kappa^2 y^2} f_{v2}, \qquad S = \sqrt{2\Omega_{ij}\Omega_{ij}}\end{aligned} \tag{3.23}$$

3.2 Flow Solver

A finite volume in-house CFD solver is employed to solve the compressible RANS equations (Eq. 3.14-3.16). The viscous flow solution is necessary to resolve flow separation at stall conditions. We have utilized the Spalart-Allmaras turbulence model (Eq. 3.22) [56] for this purpose. A RANS-based solution for turbulence and flow separation is sufficient for our neural network model, since the high Reynolds number indicates less prominent viscous forces. Also, a RANS-based solution is computationally cheaper than the alternatives if we consider the amount of required data.

The solver uses cell-centered finite-volume discretization schemes. We have employed a very accurate second-order HLLC flux scheme [57] for inviscid fluxes, which is an approximate Riemann solver. The second-order accuracy is achieved by the application of the Venkatakrishnan limiter [58]. Venkatakrishnan is an adjustable

limiter, where a parameter allows to adjust the limiter for the shock sharpness or robustness. We have set the adjustment parameter to ensure robust solutions with good fidelity. Second-order central discretization is employed on the viscous terms.

3.2.1 Boundary and Initial Conditions

There are two boundaries in the computational domain whose conditions need to be specified. These are the farfield boundary and the airfoil walls. The farfield boundary is located about 500 chord lengths away from the airfoil to minimize the boundary influence on the flow solution [59]. Non-reflecting type farfield boundary condition is used to specify the thermodynamic state by assigning the free-stream values and the desired Mach number and the angle of attack. The no-slip adiabatic boundary condition is assigned to the airfoil walls. The free-stream values are also used for the initialization of the solution in the computational domain.

3.3 Validation of the Computational Method

The validity of the computational method and the mesh independence study are carried out together with $M = 0.734$, $Re = 6.5 \times 10^6$ and angle of attack of $\alpha = 2.54^\circ$ flow over RAE 2822 airfoil. The results are compared with the experimental results [60]. Note that flow parameters reported in [60] have been updated later by wind tunnel corrections applied in the EUROVAL validation project [61].

Figure 3.1 shows the pressure coefficient C_p distribution over the airfoil surface. Also, the lift and the drag coefficients are given in Table 3.1. All numerical simulations are conducted on very high-quality and high-orthogonality structured meshes. Several structured mesh domains involving different multi-block mesh topologies, external domain sizes, and grid density distributions are investigated. Corresponding mesh independence and solution assessment studies are completed with three different meshes. The details are presented in Table 3.1 in terms of the number of nodes in chord-wise and normal directions around the airfoil, respectively. Figure 3.1 shows that the finer mesh (Mesh 2) results in a more accurate pressure distribution. However, finer mesh than Mesh 2 does not change the solution accuracy significantly. It can be

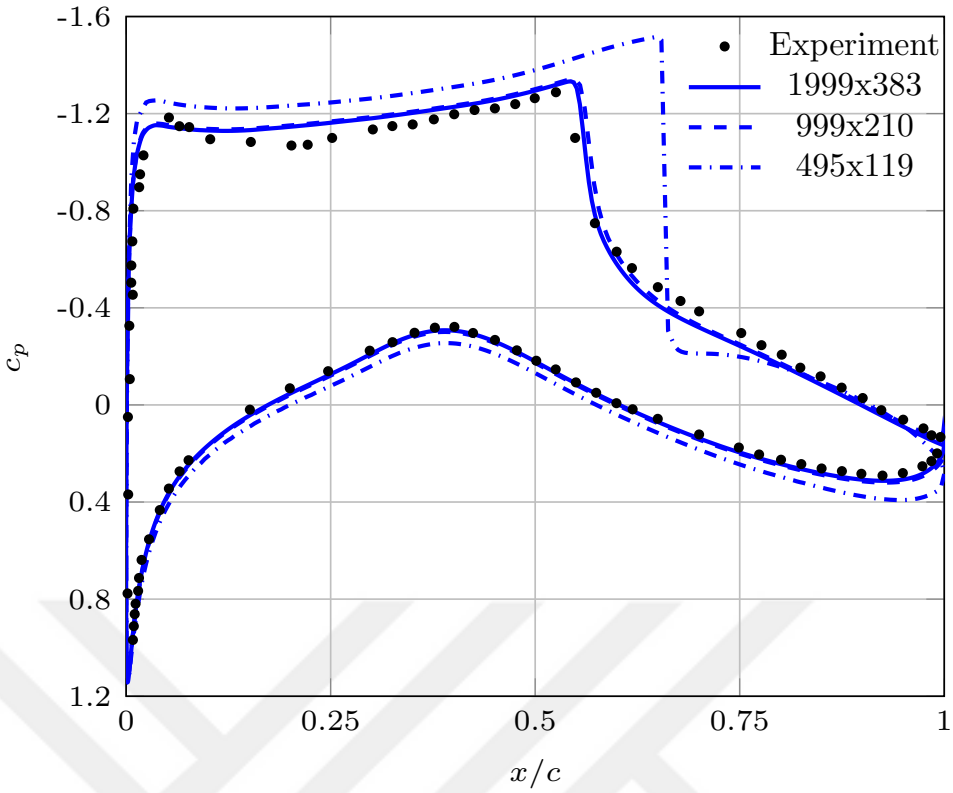
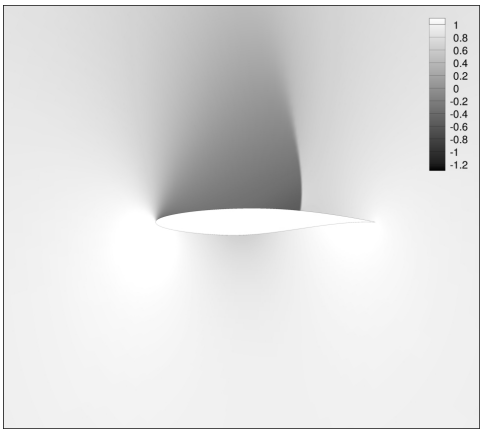


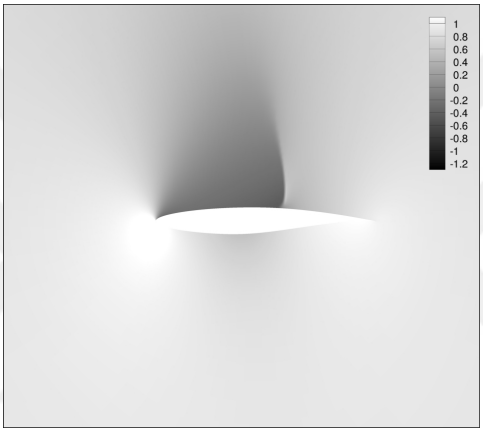
Figure 3.1: Pressure coefficient distribution over RAE 2822 airfoil surface.

concluded that Mesh 2 is found sufficient in order to perform simulations considering not only the solution accuracy but also the computational cost of the simulations. Also, the results agree well with the experimental results given in [60]. Please observe that the sharp gradients in Figure 3.1 prove the careful selection of the flux scheme, limiter function and mesh quality.

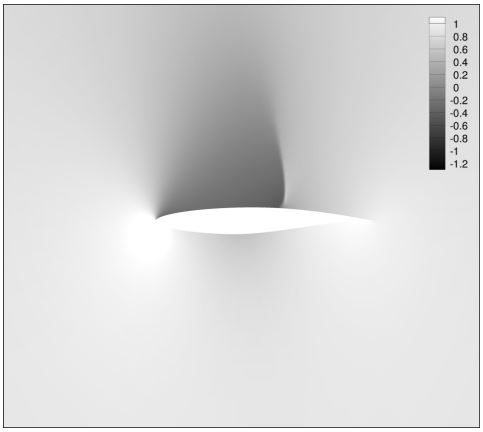
The force coefficients are also given in Table 3.1. As the number of nodes increase, the prediction of force coefficients converges to the experimental data. While sufficient convergence is achieved for the lift coefficient, the drag coefficient shows slow convergence. However, Mesh 2 is found sufficient to perform simulations since more accurate prediction of the drag is not the main concern of this study. Pressure coefficient contours around RAE 2822 airfoil obtained by using the three meshes are presented in Figure 3.2. It is observed that the location of the shock on the upper surface of the airfoil is not captured well with the coarse grid (495×119) as seen in Figure 3.1. However, 999×210 and 1999×383 meshes give almost similar flow fields.



(a)



(b)



(c)

Figure 3.2: Pressure coefficient contours of RAE 2822 by using (a) 495×119, (b) 999×210, (c) 1999×383 meshes.

Table 3.1: The comparison of the force coefficients obtained by using three different meshes with the experimental data.

	Number of Nodes	C_L	**Error (%)**	C_D	**Error (%)**
Mesh 1	495×119	0.984	23.9	0.0232	39.8
Mesh 2	999×210	0.822	3.5	0.0196	18.1
Mesh 3	1999×383	0.804	1.3	0.0183	10.2
Experiment [60]	-	0.794	-	0.0166	-

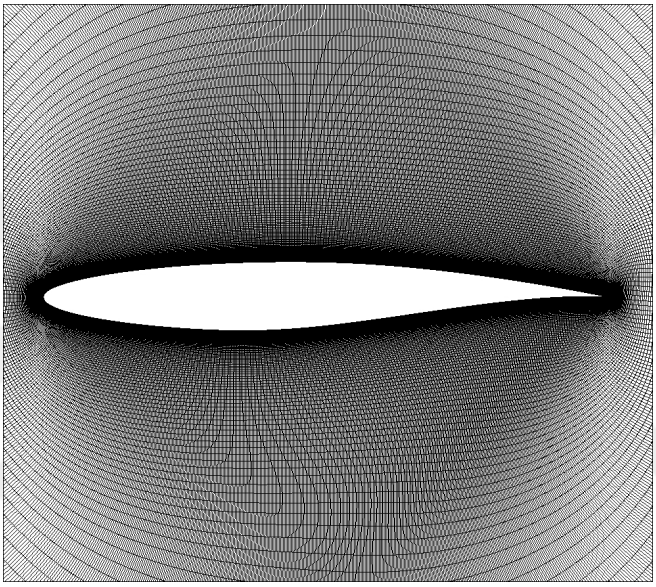


Figure 3.3: Close-up view of the computational domain.

Therefore, a standard and proven computational grid topology to be shared among all airfoils is decided. The meshing algorithm generates a single block O-topology grid with a predetermined number of cells in both chord-wise and normal directions around the airfoils in the database. The cell spacing in the normal direction is chosen appropriately in order to ensure that y^+ is below 1. The farfield is fixed at about 500 times the chord length in order to minimize the effect of boundary conditions on calculated flow variables [59]. Figure 3.3 shows the general view and details of the computational domain near the body.

CHAPTER 4

NEURAL NETWORK ARCHITECTURE

This chapter discusses the details of the proposed neural network model. In the first section, the network configuration is explained under the respective components. Following the first section, the design procedure of the proposed network model is examined by experimenting with several candidate networks in numerous attempts before we settle upon the final architecture.

4.1 Encoder-Decoder Convolutional Neural Networks

The proposed neural network model is designed based on an encoder-decoder convolutional neural network architecture which has been proven to be well-suited for image transformation tasks [62, 63, 39]. An encoder-decoder neural network model is illustrated in Figure 4.1. The model consists of the input layer, the encoder part, the decoder part and the output layer, which will be discussed in the following sections. The model aims to build a mapping function between the input and the output, such as $f(\mathbf{X}) = \mathbf{Y}$, where $\mathbf{X}$ is the airfoil shape, and $\mathbf{Y}$ denotes the flowfield that corresponds to the represented airfoil shape.

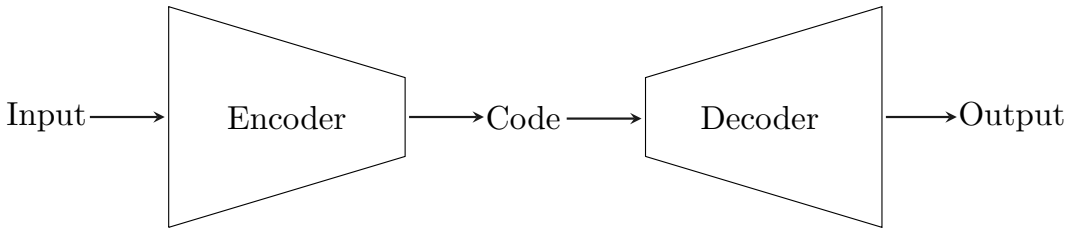


Figure 4.1: Encoder-decoder convolutional neural network

4.1.1 Input Layer

The input of the network is the airfoil shape. All of the geometrical characteristics of an airfoil should be identified in its entirety to label the airfoils. The geometrical parameters can be extracted (some of them are described in Figure 4.2a) and airfoil shapes can be labeled with those parameters. However, it is difficult to provide entire information by geometrical parametrization. In this thesis, the airfoil geometry is represented by the distance field map. The signed distance field (SDF) is proved in many studies [39, 42, 34] as a representation of geometrical shapes for neural networks training. Guo et al. [39] compared the effectiveness of SDF representations with binary representations for a variety of geometrical shapes. The binary representations use a matrix whose elements are unity if the corresponding position is on the boundary or inside the geometry. Guo et al. [39] showed that using SDF representations outperformed binary representations. However, the present study prefers to represent airfoil geometry with its distance field (DF) without assigning the signed part of the distance field for inner elements of an airfoil.

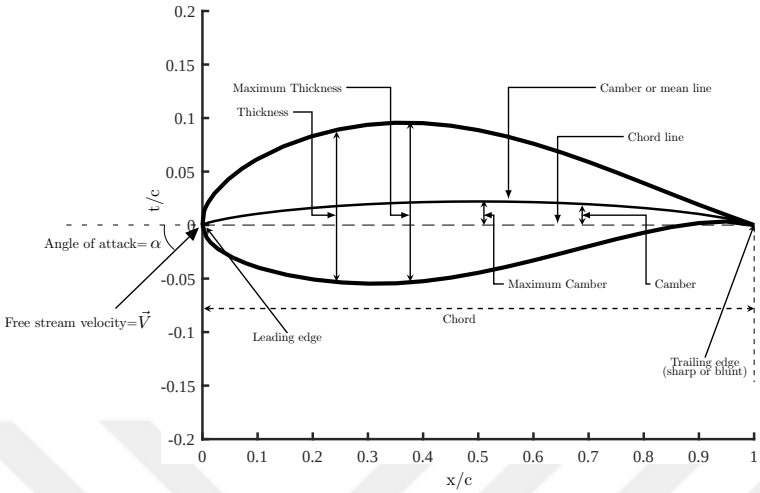
The airfoil shapes are constructed using the coordinates provided from the UIUC database [64] onto a 256×256 Cartesian grid. The DF is created by measuring the Euclidean distance of each grid to the nearest point on the airfoil boundary. The inner elements of the airfoil are assigned to zero. The distance function is defined by Eqn. 4.1, and an illustration of the DF for an arbitrary airfoil is shown in Figure 4.2b.

$$DF(i,j) = \min_{(i^*,j^*)\in\Omega} |(i,j) - (i^*,j^*)| \tag{4.1}$$

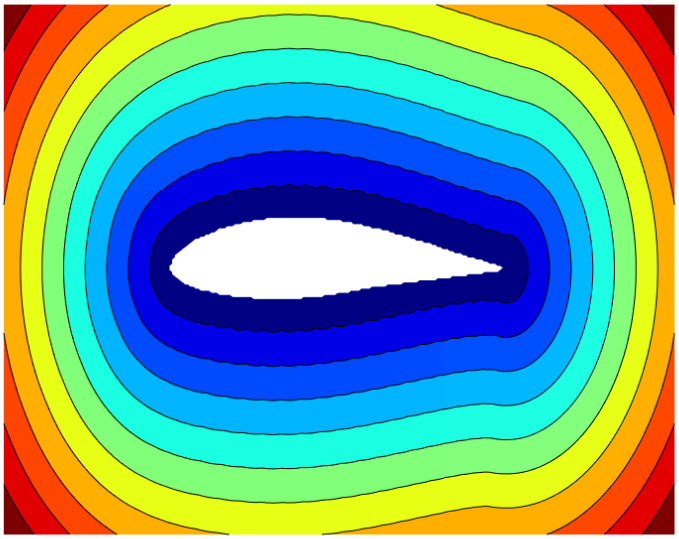
Here, (i,j) is the set of points in the Cartesian grid, and Ω denotes the boundary of the airfoil.

4.1.2 Encoder Stage

The encoder part of the network generates a latent representation, which can be called code, from the distance field of the airfoil shapes. It has a number of blocks each of which is composed of a convolutional layer, batch normalization layer and an activa-



(a) Geometrical characteristics of an airfoil. x/c is the x-location normalized to chord, t/c is the thickness-to-chord ratio.



(b) Distance field of an airfoil.

Figure 4.2: The representation of the airfoil shape.

tion layer. Convolutional layers perform a particular type of linear operation called *convolution* for feature extraction (see Section 2.1.1). The numbers of feature maps in each convolutional layer are determined by performing several experiments in the following sections. A batch normalization layer follows each convolutional layer to stabilize the learning process and accelerate the training of the model by reducing the number of training epochs [65]. Also, a nonlinear activation function is applied after the batch normalization layer. Except for the last layer prior to the decoding stage, the exponential linear unit (ELU) [66] is used as the nonlinear activation function. ELU is defined as follows:

$$\Psi(\phi) = \begin{cases} \phi & \phi > 0 \\ 0.1(e^{\phi} - 1) & \phi \leq 0 \end{cases} \tag{4.2}$$

Each feature map of the previous layer is computed as

$$\mathbf{y}_i = \Psi(\mathbf{W}_i * \mathbf{x}_{i-1} + \mathbf{b}_i) \tag{4.3}$$

where $\mathbf{x}$ is the input feature map. At the very beginning of the feature extraction process, $\mathbf{x}_0$ corresponds to the distance field denoted as $\mathbf{X}$. $\mathbf{W}$ and $\mathbf{b}$ are the weight matrix (convolutional filter) and the additive bias, respectively. $\mathbf{y}$ is the output feature map.

4.1.3 Decoder Stage

Symmetrically, the decoder part has the same number of blocks consisting of the transposed convolutional layers. The transposed convolutional layers are applied to the code extracted from the encoder part to construct the model output, $\mathbf{Y}$ corresponding to the input, $\mathbf{X}$. A transposed convolution layer performs an inverse convolution process. Using a pre-determined filter size and stride, the feature maps are upsampled instead of downsampling to reproduce the desired output, which result in 256×256 flow fields at the end of the decoder stage. Similarly, the exponential linear unit (ELU) without the batch normalization is also applied after each transposed convolution excluding the last layer.

4.2 Architecture Search

Several network architectures in numerous attempts, are given in Table 4.1. These architectures differ by the depth (number of layers) and the width (number of feature maps in each layer). The search strategy is focused on the training performance of each network for a suitable task. The convergence histories of the loss function for each model are observed to decide the best-built architecture for the given task. The training process is performed to produce pressure coefficient field C_p by utilizing the data set consisting of a total of 204 different airfoil shapes at an angle of attack, $\alpha = 0°$ and $M = 0.7$. Figure 4.3 helps to visualize the airfoils shapes used in this study.

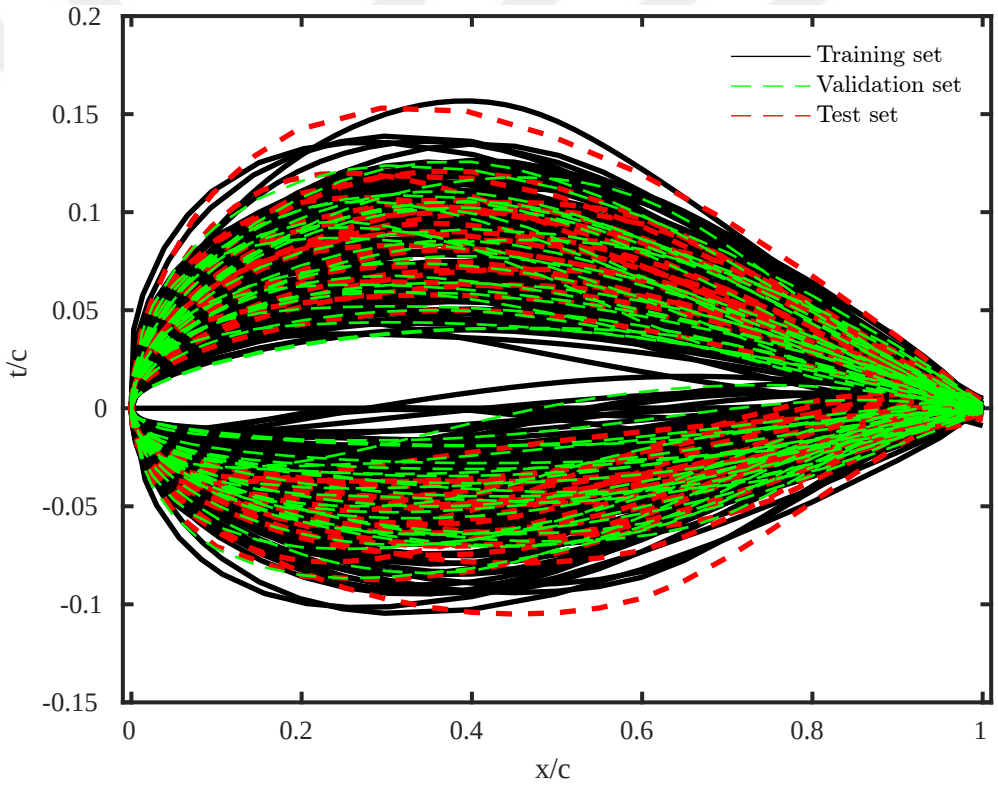


Figure 4.3: Airfoil shapes used in this study. x/c is the x-location normalized to chord, t/c is the thickness-to-chord ratio.

Table 4.1 shows several candidate network architectures as a result of a lengthy trial and error procedure being employed. Each network has a symmetrical structure and utilizes the same size of input. The networks extract the same size of features at the end of the encoder stage and provide the same output size. The first four networks

consist of eight layers in both encoder and decoder parts by varying the number of feature maps in each layer, while the number of layers varies on the remaining two networks.

We have implemented the neural network models on the deep learning framework PyTorch [67]. The models, given in Table 4.1, are trained by utilizing Adam (adaptive moment estimation) optimizer [54], which is a first-order gradient-based optimization algorithm of stochastic objective functions. Batch size of 8 and the learning rate of 10^{-4} are adopted in all the experiments. The training procedure of a neural network is summarized in Figure 4.4. The training is performed to produce pressure coefficient C_p fields as a function of the airfoil shape. Mean squared error is chosen as the loss function expressed as

$$\mathcal{L} = \frac{1}{N}\sum_{k=1}^{N}(C_{pk} - \tilde{C}_{pk})^2 \tag{4.4}$$

where $\tilde{C}_p$ is the model output for the pressure coefficient. N is the number of data points of the flow field and $k \in DF$. C_p data is interpolated using a linear triangulation-based method [68] in order to generate ground truth data onto a Cartesian grid, which corresponds to DF. A linear interpolation of the vertices of the Delaunay triangulation [69] for the raw data is performed to produce $N \times N$ grid form of ground truth data. Since C_p is a non-dimensionalized parameter and squeezed in a range for all airfoils, data normalization is unnecessary. C_p can be computed as

$$C_p = \frac{2}{\gamma M_\infty^2}\left(\frac{p}{p_\infty} - 1\right) \tag{4.5}$$

Figure 4.5 shows the training history of the loss function for all the models given in Table 4.1. The training is terminated after 2000 epochs when the loss no longer reduces. It is observed that the training loss has reduced as the number of layers increases from 4 to 8. Furthermore, the number of layers is fixed as 8, and the number of feature maps is changed. Increasing the number of feature maps results in better learning performance. Also, increasing the filter size as in Model A5 and A6 shows no considerable reduction in the loss function.

Table 4.1: Several candidate network architectures.

Layer	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**
Input	$256 \times 256 \times 1$	$256 \times 256 \times 1$	$256 \times 256 \times 1$	$256 \times 256 \times 1$	$256 \times 256 \times 1$	$256 \times 256 \times 1$
Conv1	F(2×2), S(2), C(8)	F(2×2), S(2), C(16)	F(2×2), S(2), C(32)	F(2×2), S(2), C(64)	F(2×2), S(2), C(64)	F(2×2), S(4), C(64)
Conv2	F(2×2), S(2), C(16)	F(2×2), S(2), C(32)	F(2×2), S(2), C(64)	F(2×2), S(2), C(128)	F(2×2), S(4), C(128)	F(2×2), S(4), C(128)
Conv3	F(2×2), S(2), C(32)	F(2×2), S(2), C(64)	F(2×2), S(2), C(128)	F(2×2), S(2), C(256)	F(2×2), S(2), C(256)	F(2×2), S(4), C(256)
Conv4	F(2×2), S(2), C(32)	F(2×2), S(2), C(64)	F(2×2), S(2), C(128)	F(2×2), S(2), C(256)	F(2×2), S(4), C(256)	F(2×2), S(4), C(512)
Conv5	F(2×2), S(2), C(32)	F(2×2), S(2), C(64)	F(2×2), S(2), C(128)	F(2×2), S(2), C(256)	F(2×2), S(2), C(256)	-
Conv6	F(2×2), S(2), C(64)	F(2×2), S(2), C(128)	F(2×2), S(2), C(256)	F(2×2), S(2), C(512)	F(2×2), S(2), C(512)	-
Conv7	F(2×2), S(2), C(64)	F(2×2), S(2), C(128)	F(2×2), S(2), C(256)	F(2×2), S(2), C(512)	-	-
Conv8	F(2×2), S(2), C(64)	F(2×2), S(2), C(128)	F(2×2), S(2), C(256)	F(2×2), S(2), C(512)	-	-
Code	1×1	1×1	1×1	1×1	4×4	16×16
Deconv1	F(2×2), S(2), C(64)	F(2×2), S(2), C(128)	F(2×2), S(2), C(256)	F(2×2), S(2), C(512)	F(2×2), S(2), C(256)	F(2×2), S(4), C(256)
Deconv2	F(2×2), S(2), C(64)	F(2×2), S(2), C(128)	F(2×2), S(2), C(256)	F(2×2), S(2), C(512)	F(2×2), S(2), C(256)	F(2×2), S(4), C(128)
Deconv3	F(2×2), S(2), C(32)	F(2×2), S(2), C(64)	F(2×2), S(2), C(128)	F(2×2), S(2), C(256)	F(2×2), S(4), C(256)	F(2×2), S(4), C(64)
Deconv4	F(2×2), S(2), C(32)	F(2×2), S(2), C(64)	F(2×2), S(2), C(128)	F(2×2), S(2), C(256)	F(2×2), S(2), C(128)	F(2×2), S(4), C(1)
Deconv5	F(2×2), S(2), C(32)	F(2×2), S(2), C(64)	F(2×2), S(2), C(128)	F(2×2), S(2), C(256)	F(2×2), S(4), C(64)	-
Deconv6	F(2×2), S(2), C(16)	F(2×2), S(2), C(32)	F(2×2), S(2), C(64)	F(2×2), S(2), C(128)	F(2×2), S(2), C(1)	-
Deconv7	F(2×2), S(2), C(8)	F(2×2), S(2), C(16)	F(2×2), S(2), C(32)	F(2×2), S(2), C(64)	-	-
Deconv8	F(2×2), S(2), C(1)	F(2×2), S(2), C(1)	F(2×2), S(2), C(1)	F(2×2), S(2), C(1)	-	-
Output	$256 \times 256 \times 1$	$256 \times 256 \times 1$	$256 \times 256 \times 1$	$256 \times 256 \times 1$	$256 \times 256 \times 1$	$256 \times 256 \times 1$

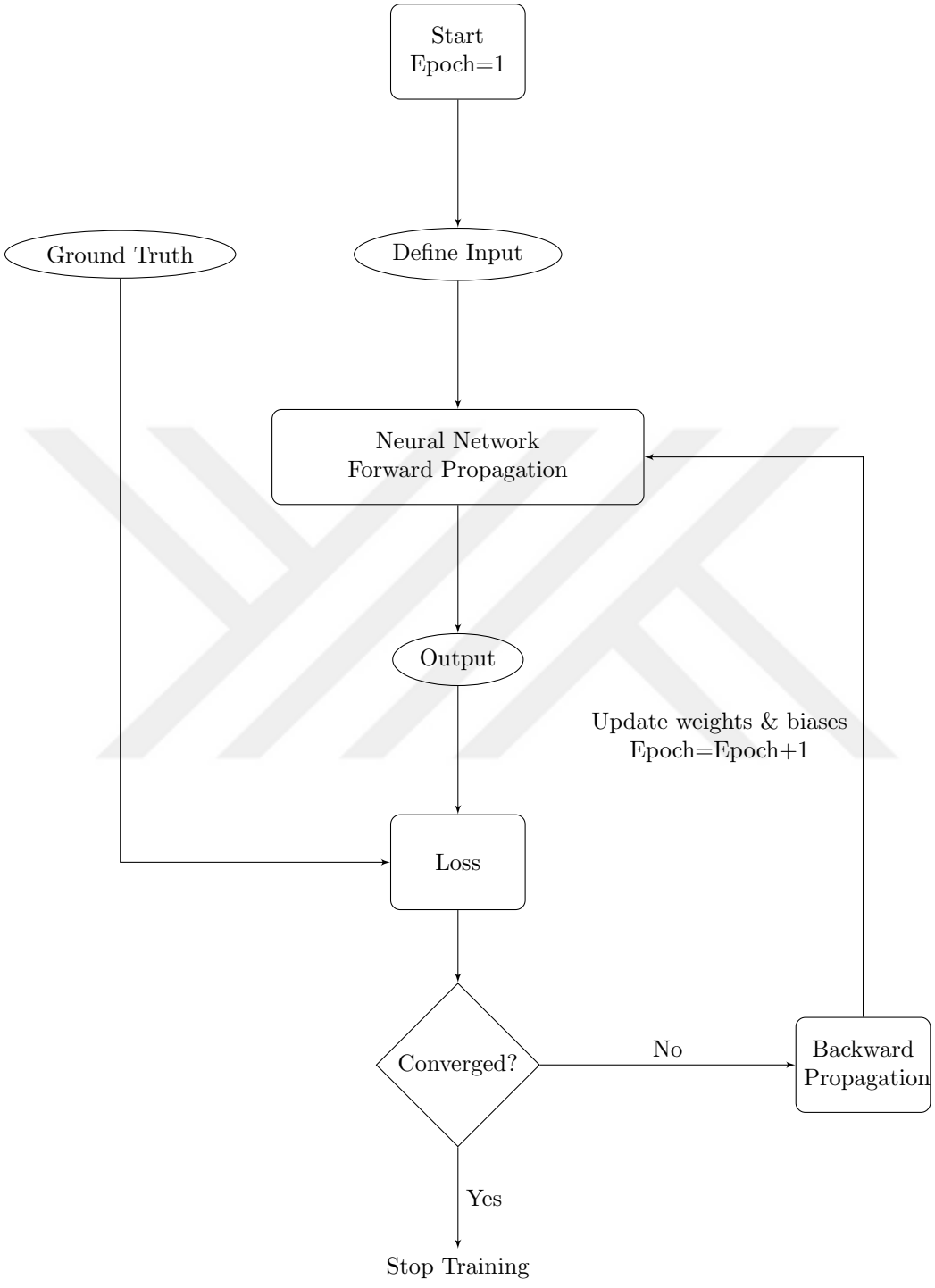


Figure 4.4: Flow chart of the network training.

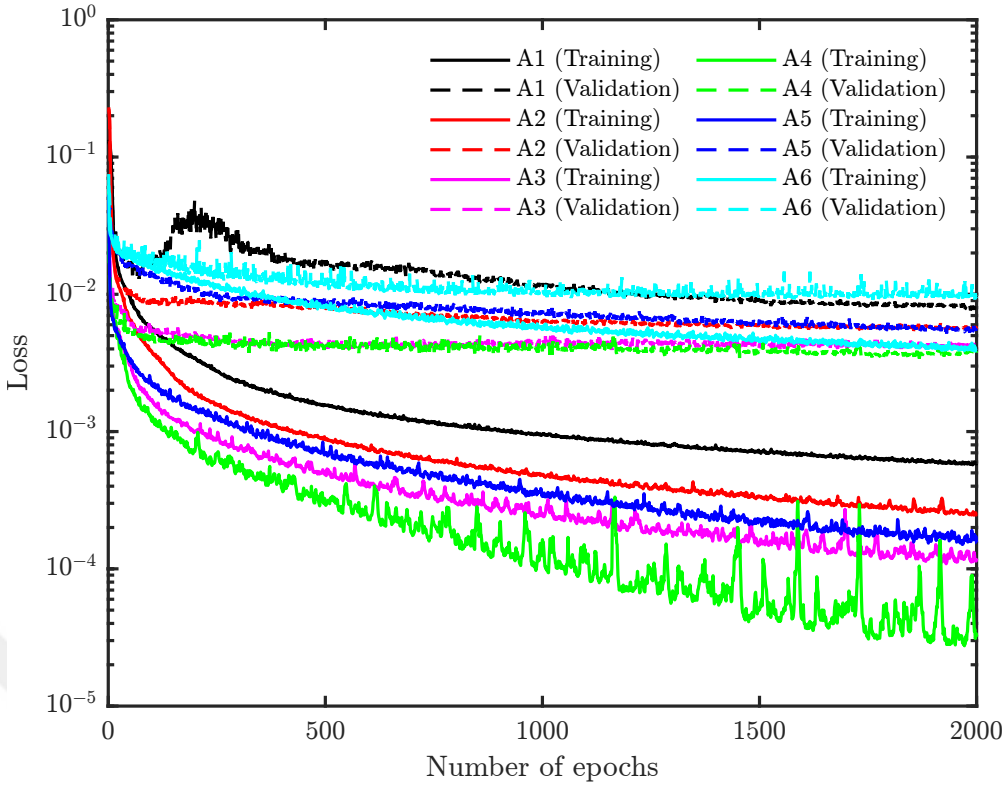


Figure 4.5: Training history of the models given in Table 4.1.

Figure 4.6 illustrates the final shape of the proposed neural network model, named CNNFOIL [70]. The distance fields of the airfoil shapes with a size of 256×256 are extracted to 512 feature maps with a size of a single data point (1×1) by downscaling with a factor of 2 by convolution operations in the encoder part. Then, those feature maps are fed to the decoder part in order to construct the flowfield by mirroring the downscaling process in which the number of the feature maps decreases and the spatial resolution of features after each transposed convolutional layer increases. Both the encoder and the decoder parts have 8 blocks. Each block consists of the same types of layers: convolutional or transposed convolutional layer, batch normalization layer (only for the encoder part) and ELU (exponential linear unit) activation layer except output blocks. All convolution and transposed convolution operations are performed using a 2×2 filter size and a stride of 2. Please note that increasing the filter size and the stride have caused checkerboard predictions. The high gradient regions are prone to cause the checkerboard pattern in the flowfield. Therefore we utilized 2×2 filter size and the stride of 2 to overcome this issue.

The learning rate of the optimizer is one of the most critical parameters for the training

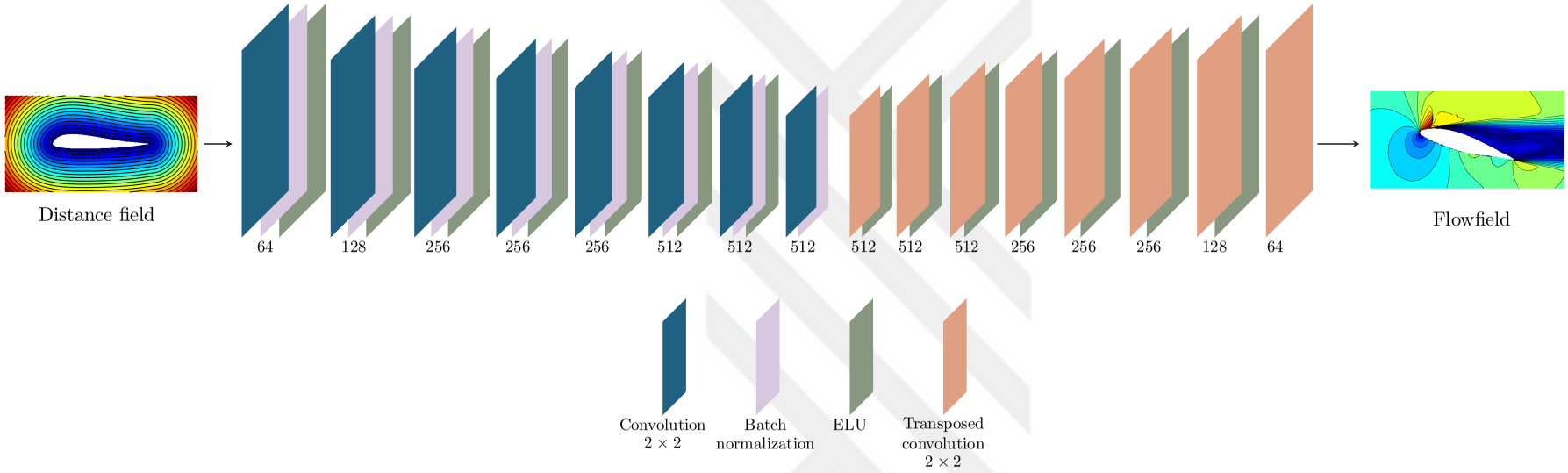


Figure 4.6: CNNFOIL: Encoder-decoder convolutional neural network.

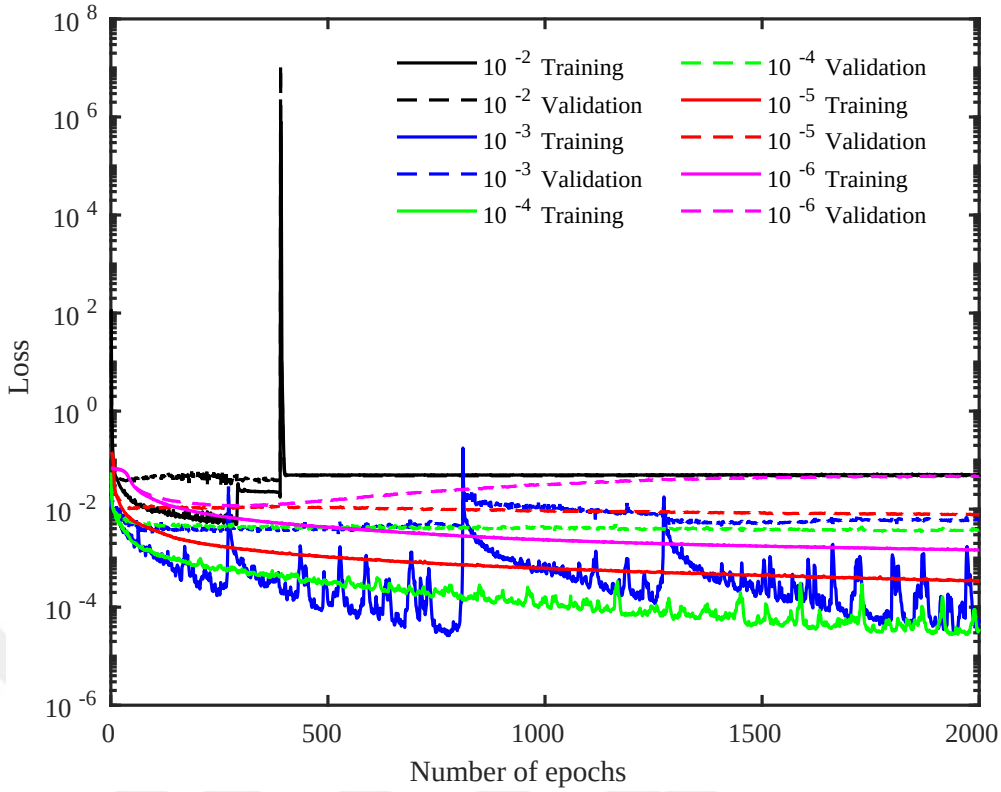


Figure 4.7: Effect of learning rate on training.

and needs to be adjusted by the user. The learning rate controls how much the weights and biases are updated with respect to the loss gradient at each epoch (See Section 2.1.4). Large learning rates lead to overshoots, and small changes in the parameters can not be learned. On the other hand, the learning progresses slowly and takes longer times in small learning rates. Figure 4.7 shows the loss curves of various learning rates. We experimented with the effect of the learning rate on the training of CNNFOIL with five different values from 10^{-2} to 10^{-6}. Figure 4.7 depicts that the training with the learning rate of 10^{-4} converges to a lower loss compared with a larger or smaller learning rate. Further training tasks are employed with a learning rate of 10^{-4}.

CHAPTER 5

FLOW FIELD PREDICTIONS AROUND AIRFOILS

The proposed neural network model is trained to produce the flow field around various airfoil shapes for a wide range of angles of attack from -10° to 20° at a free-stream Mach number, $M = 0.7$ and free-stream Reynolds number, $Re = 6 \times 10^6$. Then, both qualitative and quantitative evaluations of the predicted fields are performed through our trained model.

5.1 Network Training

The dataset of the pressure coefficient and Mach number fields are obtained for 204 airfoil shapes and angles of attack range from -10° to 20° through CFD simulations. CFD simulations are performed at a free-stream Mach number, $M = 0.7$ and free-stream Reynolds number, $Re = 6 \times 10^6$. The data of 6324 cases are divided randomly into 85% for training and validation sets and 15% unseen data during the training for the test set in order to assess the learning performance of the model. The airfoil shapes are not selected from a particular family or by considering the performance requirement. Besides, the selected airfoils are not necessarily designed for the transonic flow regime. The airfoil shapes are chosen randomly since this study is interested in the variation of geometrical characteristics of an airfoil shape.

The data is standardized by dividing each flow variable by its standard deviation combined with mean-centering as expressed as

$$x_n = \frac{x - \mu}{\sigma} \tag{5.1}$$

Here, x and x_n represent the flow variable and the normalized flow variable, respectively. μ and σ are the mean and standard deviation of the flow variable, respectively. Original flow field solution data is interpolated from the CFD solution on a structured mesh to generate ground truth data onto a 256×256 Cartesian grid, which corresponds to the distance field of each airfoil shape. The developed model is first trained by employing Adam [54] optimizer with an initial learning rate of 1×10^{-4}. The training aims to minimize the output of the loss function on the training set. The loss function measures the dissimilarity between the output flow field of the model and the ground truth. The mean square error is used as the loss function expressed as

$$\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [(C_{p_k} - \tilde{C}_{p_k})^2 + (M_k - \tilde{M}_k)^2] \tag{5.2}$$

where $\tilde{C}_p$ and $\tilde{M}$ are the model outputs for the pressure coefficient and Mach fields, respectively. N is the number of data points of the flow field and $k \in DF$. The training process is performed until the loss reaches the steady-state for the validation data set. The convergence history of the loss function is monitored for both training and validation sets, as shown in Figure 5.1. The training is terminated after 4800 epochs when the loss no longer reduces considerably for training and validation sets.

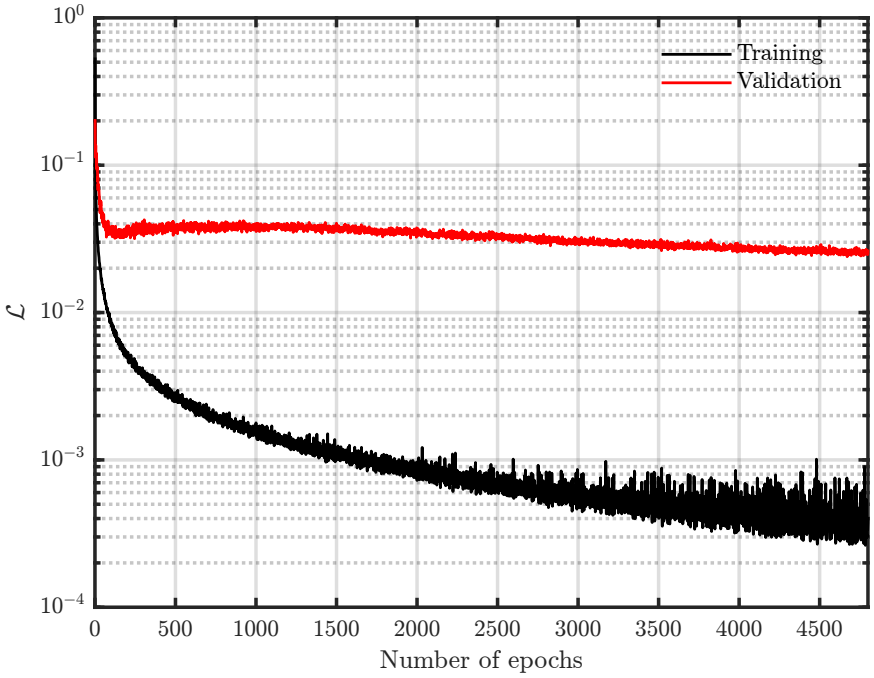


Figure 5.1: Training history of the model.

After the training is completed, the prediction performance of the model is evaluated by performing experiments utilizing the unseen cases in the test set. Several evaluation metrics are used. The model outputs are compared with the ground truth flow fields qualitatively by the absolute difference of the field quantity between the model output and the ground truth. The mean absolute percentage error ($\mathbf{MAPE}$) is investigated for each case. $\mathbf{MAPE}$ is defined as

$$\mathbf{MAPE} = \frac{1}{N}\sum_{k=0}^{N}\left|\frac{y^k - \tilde{y}^k}{y^k}\right| \times 100 \tag{5.3}$$

where N is the number of data points and $k \in DF$, y and $\tilde{y}$ are the ground truth and the model output, respectively. Accuracy can be measured as $(100 - \mathbf{MAPE})$. Accuracy with the threshold ($\mathbf{AWT}$) is also commonly used in prior studies [71, 72, 73]. $\mathbf{AWT}$ is defined as

$$\max(\frac{y}{\tilde{y}}, \frac{\tilde{y}}{y}) < \delta \tag{5.4}$$

Here, y and $\tilde{y}$ are the ground truth and the model output, respectively. δ is the threshold, $k \in DF$ denotes the data points in the field. Since the C_p field is clustered around 0, (that indicates pressures close to the free stream pressure), the $\mathbf{AWT}$ error estimations can be calculated too large to assess the quality of the estimation. Errors estimated with this method can be misleading since division by a near-zero C_p may produce large errors even if the actual error on the pressure is small. This issue can be easily addressed by applying relative absolute error for the data points where $|y| < 0.1$ by adding a bias to the denominator as $\frac{|y-\tilde{y}|}{1+|y|}$. It is known that the closeness to zero and the bias are not invariant and scale-dependent. This new error definition is more reliable and improves the interpretability of the prediction performance of the model.

5.2 Experiments

In order to evaluate the neural network modeling, several experiments are performed to study how the model responds to the effect of changes to the airfoil characteristics.

The shape effects are investigated by changing some key geometric components such as the maximum camber and the maximum thickness at the same angle of attack. Also, the results of the model and the CFD simulations are compared for two arbitrary airfoils at various angles of attack. The results are organized in the following sequence for each test: first, the ground truth and the model output of the C_p and M field contour plots are compared. Then, the accuracy measurements of each case are presented. Also, we present the C_p distribution along the airfoil surfaces. Note that the airfoil shapes are masked through the training process, and the boundary data is not provided to the model. Therefore, the C_p data is extracted from the immediate vicinity of the surface (approximately 1% of the chord) in the normal direction. This is a valid assumption that the static pressure can be considered constant through the normal direction in the boundary layer based on the boundary layer theory [74].

5.2.1 Effect of maximum thickness

Several airfoil pairs of different maximum thicknesses among the test set are selected to evaluate the performance of the model. Key parameters of the airfoils are listed in Table 5.1. The maximum thickness-to-chord ratio and its location, maximum camber-to-chord ratio and its location are presented in Table 5.1. Please note that each airfoil pair are investigated only at the available angle of attack due to random splitting of the data into training, validation and test sets. As mentioned before, the airfoil shapes in this study are chosen randomly since this study is interested in the variation of geometrical characteristics of an airfoil shape.

Figure 5.2 shows the shapes of the test airfoils. Figure 5.2 also previews the effect of the airfoil thickness on the surface C_p distribution. The comparison between the ground truth and the model output for the C_p and M fields, including the corresponding absolute difference for airfoil pairs is shown in Figure 5.3. As the airfoil becomes thicker, the overall structure of the flow fields does not change significantly between airfoils in each pair. However, changing airfoil thickness can alter the shock location or thickness of the boundary layer, as shown in Figure 5.4-5.6.

C_p distribution comparison in Figure 5.4 shows a noticeable difference between the airfoils in Pair 1 on the stagnation point. A shock wave can be observed in the airfoils

Table 5.1: Shape parameters of the maximum thickness test cases

Pairs	**Airfoil**	**Type**	**Maximum Thickness**		**Maximum Camber**		**AoA** (°)
			t/c (%)	@c (%)	b/c (%)	@c (%)	
Pair 1	NACA 1408	General	8	30	1.0	40	18
	NACA 1412	Aviation	12	30	1.0	40	18
Pair 2	HQ 3011	Sailplane	11	35	3.0	50	8
	HQ 3014		14	35	3.0	50	8
Pair 3	NACA 63 − 206	General	6	35	1.1	50	5
	NACA 63 − 215	Aviation	15	35	1.1	50	5

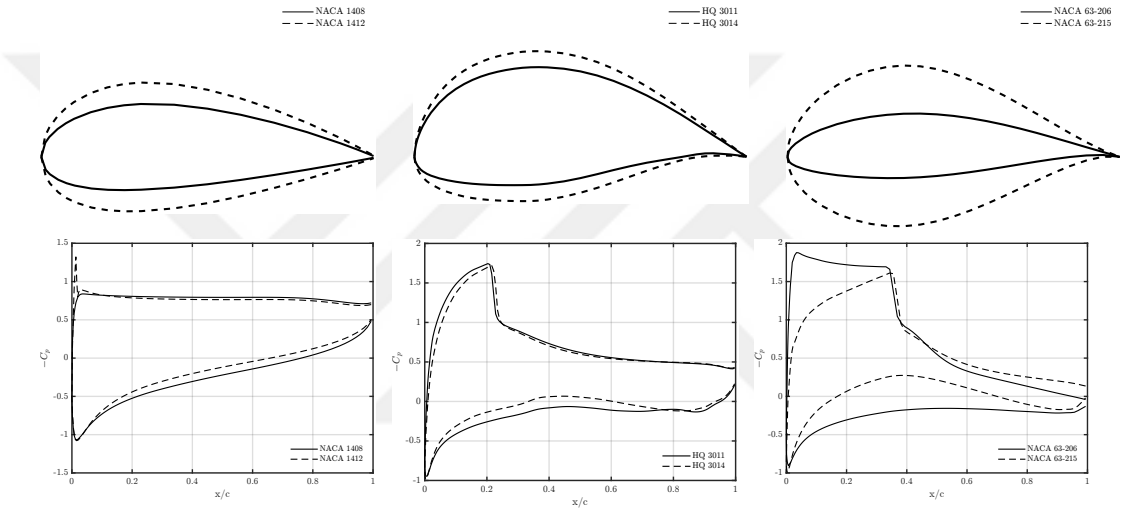


Figure 5.2: Shapes of the test airfoils in Table 5.1 and the comparison of the surface C_p distribution between the airfoils in each pair. (left) Pair 1, (middle) Pair 2, (right) Pair 3.

in Pair 2, as shown in Figure 5.3. Thickness variation have resulted in a subtle change in the boundary layer thickness and the shock location. Unlike the other two pairs, there is a larger difference in the airfoil thickness between the airfoils in Pair 3. This difference resulted in a major difference in the surface C_p distribution, as shown in Figure 5.2 and Figure 5.6. A relatively flat trend in the pressure distribution can be observed for the thin airfoil against the thick airfoil on the suction side from leading edge up to shock wave as well as the most of the pressure side. It is observed that the neural network model has captured the effects of the thickness variation in the flow field of various airfoil pairs well. Even though the magnitude of the error is

considerable on the high gradient zones of C_p and M data along the flow field, the accuracy of the model is high enough to reproduce the overall structure of the flow field.

Table 5.2 reports the accuracy of the predicted flow fields with different metrics mentioned before. The table is organized as C_p evaluations in three columns followed by Mach number evaluations in the same order. For both C_p and M evaluations, the first two columns correspond to the percentage of the number of data points that satisfy the given metric, $\mathbf{AWT}$ in Eq. 5.4. The third column corresponds to the relation $100-\mathbf{MAPE}$. The threshold is chosen ($\delta = 1.1$ and $\delta = 1.2$) for $\mathbf{AWT}$. The threshold allows monitoring a certain closeness of the predicted data to the ground truth. As the threshold approaches 1, the $\mathbf{AWT}$ indicates more accurate data points.Data points having more than 100% error are treated as outliers and excluded from the $\mathbf{MAPE}$ calculations. At most 2% of data in a predicted field are treated as outliers. Even though the outliers are excluded from the $\mathbf{MAPE}$ calculations, they are still present and can be observed in C_p and M fields given in Figure 5.3. Table 5.2 shows that our model achieves high performance for both C_p and M field predictions.

Table 5.2: Prediction accuracy results of C_p and M fields of the test cases in Table 5.1.

Case	C_p			M		
	$AWT(\delta = 1.1)(\%)$	$AWT(\delta = 1.2)(\%)$	$100 - MAPE$	$AWT(\delta = 1.1)(\%)$	$AWT(\delta = 1.2)(\%)$	$100 - MAPE$
NACA 1408	99.762	99.826	98.483	99.570	99.815	98.716
NACA 1412	99.240	99.971	97.686	99.835	99.957	98.950
HQ 3011	99.498	99.818	96.832	99.812	99.946	99.436
HQ 3014	99.555	99.800	96.022	99.603	99.891	99.234
NACA 63 – 206	99.388	99.735	94.054	99.655	99.848	99.500
NACA 63 – 215	99.188	99.768	92.676	99.467	99.795	99.329
Test set	97.424	99.275	94.045	99.180	99.716	99.012

5.2.2 Effect of maximum camber

Similarly, in order to further investigate the sensitivity of the model predictions to handle the airfoil shape variation, airfoil shapes with different maximum cambers from the test set are examined. The shape parameters of the test airfoils are presented in Table 5.3.

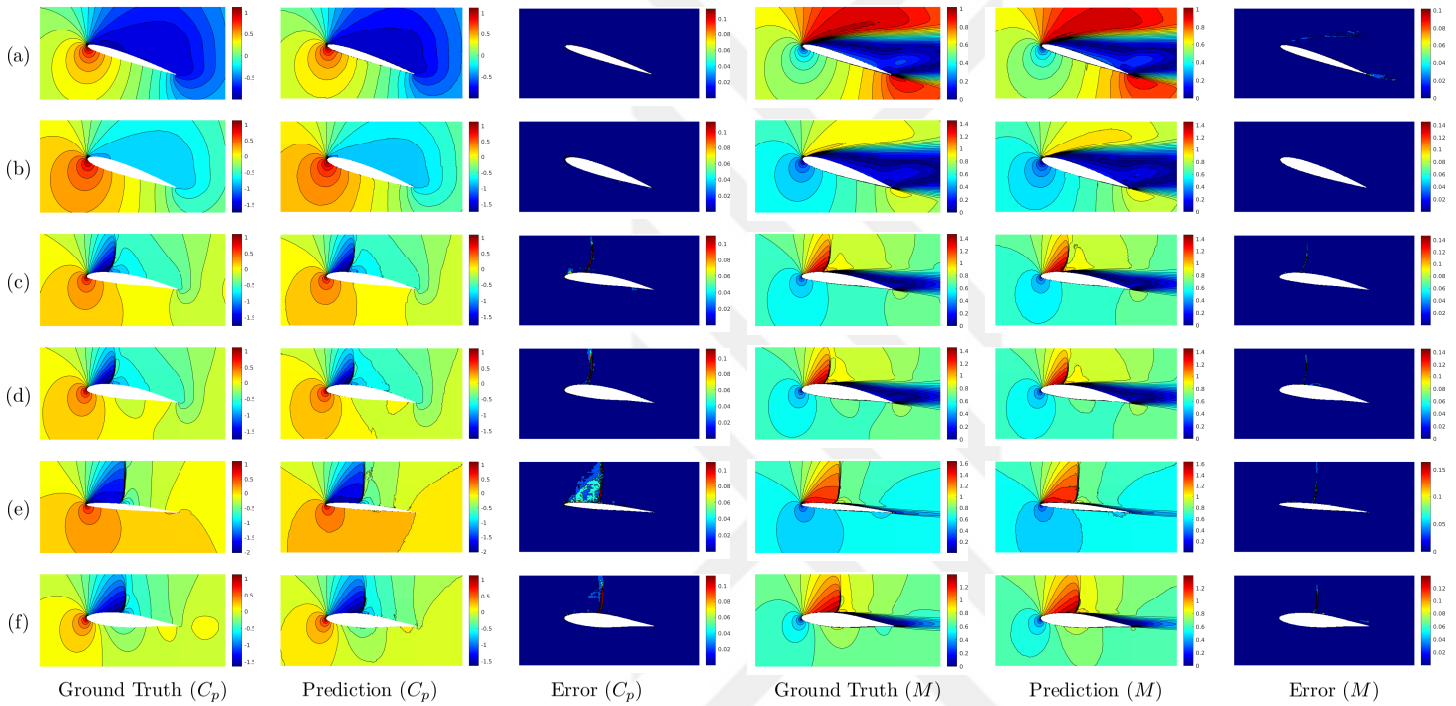
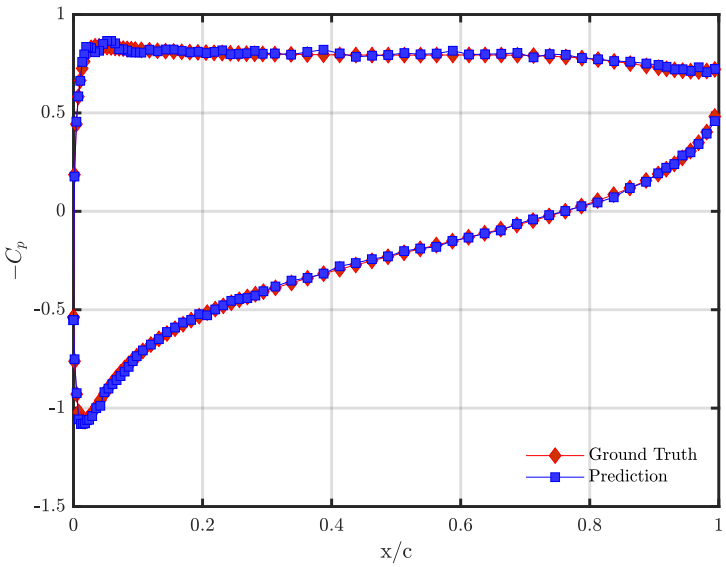
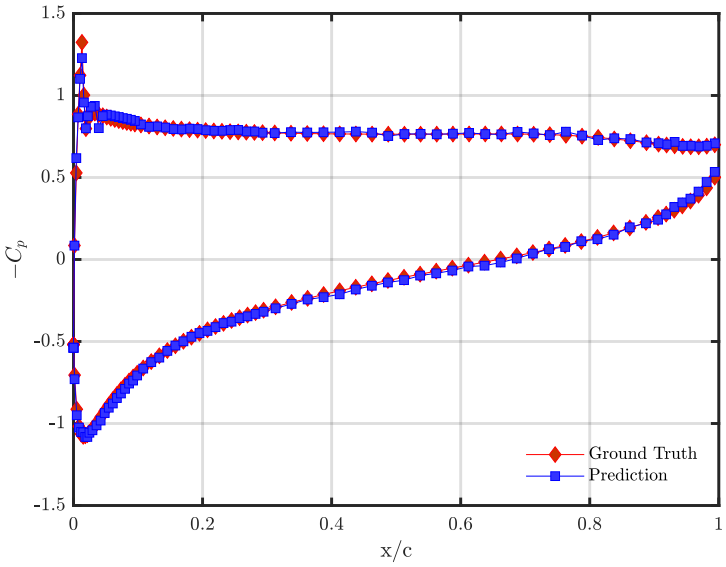


Figure 5.3: The comparison of the C_p and M fields of the test cases in Table 5.1. (a) NACA 1408, (b) NACA 1412, (c) HQ 3011, (d) HQ 3014, (e) NACA $63-206$, (f) NACA $63-215$.

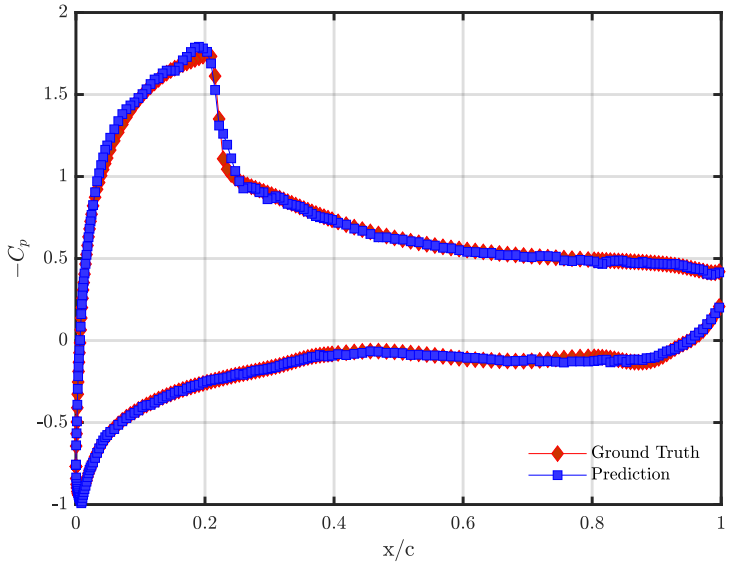


(a)

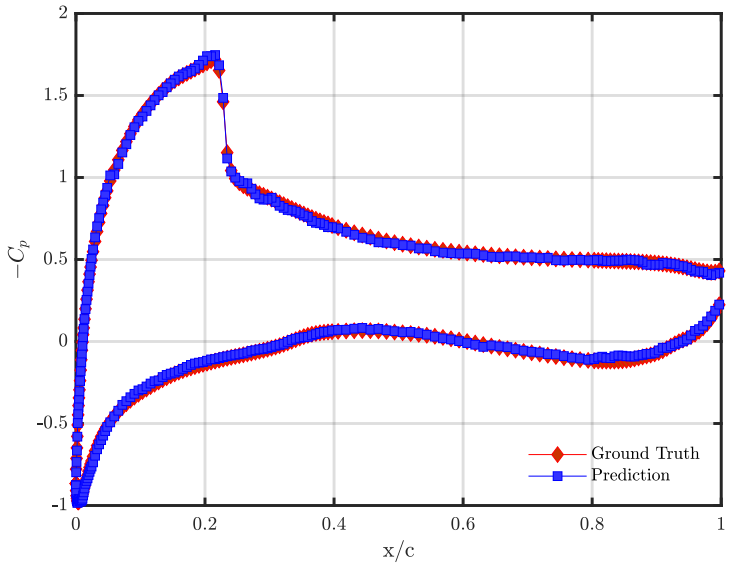


(b)

Figure 5.4: Surface pressure coefficient distribution of the test cases in Table 5.1. (a) NACA 1408, (b) NACA 1412.

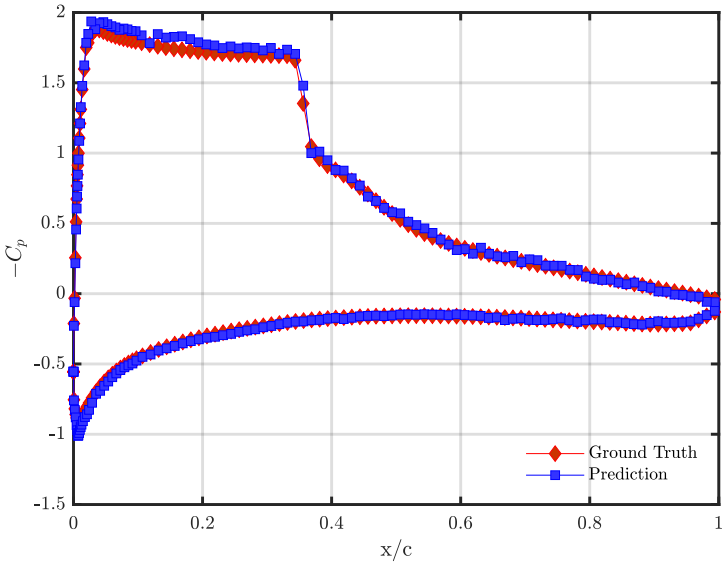


(a)

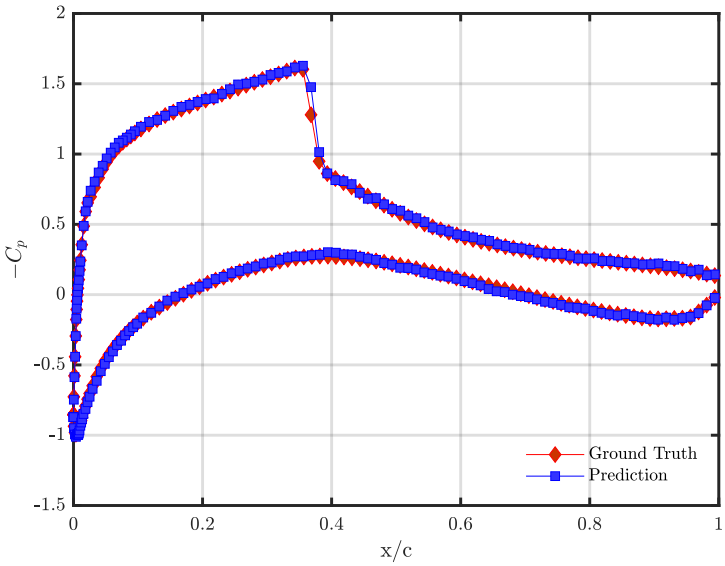


(b)

Figure 5.5: Surface pressure coefficient distribution of the test cases in Table 5.1. (a) HQ 3011, (b) HQ 3014.



(a)

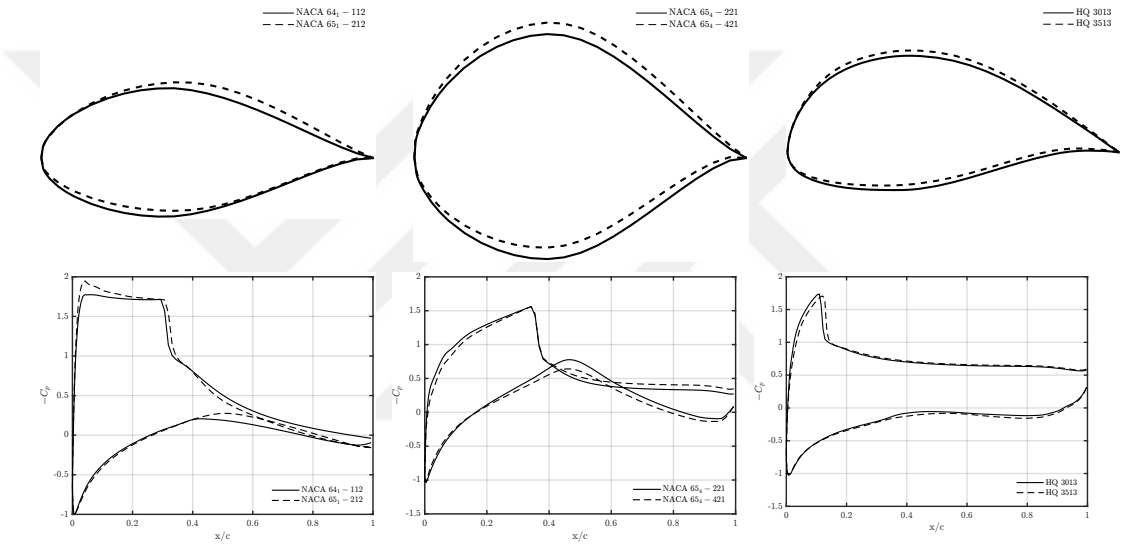


(b)

Figure 5.6: Surface pressure coefficient distribution of the test cases in Table 5.1. (a) NACA $63-206$, (b) NACA $63-215$.

Table 5.3: Shape parameters of the maximum camber test cases

Pairs	**Airfoil**	**Type**	**Maximum Thickness**		**Maximum Camber**		**AoA** (°)
			t/c (%)	@c (%)	b/c (%)	@c (%)	
Pair 1	NACA $64_1 - 112$	Laminar	12	40	0.6	50	-6
	NACA $65_1 - 212$	Flow	12	40	1.1	50	-6
Pair 2	NACA $65_4 - 221$	Laminar	21	40	1.1	50	7
	NACA $65_4 - 421$	Flow	21	40	2.2	50	7
Pair 3	HQ 3013	Sailplane	13	35	3.0	50	12
	HQ 3513		13	35	3.5	50	12



(a)

Figure 5.7: Shapes of the test airfoils in Table 5.3 and the comparison of the surface C_p distribution between the airfoils in each pair. (left) Pair 1, (middle) Pair 2, (right) Pair 3.

Figure 5.7 shows the shapes of the test airfoils. Figure 5.7 also previews the effect of the airfoil camber on the surface C_p distribution. In particular, the impact of the camber change is reflected as the pressure peak on the pressure side between the airfoils in Pair 1 as shown in Figure 5.7. For the Pair 2, the effect of the camber change shows itself with subtle changes along the airfoil surface. However, a very minimal impact is observed between the airfoils in Pair 3. The location of shock wave is shifted to the middle of the airfoil as the maximum camber increases.

Table 5.4: Prediction accuracy results of C_p and M fields of the test cases in Table 5.3.

Case	C_p			M		
	$AWT(\delta = 1.1)(\%)$	$AWT(\delta = 1.2)(\%)$	$100 - MAPE$	$AWT(\delta = 1.1)(\%)$	$AWT(\delta = 1.2)(\%)$	$100 - MAPE$
NACA $64_1 - 112$	98.140	99.789	91.629	99.486	99.903	99.498
NACA $65_1 - 212$	99.105	99.517	94.172	99.347	99.741	99.526
NACA $65_4 - 221$	98.651	99.733	93.401	99.431	99.886	99.190
NACA $65_4 - 421$	97.792	99.791	94.168	99.593	99.887	99.281
HQ 3013	99.610	99.877	97.690	99.809	99.951	99.238
HQ 3513	99.476	99.820	97.064	99.619	99.914	98.986
Test set	97.424	99.275	94.045	99.180	99.716	99.012

The C_p and M fields are compared against the CFD simulations in Figure 5.8. Although the neural network model shows relatively high errors in high gradient regions, overall flow fields agree well. The model successfully responds the subtle changes of the airfoil geometry in the flow fields and the surface C_p distributions given in Figure 5.9-5.11 due to the small camber variations shown in Figure 5.7. Figure 5.9-5.11 show that the model predictions agree well with CFD simulation results and reacts well to subtle changes of the C_p distributions shown in Figure 5.7 as the airfoil shape changes. Table 5.4 quantitatively validates the visual accuracy observed on both C_p and M field predictions in Figure 5.8.

5.2.3 The angle of attack variation

Similar to previous tests, the performance of the neural network model is also assessed by considering the evolution of the flow field as the angle of attack changes. Two different airfoil shapes, Eppler 547 and NACA $66_3 - 218$ chosen arbitrarily are examined. These airfoils have a large number of angle of attack cases sweeping an extensive range in the test set.

5.2.3.1 Eppler 547

Eppler 547 is a general-aviation airfoil, has a maximum thickness-to-chord ratio of 17.4 at 40% chord and a maximum camber-to-chord ratio of 2.0 at the 30.7% chord. Figure 5.12 shows the comparison between the ground truth and the model output

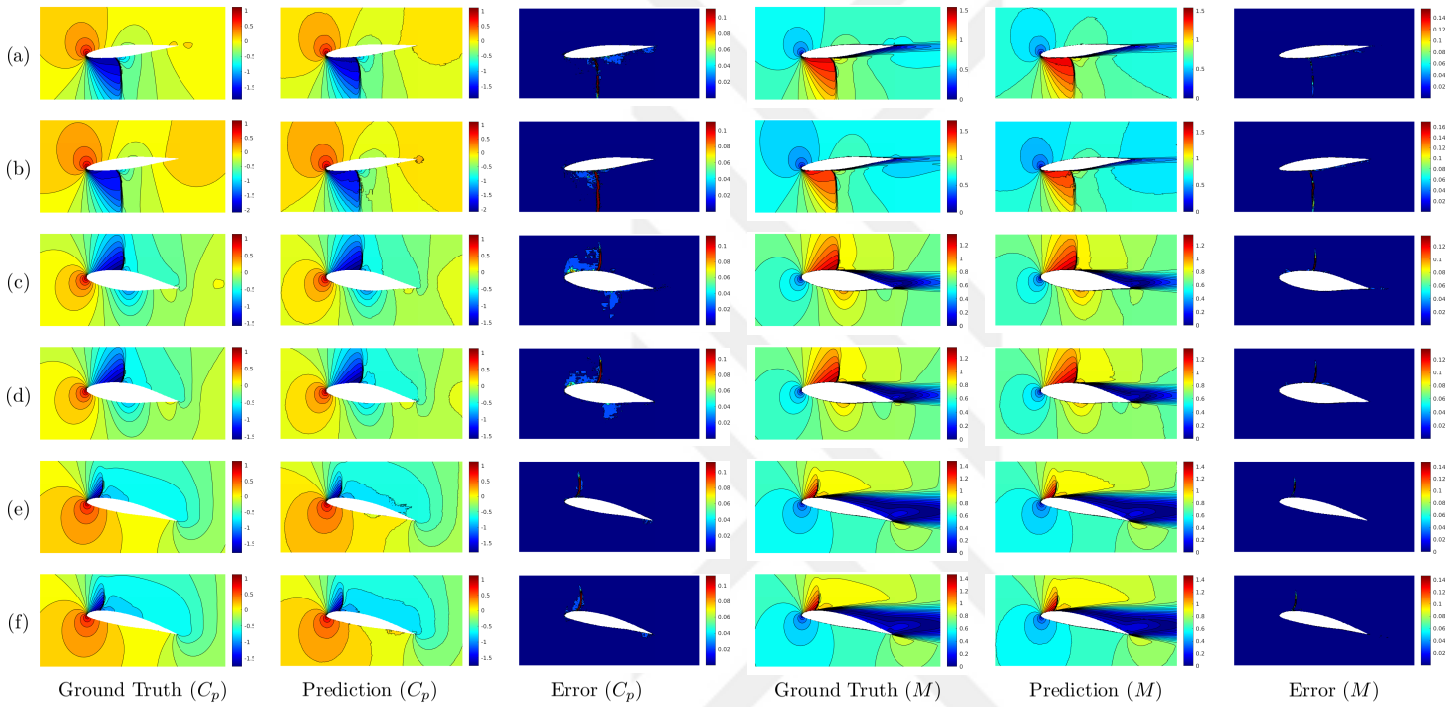
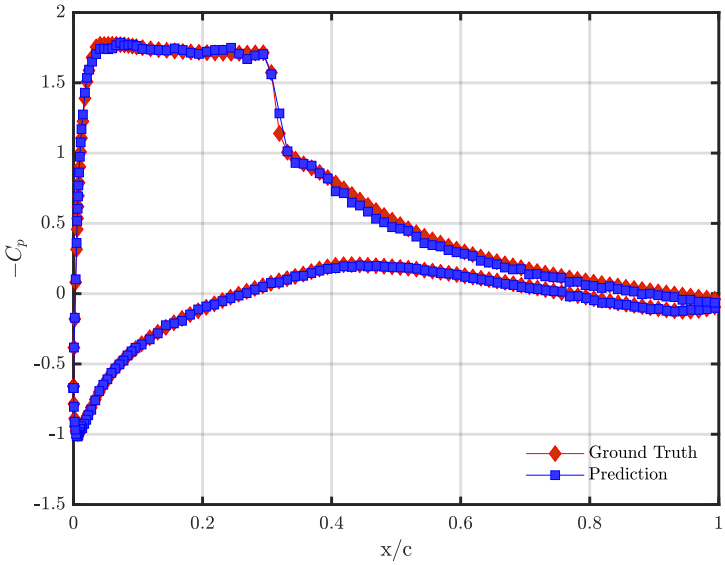
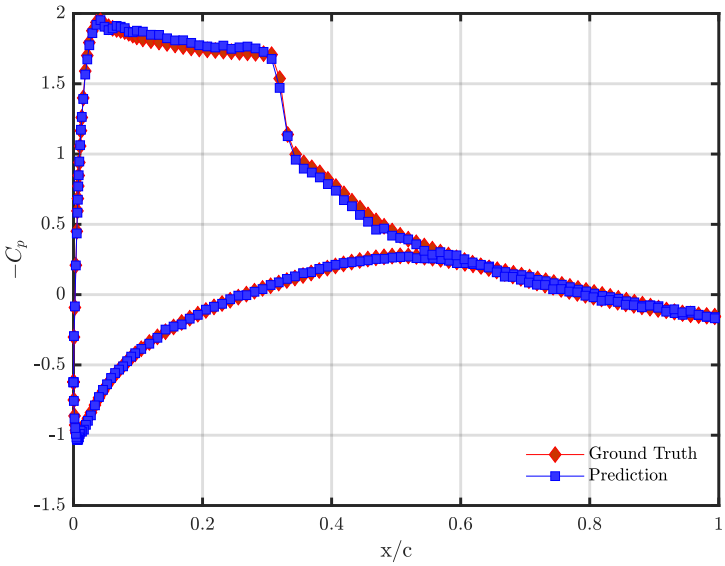


Figure 5.8: The comparison of the C_p and M fields of the test cases in Table 5.3. (a) NACA $64_1 - 112$, (b) NACA $65_1 - 212$, (c) NACA $65_4 - 221$, (d) NACA $65_4 - 421$, (e) HQ 3013, (f) HQ 3513.

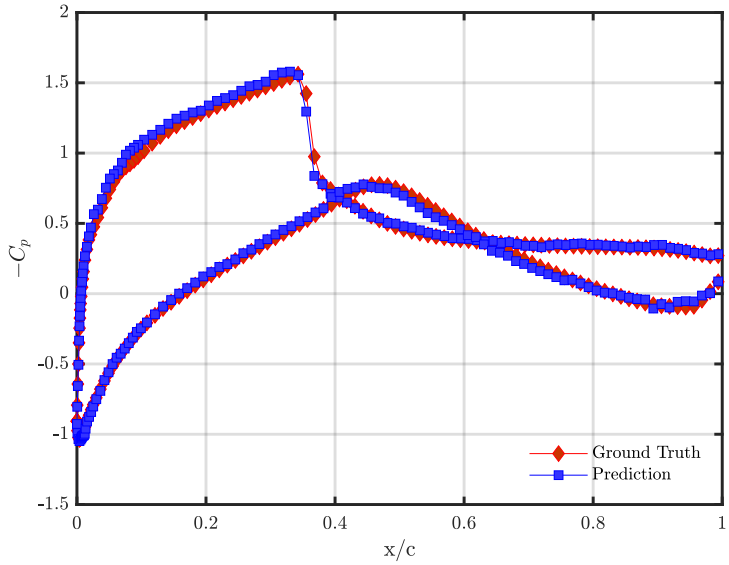


(a)

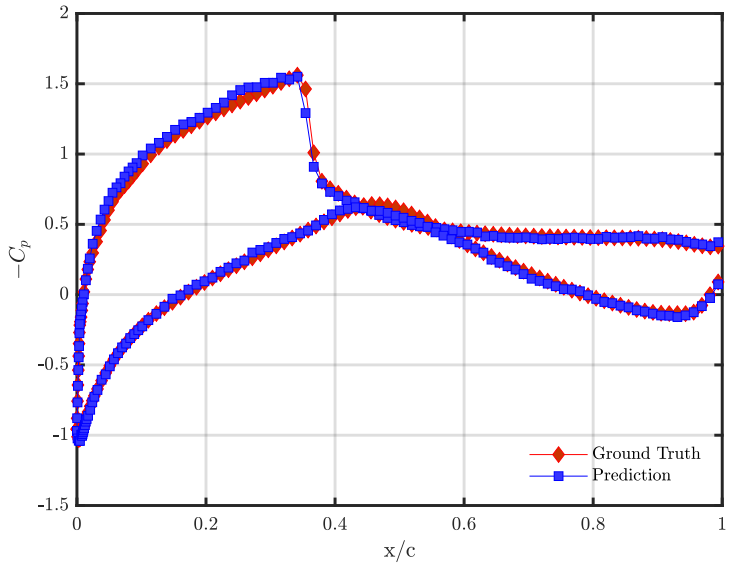


(b)

Figure 5.9: Surface pressure coefficient distribution of the test cases in Table 5.1. (a) NACA $64_1 - 112$, (b) NACA $65_1 - 212$.

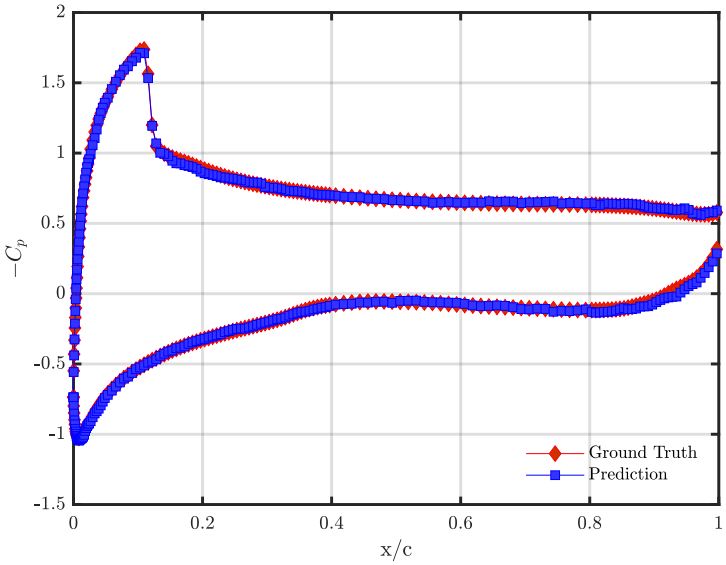


(a)

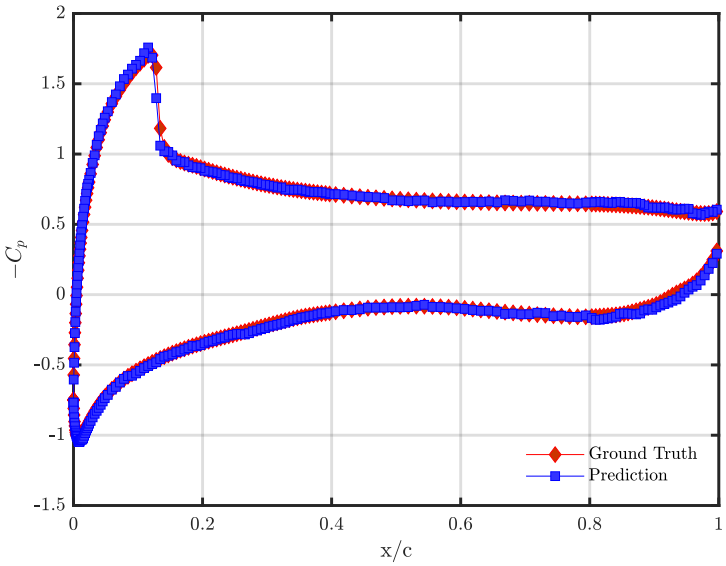


(b)

Figure 5.10: Surface pressure coefficient distribution of the test cases in Table 5.1. (a) NACA $65_4 - 221$, (b) NACA $65_4 - 421$.



(a)



(b)

Figure 5.11: Surface pressure coefficient distribution of the test cases in Table 5.1. (a) HQ 3013, (b) HQ 3513.

Table 5.5: Accuracy results of C_p and M predictions of Eppler 547 for different angles of attack.

AoA	C_p			M		
	$AWT(\delta = 1.1)(\%)$	$AWT(\delta = 1.2)(\%)$	$100 - MAPE$	$AWT(\delta = 1.1)(\%)$	$AWT(\delta = 1.2)(\%)$	$100 - MAPE$
$-8°$	98.077	99.656	91.932	99.622	99.829	99.385
$-4°$	99.473	99.851	95.659	99.864	99.956	99.685
$0°$	98.257	99.727	95.286	99.704	99.893	99.605
$4°$	99.400	99.861	93.655	99.672	99.930	99.472
$8°$	99.227	99.937	95.055	99.757	99.937	98.238
$14°$	99.135	99.898	97.711	99.801	99.932	99.119
$16°$	99.628	99.885	97.854	99.528	99.877	98.768

for the C_p and M fields of Eppler 547 for several angles of attack cases from the test set. The absolute differences between the ground truth and the model output are also included in Figure 5.12. The neural network model performs well in capturing flow field evolution as the angle of attack changes. The model simulates well the movement of the low-pressure region from the pressure to the suction side of the airfoil as the angle of attack changes from negative to positive even though almost 2/3 of the data consists of positive angle of attack results.

Flow separation starts from the trailing edge and moves toward the leading edge with the increasing angle of attack until the recirculation zone expands through the chord length. As the angle of attack is further increased, the flow will eventually separate almost from the leading edge. The shock location and strength, and the boundary layer/ separation location are also captured very well. Both C_p and M field predictions in Figure 5.12 show very good agreement with all the characteristic features for the angle of attack variations. As expected, maximum errors are observed at the separation location and around the shock wave. To overcome this issue, we can increase the resolution of the flow field; however, this will add extra cost to the training time. Table 5.5 presents the model performance evaluations for the cases given in Figure 5.12.

Figure 5.13 shows the C_p distributions on the Eppler airfoil surface. The model predictions agree well with the ground truth. For the negative angles of attack, a shock wave develops on the lower surface. As the angle of attack increases, the shock wave moves to the leading edge and becomes stronger. The evolution of C_p distribution,

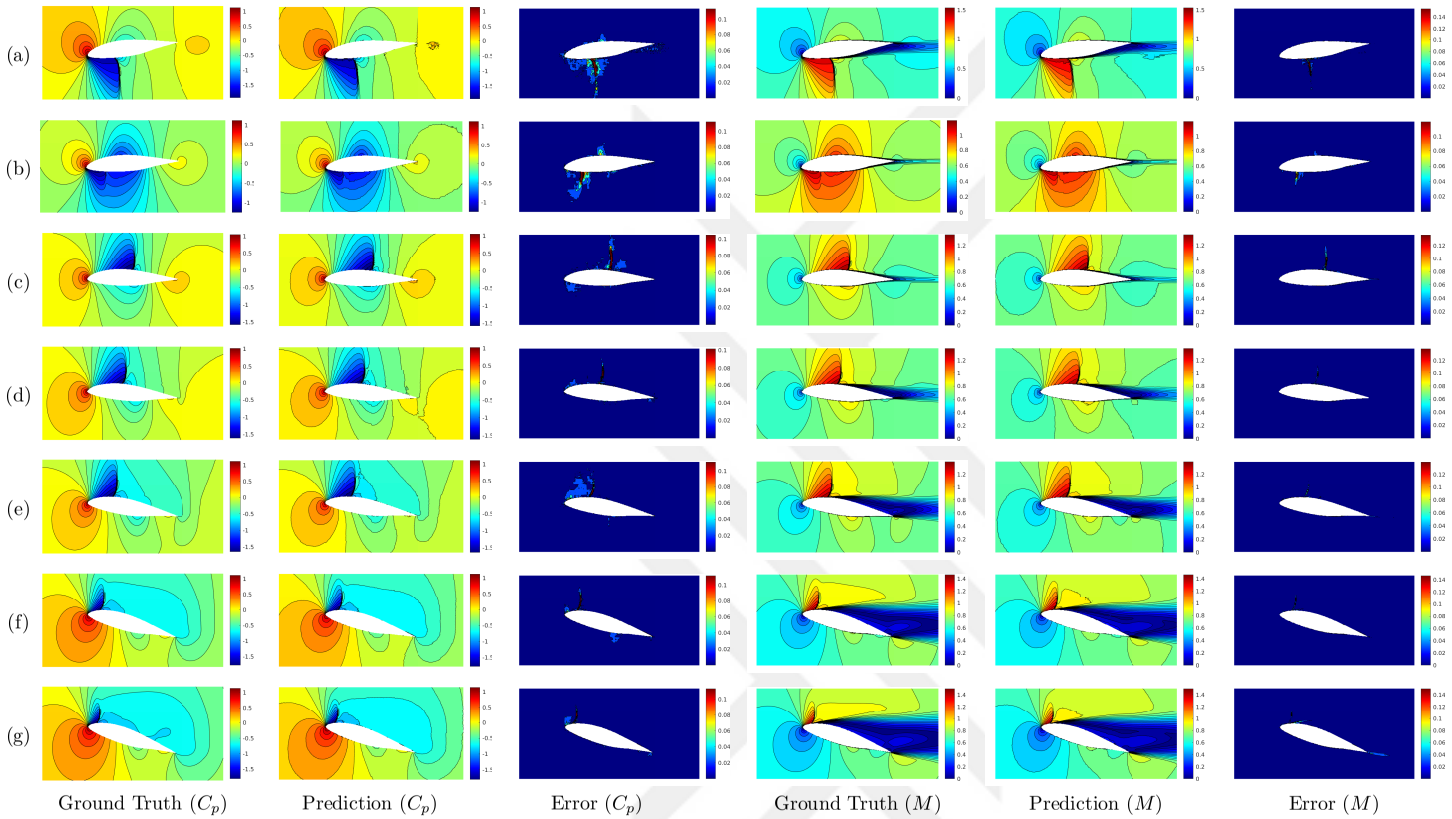


Figure 5.12: The comparison of the C_p and M fields of Eppler 547 for different angles of attack. (a) $\alpha = -8$, (b) $\alpha = -4$, (c) $\alpha = 0$, (d) $\alpha = 4$, (e) $\alpha = 8$, (f) $\alpha = 14$, (g) $\alpha = 16$.

including the shock waves, is well predicted. The predicted shock strength and sharpness are generally predicted well. The locations of the shocks are estimated within 2% of the chord (two pixels) compared to the ground truth. Note that the shift of the shock location results in large error estimations at these locations. Both location and the strength of the shocks are estimated well. It proves us the error plots given in Figure 5.13 may be misleading, and the prediction accuracy exceeds the expectations. The data points for the C_p distribution in practice results in a small deviation for the integration of the forces.

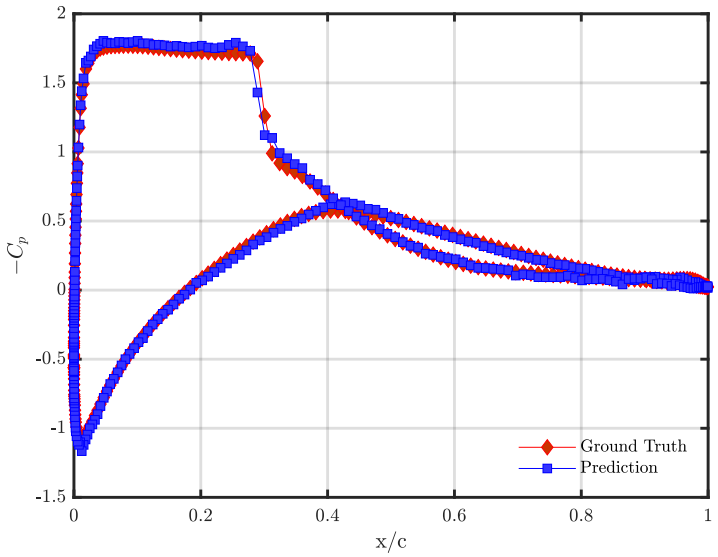
Furthermore, aerodynamic coefficients are extracted from the pressure coefficient distribution on the airfoil surface. The net aerodynamic force of an airfoil immersed in the fluid flow arises from the pressure and the shear stress distributions over the airfoil surface, as depicted on a surface element in Figure 5.14. The normal and axial components of aerodynamic force, F, can be computed by Eq. 5.5 and Eq. 5.6.

$$F_N = -\oint_{S_A} p n_y ds - \oint_{S_A} \tau n_y ds \tag{5.5}$$

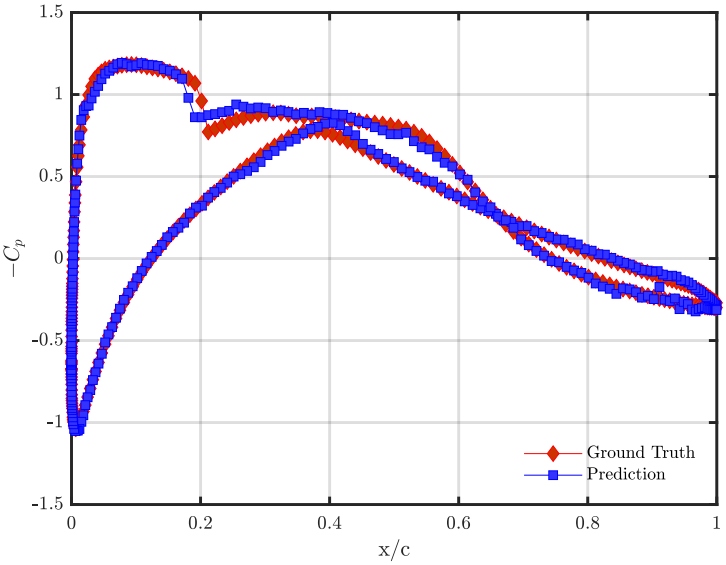
$$F_A = -\oint_{S_A} p n_x ds + \oint_{S_A} \tau n_x ds \tag{5.6}$$

where S_A is the airfoil surface area, τ is the shear stress and n_x, and n_y are the components of the normal vector of the surface element ds. It should be noted that only the pressure contribution on the net force is available due to the reduced resolution of the boundary layer. For wall shear stress resolution, the data should be represented with a much finer detail than the current 256×256 Cartesian grid. Besides, the results of our CFD analysis reveal that the pressure forces dominate the net aerodynamic forces. This is observed clearly for the lift and also for the drag forces at transonic speeds in more than a few degrees of angle of attack. Therefore, we present only the pressure contribution in Eq. 5.5 and Eq. 5.6. If desired, this contribution can be calculated approximately and added to the drag calculation as a further fix. In this study, we will keep our attention on the pressure-based drag and lift.

As mentioned before, for the training, we have used airfoil shapes rotated by α. In other words, the x-direction is the wind axis. Therefore, the force calculations in

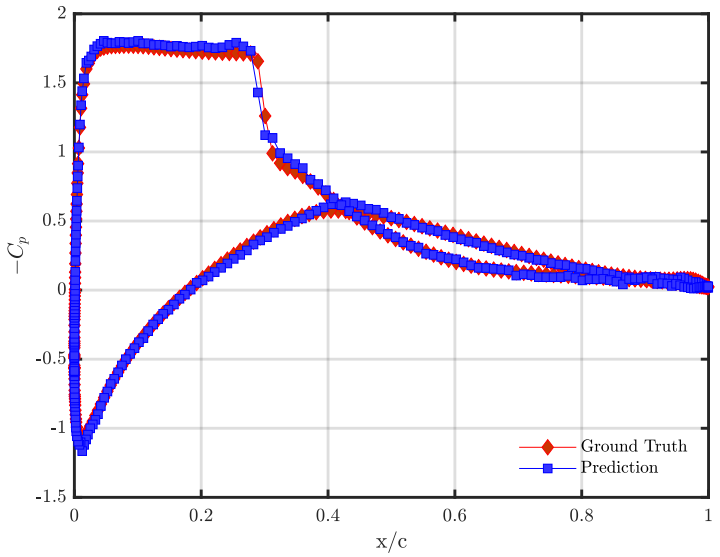


(a)

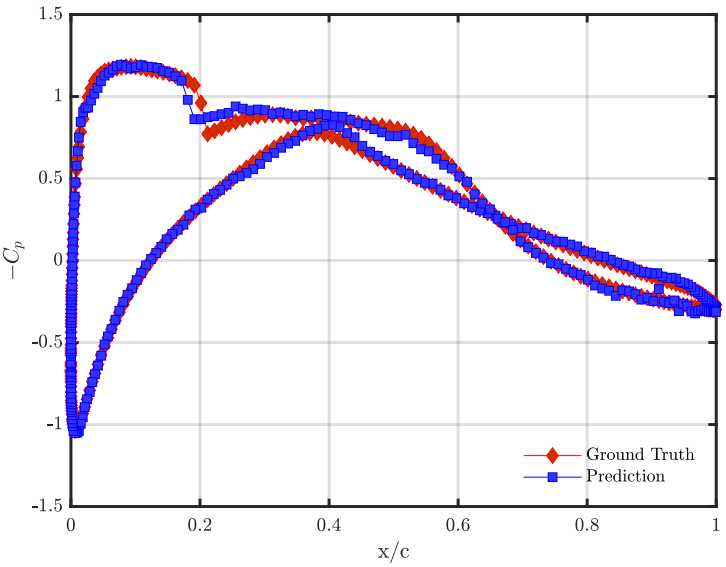


(b)

Figure 5.13: Surface pressure coefficient distribution on Eppler 547. (a) $\alpha = -8°$, (b) $\alpha = -4°$

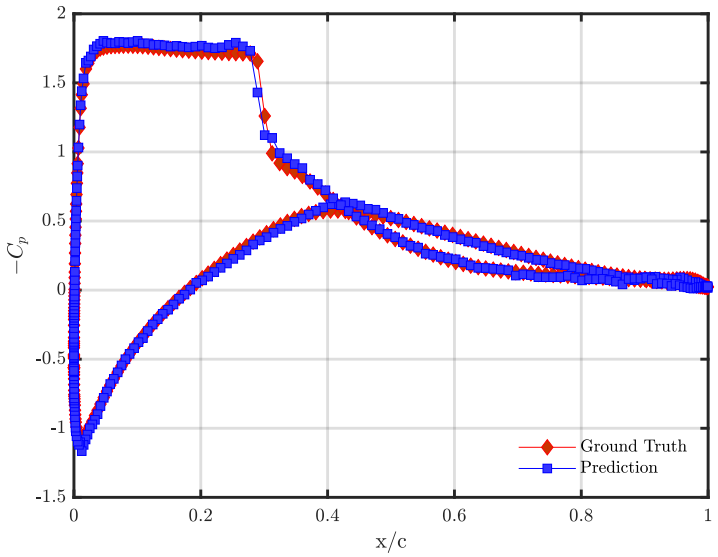


(c)

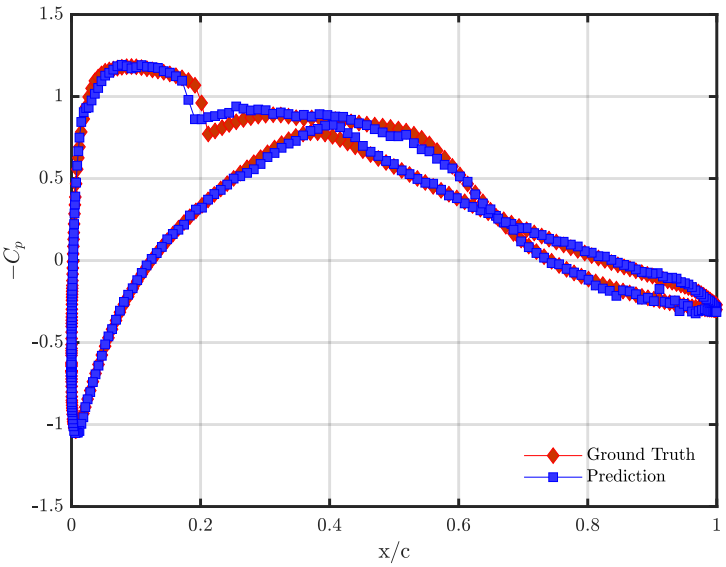


(d)

Figure 5.13: Surface pressure coefficient distribution on Eppler 547. (c) $\alpha = 0°$, (d) $\alpha = 4°$ (cont.)

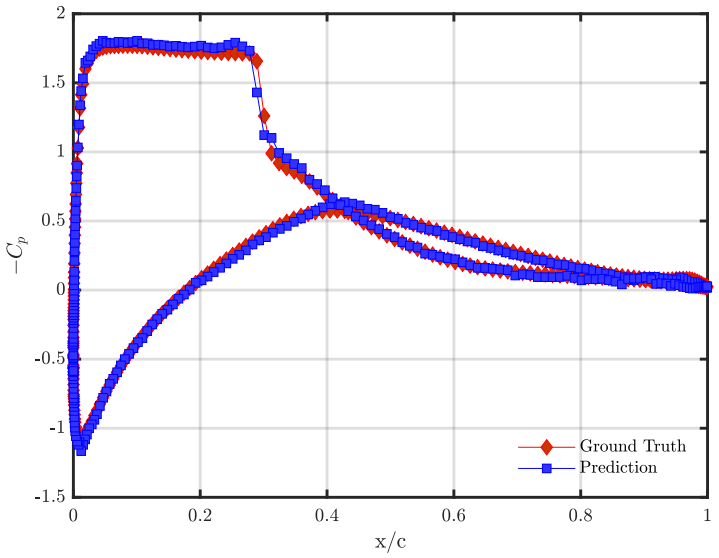


(e)



(f)

Figure 5.13: Surface pressure coefficient distribution on Eppler 547. (e) $\alpha = 8°$, (f) $\alpha = 14°$ (cont.)



(g)

Figure 5.13: Surface pressure coefficient distribution on Eppler 547. (g) $\alpha = 16°$ (cont.)

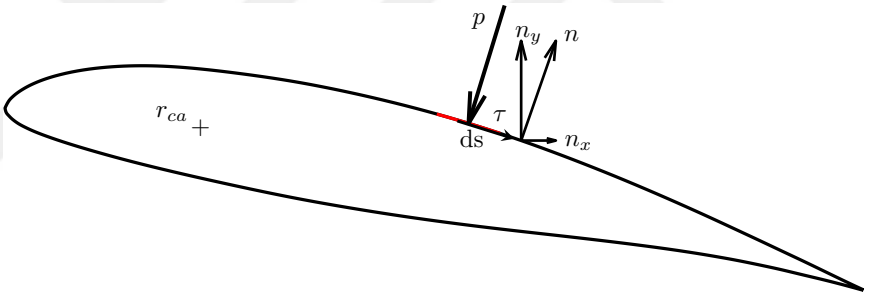


Figure 5.14: Pressure and shear stress on an airfoil surface element.

x directions yield pressure drag, and similarly, the normal component of the force corresponds to the lift. When we integrate the pressure coefficient distribution along the airfoil surface, we can obtain the lift and the pressure drag coefficients as follows

$$C_L = -\frac{1}{A_{ref}} \oint_{S_A} C_p n_y ds \tag{5.7}$$

$$C_{D_P} = -\frac{1}{A_{ref}} \oint_{S_A} C_p n_x ds \tag{5.8}$$

Also, the moment coefficient can be computed by integrating the moment of the pres-

Table 5.6: The comparison of aerodynamic coefficients between the model output and the ground truth with respect to the angle of attack variation for Eppler 547 airfoil.

AoA	$C_{L,true}$	$C_{L,prediction}$	$\|Error\|_{C_L}$ (%)	$C_{D_P,true}$	$C_{D_P,prediction}$	$\|Error\|_{C_{D_P}}$ (%)	$C_{M,true}$	$C_{M,prediction}$	$\|Error\|_{C_M}$ (%)
$-8°$	-0.537	-0.549	2.2	0.069	0.071	3.1	-0.078	-0.080	2.8
$-4°$	-0.304	-0.296	2.6	0.025	0.026	2.1	-0.049	-0.053	8.4
$0°$	0.306	0.302	1.2	0.025	0.027	8.2	-0.058	-0.056	3.4
$4°$	0.474	0.481	1.3	0.057	0.055	3.3	-0.048	-0.048	0.0
$8°$	0.628	0.638	1.6	0.106	0.105	1.3	-0.056	-0.056	1.2
$14°$	0.808	0.809	0.2	0.219	0.218	0.8	-0.093	-0.093	0.5
$16°$	0.855	0.853	0.3	0.267	0.264	1.2	-0.110	-0.108	1.8

sure coefficient distribution about the aerodynamic center. Note that the shear stress contribution on the moment is neglected due to the above-mentioned reasons. CFD results also reveals that the viscous contribution is small.

$$C_M = -\frac{1}{A_{ref} l_{ref}} \oint_{S_A} C_p \vec{n} \times (r - r_{ca}) ds \tag{5.9}$$

where r is the non-dimensional location along the chord length, r_{ca} denotes the aerodynamic center, which is the quarter-chord location. The reference area (A_{ref}) and the reference length (l_{ref}) are the chord length of the airfoil. Table 5.6 shows the calculated aerodynamic coefficients for the given cases. It is seen that there is a good agreement between the model predictions and the ground truth within a few percent, whereas some deviation is observed in the aerodynamic coefficients when the angle of attack decreases. This deviation is due to the increasing contribution of the viscous forces as the angle of attack decreases. The pressure difference, particularly around the shock wave as indicated in Figure 5.12, causes the variation in the aerodynamic coefficients.

5.2.3.2 NACA $66_3 - 218$

NACA $66_3 - 218$ is designed to maintain laminar flow, has the maximum thickness-to-chord ratio of 18 at 45% chord and the maximum camber-to-chord ratio of 1.1 at 50% chord. The results are shown in the following sequence similar to previous cases: C_p and M fields, surface pressure coefficient distribution, and aerodynamic coefficients.

Figure 5.15 shows the comparison between the ground truth and the model output for the C_p and M fields, including the corresponding absolute difference for each angle of attack. Although the overall flow features of NACA $66_3 - 218$ in Figure 5.15 are similar to Eppler 547. As the angle of attack increases, the shock-induced flow separation is observed. However, there are subtle differences with Eppler 547. The shock waves do not emerge as sharp as Eppler 547. Besides, the flow separation shifts toward the mid-chord of the airfoil, as can be seen from the common $\alpha = 8^\circ$ cases.

Figure 5.16 shows the C_p distributions on the airfoil surface. At $\alpha = -7^\circ$, the pressure results in a plateau-shaped distribution on the pressure side of the airfoil. Then, a suction peak is observed near the $x/c = 0.6$ at $\alpha = -1^\circ$. The pressure gradient near the $x/c = 0.6$ becomes stronger with a further increase in the angle of attack to $\alpha = 0^\circ$ and $\alpha = 1^\circ$. At $\alpha = 3^\circ$, a shock is observed which induces to flow separation as can also be seen in Figure 5.15. The shock has become stronger and shifted towards the leading edge at $\alpha = 8^\circ$. At $\alpha = 18^\circ$ and $\alpha = 19^\circ$, flow is massively separated from the airfoil. Although the neural network model has difficulty in accurately predicting the shock locations, network predictions are agreed reasonably well with the CFD simulations as given in the accuracy measurements in Table 5.7.

Table 5.7: Accuracy results of C_p and M predictions of NACA $66_3 - 218$ for different angles of attack.

AoA	C_p			M		
	$AWT(\delta = 1.1)(\%)$	$AWT(\delta = 1.2)(\%)$	$100 - MAPE$	$AWT(\delta = 1.1)(\%)$	$AWT(\delta = 1.2)(\%)$	$100 - MAPE$
-7°	98.675	99.639	92.312	99.722	99.903	99.394
-1°	98.525	99.748	94.269	99.639	99.818	99.555
0°	99.232	99.729	94.467	99.495	99.730	99.501
1°	98.296	99.789	94.019	99.672	99.890	99.545
3°	96.781	99.557	93.145	99.704	99.890	99.468
8°	98.913	99.602	94.702	99.477	99.800	99.257
18°	99.310	99.784	97.793	99.281	99.831	98.612
19°	98.993	99.665	97.900	99.644	99.872	98.747

Table 5.8 presents the aerodynamic coefficients for the NACA $66_3 - 218$ at various angles of attack. Please note that the force coefficients are obtained by integrating the pressure distribution along the airfoil surface, as explained in the previous experiment. There is a reasonable agreement between the model predictions and the ground truths. However, a large discrepancy is observed on the C_L for $\alpha = -1^\circ$ and

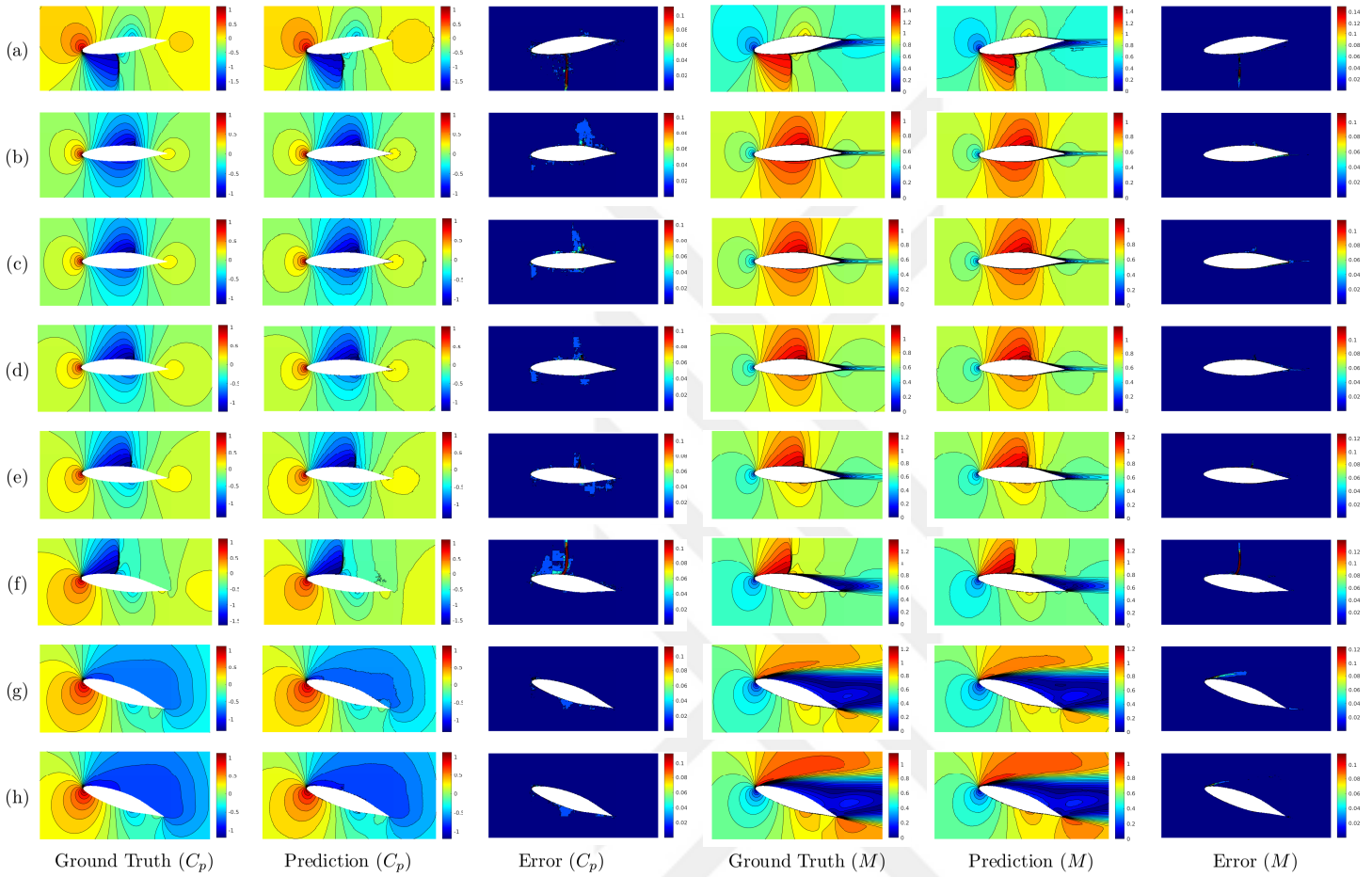
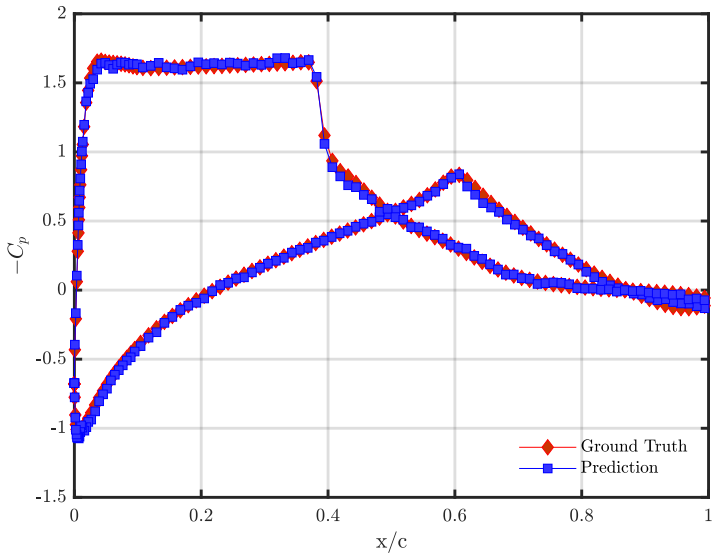
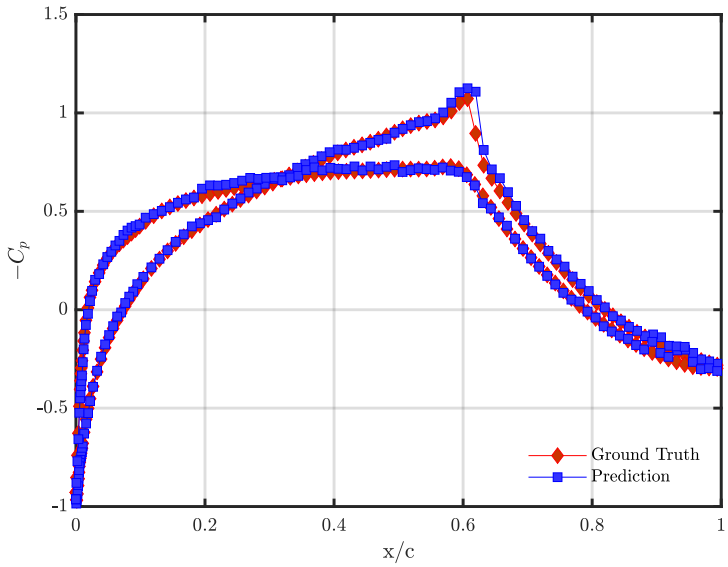


Figure 5.15: The comparison of the C_p and M fields of NACA $66_3 - 218$ for different angles of attack. (a) $\alpha = -7°$, (b) $\alpha = -1°$, (c) $\alpha = 0°$, (d) $\alpha = 1°$, (e) $\alpha = 3°$, (f) $\alpha = 8°$, (g) $\alpha = 18°$, (h) $\alpha = 19°$.

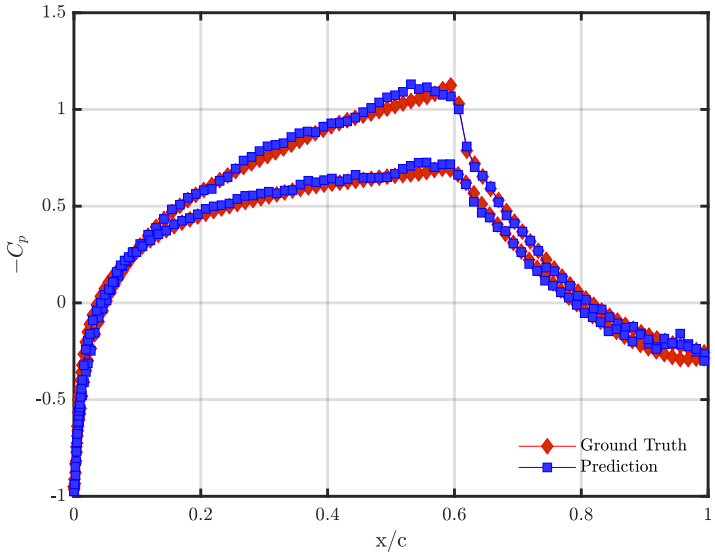


(a)

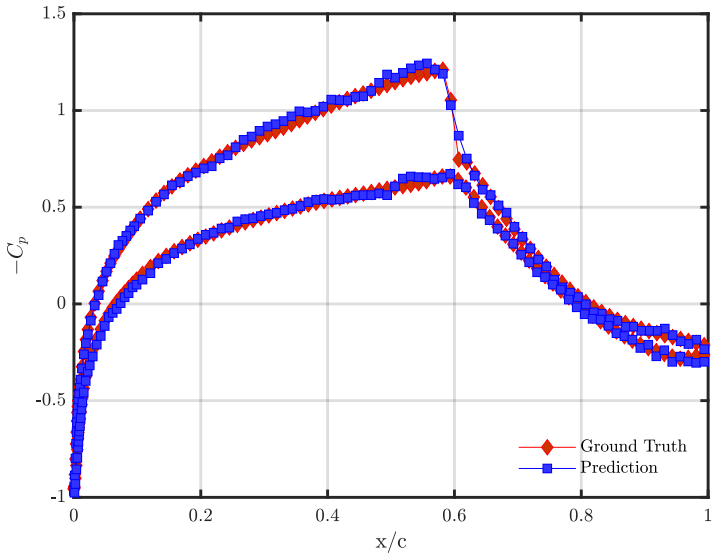


(b)

Figure 5.16: Surface pressure coefficient distribution on NACA $66_3 - 218$. (a) $\alpha = -7°$, (b) $\alpha = -1°$

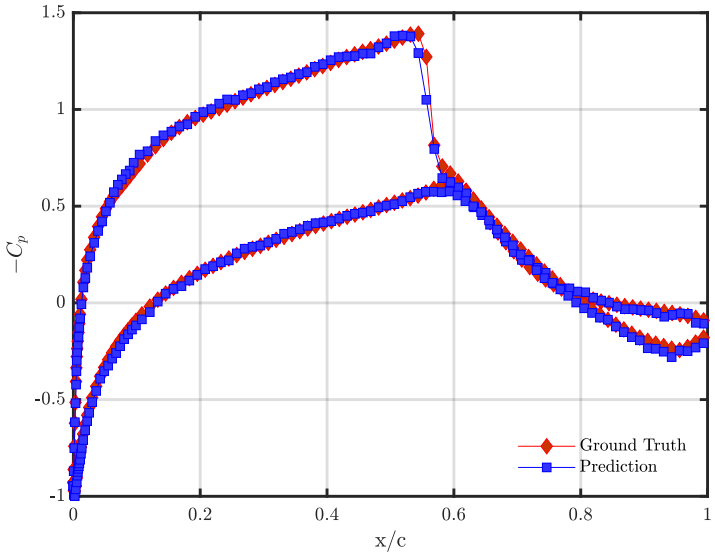


(c)

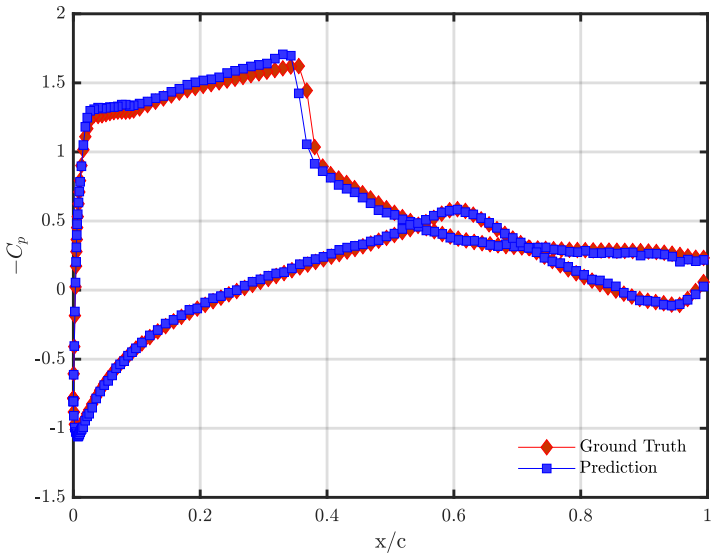


(d)

Figure 5.16: Surface pressure coefficient distribution on NACA 66_3-218. (c) $\alpha = 0°$, (d) $\alpha = 1°$ (cont.)

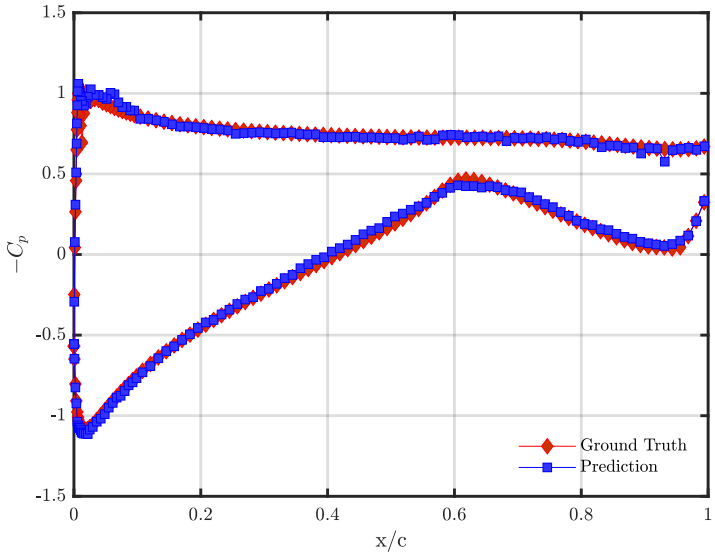


(e)

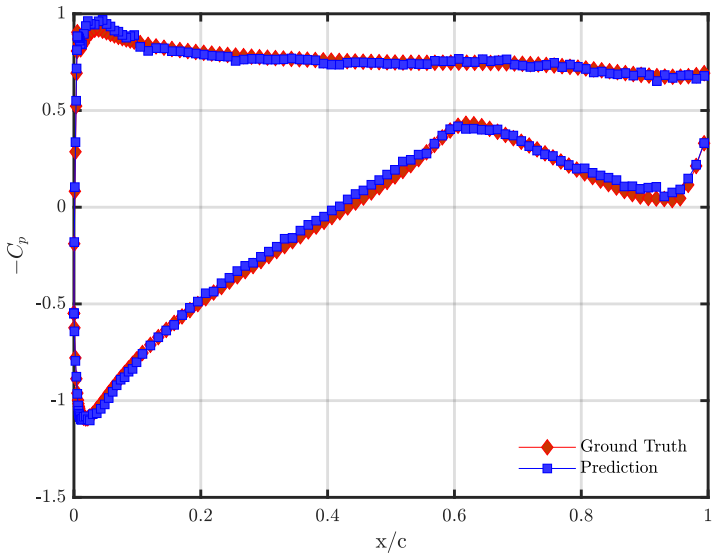


(f)

Figure 5.16: Surface pressure coefficient distribution on NACA 66_3-218. (e) $\alpha = 3°$, (f) $\alpha = 8°$ (cont.)



(g)



(h)

Figure 5.16: Surface pressure coefficient distribution on NACA $66_3 - 218$. (g) $\alpha = 18°$, (h) $\alpha = 19°$ (cont.)

Table 5.8: The comparison of aerodynamic coefficients between the model output and the ground truth with respect to the angle of attack variation for NACA $66_3 - 218$.

AoA	$C_{L,true}$	$C_{L,prediction}$	$\|Error\|_{C_L}$ (%)	$C_{D_P,true}$	$C_{D_P,prediction}$	$\|Error\|_{C_{D_P}}$ (%)	$C_{M,true}$	$C_{M,prediction}$	$\|Error\|_{C_M}$ (%)
$-7°$	-0.599	-0.605	1.1	0.054	0.055	2.3	-0.073	-0.074	0.6
$-1°$	0.016	0.023	42.3	0.018	0.020	13.9	-0.040	-0.044	8.9
$0°$	0.148	0.142	4.2	0.017	0.017	0.2	-0.041	-0.035	15.1
$1°$	0.271	0.285	5.2	0.018	0.020	6.7	-0.041	-0.044	6.8
$3°$	0.468	0.476	1.7	0.027	0.026	1.8	-0.036	-0.038	5.6
$8°$	0.703	0.696	1.0	0.080	0.076	5.1	-0.009	-0.003	69.6
$18°$	0.737	0.733	0.6	0.313	0.311	0.8	-0.092	-0.087	5.7
$19°$	0.765	0.750	2.0	0.340	0.335	1.4	-0.102	-0.095	6.9

the C_M for $\alpha = 8°$ in terms of relative percentage error. It appears that the NACA $66_3 - 218$ generates almost zero-lift at $\alpha = -1°$, as shown in Figure 5.15. Even though the results are fairly well predicted, the close-to-zero values have biased the relative percentage errors.

5.3 Out-of-dataset generalization performance

In order to give an insight on the generalization performance of the neural network model, we have tested the model on an airfoil shape has not seen before (out-of-dataset) besides held-out test set. NACA 0012 at $\alpha = 8°$ is used to test the model. This airfoil is a very standard one and is a part of the NACA $00-$ series. Please note that none of the NACA $00-$ series airfoils is included in the database. The predictions of C_p and M fields with surface C_p distribution are given in Figure 5.17. The results agree well with the ground truth. Shock location and separation are estimated well. As expected, errors are accumulated at the high gradient regions. This behavior is similar to previous experiments. We have achieved about 92 % and 99 % accuracy for C_p and M field predictions, respectively. The results show that the model can provide successful performance for the independent blind test. We argue that overfitting is not an issue and our model's predictive capability extends general airfoil shapes.

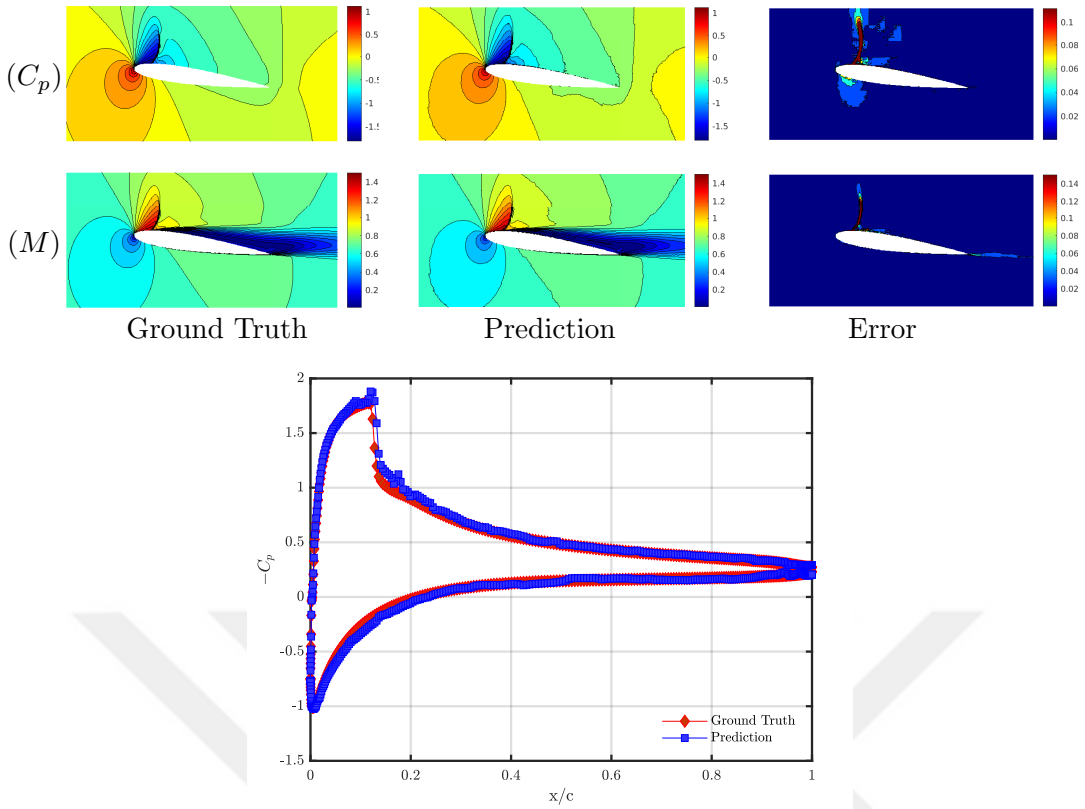


Figure 5.17: Network model predictions on NACA 0012 at $\alpha = 8°$.

5.4 Discussion of the Data Representation

Representation of CFD data is a central step for an effective data feed to neural networks. In this study, CFD data is projected onto the 256×256 Cartesian grid for an uncomplicated CNN implementation. However, the Cartesian grid projection costs an inevitable loss of accuracy due to the interpolation of grid data. This is particularly important in the vicinity of airfoil walls. Mesh refinement is necessary to capture sudden changes in the flow features generated by the no-slip boundary condition. Figure 5.18 shows the comparison of raw CFD data with interpolated data with respect to the surface pressure coefficient distribution on NACA $66_3 - 218$ surface at $\alpha = 8°$. Interpolated data agrees well with the raw CFD data. Please notice considerable deviations between the CFD data and the interpolated data at the suction side and around the stagnation location at the leading edge. Also, there is a slight shift on the shock location.

On the other hand, this deviation does not discredit the interpolated data. A com-

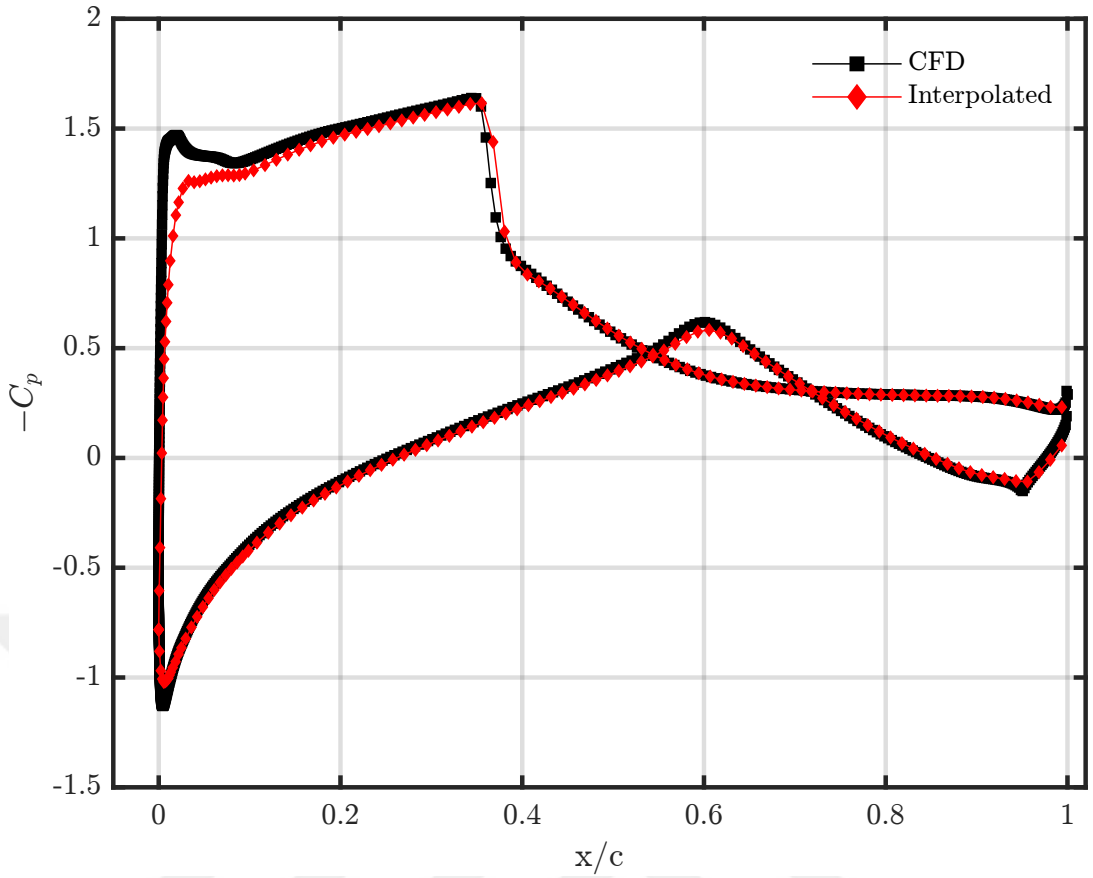


Figure 5.18: Comparison of the surface pressure coefficient distribution on NACA $66_3 - 218$ surface at $\alpha = 8°$.

parison of the lift coefficient and the drag polar is made between the CFD results, interpolated data and the model output in Figure 5.19 and 5.20, respectively. While the interpolated data shows good agreement with the CFD data for the lift coefficients, as shown in Figure 5.19, there are some offsets in the drag polar (see Fig. 5.20) in the drag direction. However, these offsets cannot be associated only with poor resolution in the leading edge. The main contributing factor is the lack of friction drag, as mentioned before. The model outputs agree well with the interpolated data (ground truth), as shown in Figures 5.19 and 5.20.

Figure 5.21 shows the histograms of the accuracy results of the whole test set with respect to the $100 - MAPE$. Overall, considering the $MAPE$ metric, we achieve overall 94 % and 99 % accuracy for C_p and M field predictions, respectively. Better performance is achieved in the M predictions compared to the C_p predictions. This difference stems from the fact that the C_p predictions are more sensitive to higher

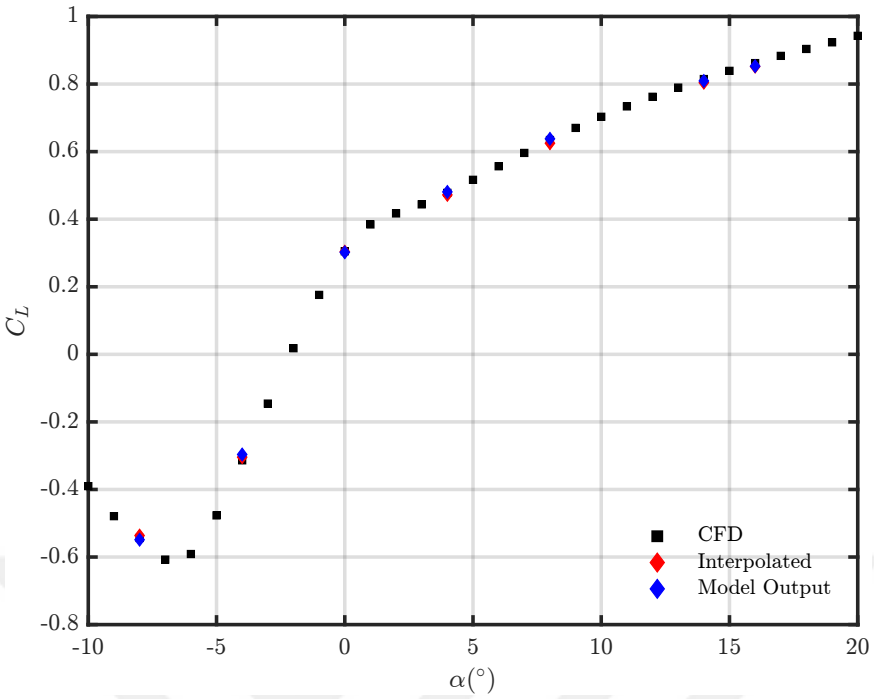


Figure 5.19: Comparison of the lift coefficient of NACA $66_3 - 218$.

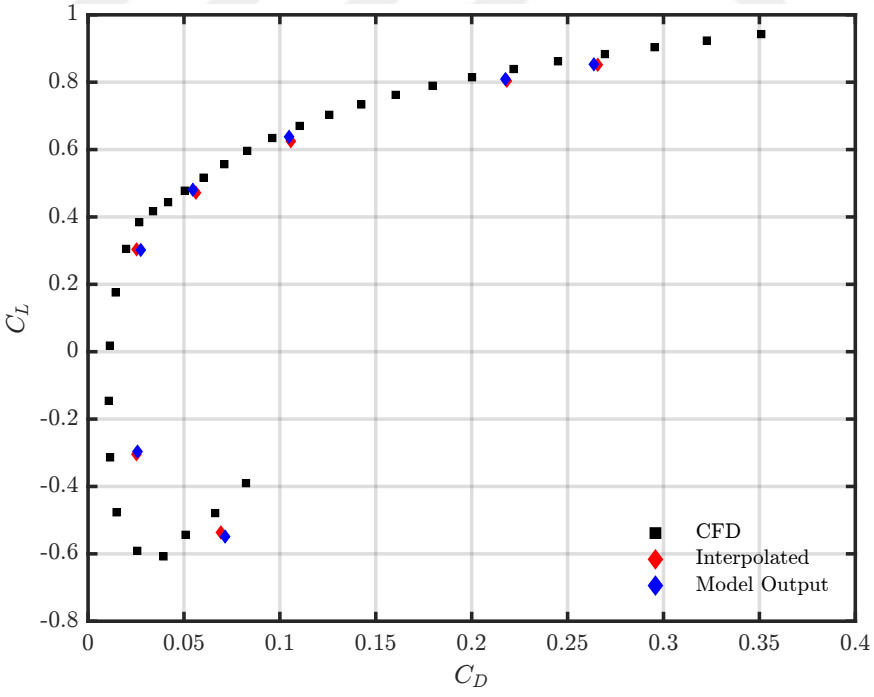
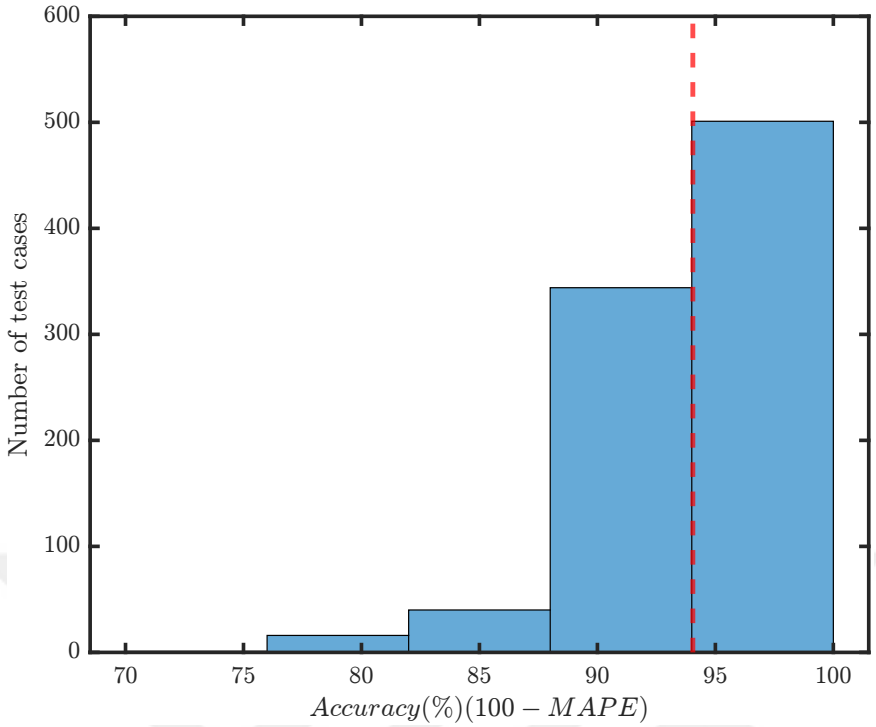


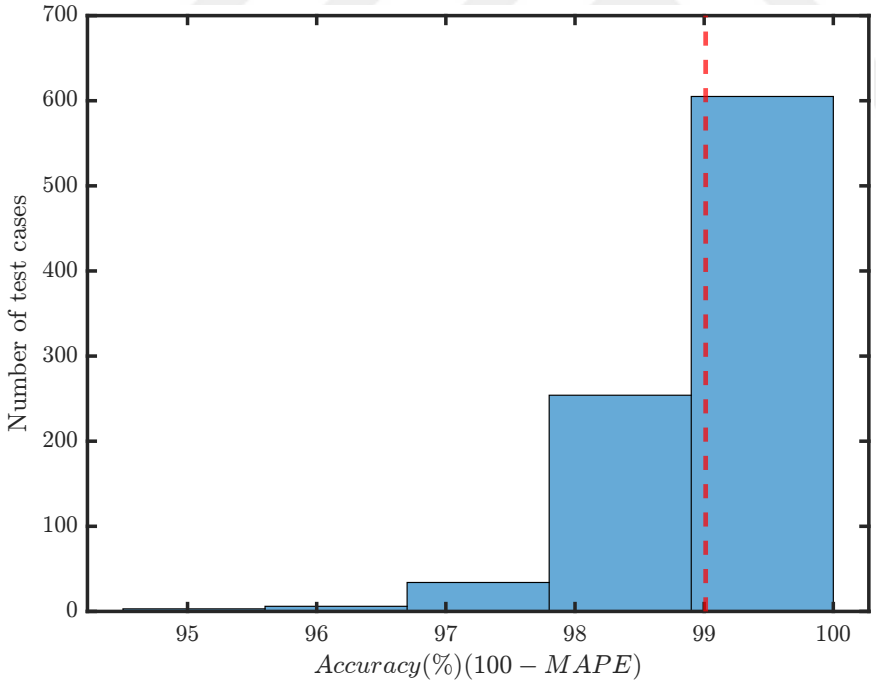
Figure 5.20: Drag polar of NACA $66_3 - 218$.

gradient regions, especially shocks than M predictions, as shown in Figures 5.3, 5.8, 5.12, 5.15. This is consistent with pressure-Mach relation for isentropic expansion and (normal and oblique) shocks for an ideal gas. Close to the sonic flow, pressure changes faster than the Mach number, as it can be observed from compressible flow tables. Therefore larger gradients, hence larger errors, on pressure coefficient are expected. Even a slight shift in the shock location can result in larger errors in C_p fields than in M fields.

Figure 5.21a indicates that in a few cases, the accuracy of the C_p predictions fell below 80 %. Therefore, we also investigate the worst case to assess the predictive ability of the model. Figure 5.22 depicts the predictions of C_p and M fields with surface C_p distribution of vr15 airfoil at $\alpha = 2^\circ$. As expected, the largest error is in the shock wave region. The shock location is estimated with a shift, which significantly affects the evaluation of the results. This deviation has also appeared on the C_p distribution along the airfoil surface. Nevertheless, the overall pattern is approximated quite well other than the shock wave region. We can argue that the neural network does an excellent job in learning the flow around airfoils for the given conditions. When we consider the nonlinearity of the problem, including discontinuities of shock waves and flow separation, our model performs remarkably well.



(a)



(b)

Figure 5.21: Histograms of the test set accuracy results. (a) C_p field, (b) M field. Red lines show the mean of the accuracy measurements.

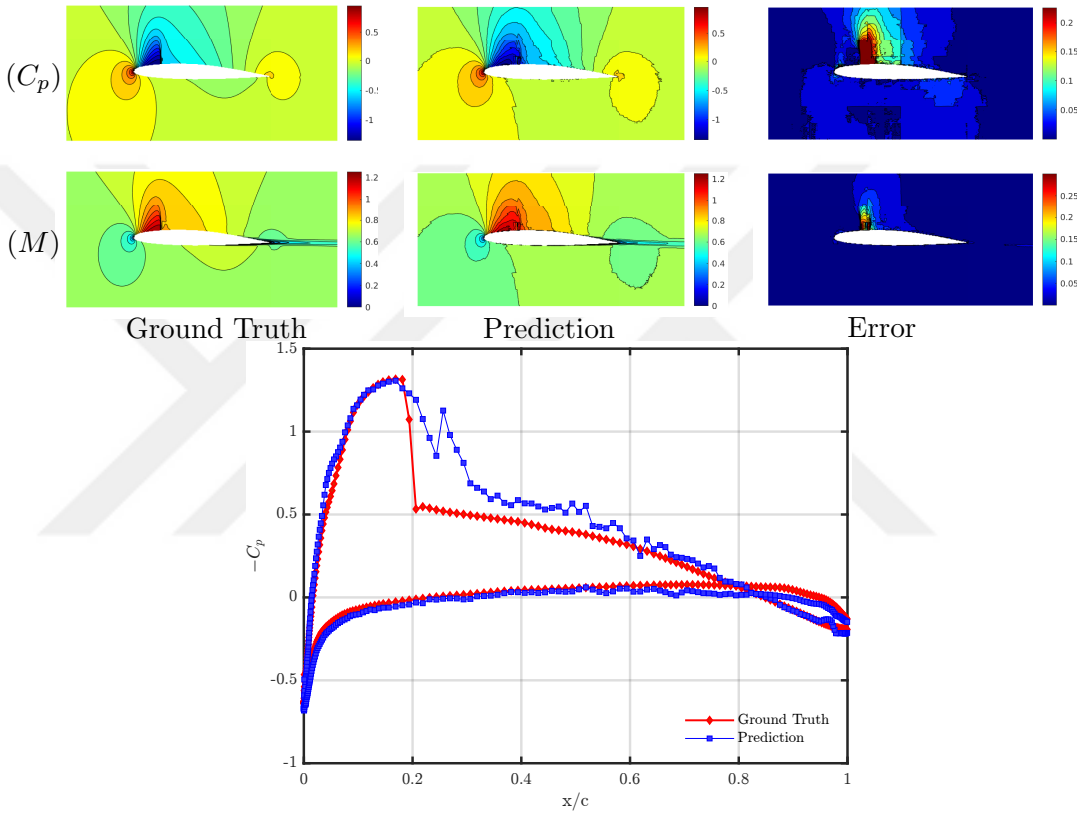


Figure 5.22: Network model predictions on vr15 airfoil at $\alpha = 2°$.

CHAPTER 6

CONCLUSIONS

In this thesis, a deep learning methodology is developed for the prediction of pressure and Mach fields around airfoils for the angle of attack range from -10° to 20° at free-stream Mach number, $M = 0.7$ and free-stream Reynolds number, $Re = 6 \times 10^6$. The developed methodology is based on an encoder-decoder convolutional neural network model. Each airfoil shape is, which is the input of the model, is represented with the distance field map for better shape learning. We have solved compressible Reynolds-averaged Navier-Stokes equations with a finite-volume in-house CFD solver to generate the database needed to train the deep learning model. CFD data is interpolated using a linear triangulation-based method in order to generate ground truth data onto a Cartesian grid, which corresponds to the distance field.

A careful network architecture search is conducted by focusing on the training performance of each candidate network for a suitable task. The training is performed in order to produce pressure coefficient field as a function of airfoil shape at a fixed angle of attack, $\alpha = 0^\circ$. The final form of the proposed neural network architecture has 8 blocks in both encoder and decoder parts. Each block consists of the same type of layers: convolutional or transposed convolutional layers, batch normalization layer (only for the encoder part) and exponential linear unit (ELU) activation layer except output layers. All convolution and transposed convolution operations are performed using a 2×2 filter size and a stride of 2. The neural network model is trained by utilizing Adam optimizer using a batch size of 8 and the learning rate of 10^{-4}.

After the training of the model is completed, the prediction performance of the model is evaluated by performing experiments utilizing the unseen cases in the test set. Several experiments are performed to study how the model responds to the effect of

changes to the airfoil characteristics. The shape effects are investigated by changing some key geometric parameters such as the maximum camber and the maximum thickness at the same angle of attack. Also, the angle of attack variation is studied for several airfoil shapes.

The predictive ability of the model is demonstrated by the flow field contours and validated various accuracy metrics quantitatively. The model outputs are compared with the ground truth by considering the absolute difference contour plots between the model output and the ground truth. When we compared with the CFD simulations, we observe that the model predictions are very accurate for both capturing effects of shape and angle of attack variations on the flow field. The errors are accumulated around the high gradient of C_p and M along the flow direction. Reminding that most of the errors are stemmed from the pixel location shift of the shocks, the neural network does an excellent job in learning the flow around airfoils.

Furthermore, aerodynamic coefficients are extracted from the pressure coefficient distribution on the airfoil surface. We keep our attention on the lift and the pressure-based drag due to the reduced resolution of the boundary layer. It is seen that there is a good agreement between the model predictions and the ground truth. Overall, considering the $MAPE$ metric, we achieve overall 94 % and 99 % accuracy for C_p and M field predictions, respectively.

The main contribution of this study is that the developed model made it possible to avoid time-consuming CFD simulations with a slight loss of accuracy. Table 6.1 shows the time cost of a single flow field prediction. Since the utilized CFD solver does not yet support GPU computing, a performance comparison in terms of computational cost is not employed. However, one could infer that notably, our neural network model is capable of reducing the computational cost dramatically. Actually, in terms of required hardware cost for the estimation of a single flow field, we have achieved almost three orders of speed-up with a much cheaper computational resource. We, therefore, propose this study as a contribution to the growing body of research to respond to the need of reducing the computational cost of the time-consuming CFD simulations having regard to the accuracy of the results. We strongly believe that this work has the potential to help researchers in design optimization stud-

Table 6.1: The time cost of a single flow field prediction

	Resource	**Time Cost**
CFD (per case)	32 Xeon E5-2690 cores	5100 s
Network Training	2×TESLA V100	360 h
Network Prediction	GTX 1050 Ti	0.9 s

ies.

6.1 Future Works

Further to this study, the presented model can be trained with a wide range of the Reynolds numbers and the Mach numbers to investigate the effects of the Reynolds number on the flow field in different flow regimes.

Also, physical loss functions that satisfy conservation laws, i.e., conservation of mass and momentum, could help the developed model to produce more physically accurate results. This ensures that each quantity such as pressure, velocity or density will be locally conserved at each pixel.

Graph convolutional neural networks can be utilized in order to achieve a better representation of the ground truth particularly in unstructured meshes.

On the other hand, using domain adaptation techniques and transfer learning approaches, it is possible to extend the model to new domains. Our proposed architecture and model parameters can be used as an initial starting point in a transfer learning setting. This would relieve the researchers from the time-consuming CFD simulations to some extent, reducing the amount of training data needed.

REFERENCES

[1] T Melin. Parametric airfoil catalog. *Linköping University,*, 2013.

[2] John P Fielding. *Introduction to aircraft design*, volume 11. Cambridge University Press, 2017.

[3] Ira H Abbott and Albert E Von Doenhoff. *Theory of wing sections: including a summary of airfoil data*. Courier Corporation, 2012.

[4] J Nathan Kutz. Deep learning in fluid dynamics. *Journal of Fluid Mechanics*, 814:1–4, 2017.

[5] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

[6] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *arXiv preprint arXiv:1703.00848*, 2017.

[7] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[8] Dong Yu and Li Deng. *AUTOMATIC SPEECH RECOGNITION*. Springer, 2016.

[9] Meiyin Wu and Li Chen. Image recognition based on deep learning. In *2015 Chinese Automation Congress (CAC)*, pages 542–546. IEEE, 2015.

[10] Eunsuk Chong, Chulwoo Han, and Frank C Park. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, 83:187–205, 2017.

[11] The brain: Understanding neurobiology. `https://science.education.nih.gov/supplements/webversions/`

BrainAddiction/guide/lesson2-1.html. Accessed: 2021-02-20.

[12] William E Faller and Scott J Schreck. Neural networks: applications and opportunities in aeronautics. *Progress in Aerospace Sciences*, 32(5):433–456, 1996.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

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

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

[16] Ze Jia Zhang and Karthikeyan Duraisamy. Machine learning methods for data-driven turbulence modeling. In *22nd AIAA Computational Fluid Dynamics Conference*, page 2460, 2015.

[17] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016.

[18] Anand Pratap Singh, Shivaji Medida, and Karthik Duraisamy. Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. *AIAA Journal*, pages 2215–2227, 2017.

[19] Linyang Zhu, Weiwei Zhang, Jiaqing Kou, and Yilang Liu. Machine learning methods for turbulence modeling in subsonic flows around airfoils. *Physics of Fluids*, 31(1):015105, 2019.

[20] Weishuo Liu and Jian Fang. Iterative framework of machine-learning based turbulence modeling for reynolds-averaged navier-stokes simulations. *arXiv preprint arXiv:1910.01232*, 2019.

[21] Romit Maulik, Himanshu Sharma, Saumil Patel, Bethany Lusch, and Elise Jennings. Accelerating rans turbulence modeling using potential flow and machine learning. *arXiv preprint arXiv:1910.10878*, 2019.

[22] Guido Novati, Hugues Lascombes de Laroussilhe, and Petros Koumoutsakos. Automating turbulence modelling by multi-agent reinforcement learning. *Nature Machine Intelligence*, pages 1–10, 2020.

[23] Cheng Yang, Xubo Yang, and Xiangyun Xiao. Data-driven projection method in fluid simulation. *Computer Animation and Virtual Worlds*, 27(3-4):415–424, 2016.

[24] Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. Accelerating eulerian fluid simulation with convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3424–3433. JMLR. org, 2017.

[25] Nils Wandel, Michael Weinmann, and Reinhard Klein. Unsupervised deep learning of incompressible fluid dynamics. *arXiv preprint arXiv:2006.08762*, 2020.

[26] Rajkumar Thirumalainambi and Jorge Bardina. Training data requirement for a neural network to predict aerodynamic coefficients. In *Independent Component Analyses, Wavelets, and Neural Networks*, volume 5102, pages 92–103. International Society for Optics and Photonics, 2003.

[27] Dilek Funda Kurtulus. Ability to forecast unsteady aerodynamic forces of flapping airfoils by artificial neural network. *Neural Computing and Applications*, 18(4):359, 2009.

[28] Dmitry I Ignatyev and Alexander N Khrabrov. Neural network modeling of unsteady aerodynamic characteristics at high angles of attack. *Aerospace Science and Technology*, 41:106–115, 2015.

[29] Tharindu P Miyanawala and Rajeev K Jaiman. An efficient deep learning technique for the navier-stokes equations: Application to unsteady wake flow dynamics. *arXiv preprint arXiv:1710.09099*, 2017.

[30] Emre Yilmaz and Brian German. A convolutional neural network approach to training predictors for airfoil performance. In *18th AIAA/ISSMO multidisciplinary analysis and optimization conference*, page 3660, 2017.

[31] Zelong Yuan, Yixing Wang, Yasong Qiu, Junqiang Bai, and Gang Chen. Aerodynamic coefficient prediction of airfoils with convolutional neural network. In *Asia-Pacific International Symposium on Aerospace Technology*, pages 34–46. Springer, 2018.

[32] Yao Zhang, Woong Je Sung, and Dimitri N Mavris. Application of convolutional neural network to predict airfoil lift coefficient. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 1903, 2018.

[33] Jonathan Viquerat and Elie Hachem. A supervised neural network for drag prediction of arbitrary 2d shapes in laminar flows at low reynolds number. *Computers & Fluids*, 210:104645, 2020.

[34] Xinyu Hui, Junqiang Bai, Hui Wang, and Yang Zhang. Fast pressure distribution prediction of airfoils using deep learning. *Aerospace Science and Technology*, 105:105949, 2020.

[35] Gang Sun, Yanjie Sun, and Shuyue Wang. Artificial neural network based inverse design: Airfoils and wings. *Aerospace Science and Technology*, 42:415–428, 2015.

[36] Emre Yilmaz and Brian German. A deep learning approach to an airfoil inverse design problem. In *2018 Multidisciplinary Analysis and Optimization Conference*, page 3420, 2018.

[37] Kumru Didem Atalay, Berna Dengiz, Tahir Yavuz, Emre Koç, and Yusuf Tansel İç. Airfoil–slat arrangement model design for wind turbines in fuzzy environment. *Neural Computing and Applications*, pages 1–9, 2020.

[38] S Ashwin Renganathan, Romit Maulik, and Jai Ahuja. Enhanced data efficiency using deep neural networks and gaussian processes for aerodynamic design optimization. *Aerospace Science and Technology*, 111:106522, 2021.

[39] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 481–490. ACM, 2016.

[40] Xiaowei Jin, Peng Cheng, Wen-Li Chen, and Hui Li. Prediction model of velocity field around circular cylinder over various reynolds numbers by fusion convolutional neural networks based on pressure on the cylinder. *Physics of Fluids*, 30(4):047105, 2018.

[41] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870:106–120, 2019.

[42] Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2):525–545, 2019.

[43] Junfeng Chen, Jonathan Viquerat, and Elie Hachem. U-net architectures for fast prediction of incompressible laminar flows. *arXiv preprint arXiv:1910.13532*, 2019.

[44] Sangseung Lee and Donghyun You. Data-driven prediction of unsteady flow over a circular cylinder using deep learning. *Journal of Fluid Mechanics*, 879:217–254, 2019.

[45] Vinothkumar Sekar, Qinghua Jiang, Chang Shu, and Boo Cheong Khoo. Fast flow field prediction over airfoils using deep learning approach. *Physics of Fluids*, 31(5):057103, 2019.

[46] Nils Thuerey, Konstantin Weißenow, Lukas Prantl, and Xiangyu Hu. Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal*, 58(1):25–36, 2020.

[47] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[48] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013.

[49] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[50] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[51] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[52] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[53] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[54] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[55] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[56] Philippe Spalart and Steven Allmaras. A one-equation turbulence model for aerodynamic flows. In *30th aerospace sciences meeting and exhibit*, page 439, 1992.

[57] Eleuterio F Toro, Michael Spruce, and William Speares. Restoration of the contact surface in the hll-riemann solver. *Shock waves*, 4(1):25–34, 1994.

[58] Venkat Venkatakrishnan. Convergence to steady state solutions of the euler equations on unstructured grids with limiters. *Journal of computational physics*, 118(1):120–130, 1995.

[59] James L Thomas and MD Salas. Far-field boundary conditions for transonic lifting solutions to theeuler equations. *AIAA journal*, 24(7):1074–1080, 1986.

[60] PH Cook, MCP Firmin, MA McDonald, and Royal Aircraft Establishment. *Aerofoil RAE 2822: pressure distributions, and boundary layer and wake measurements*. RAE, 1977.

[61] W Haase, F Brandsma, E Elsholz, MA Leschziner, and D Schwamborn. Euroval—a european initiative on validation of cfd codes—results of the ec. *BRITE-EURAM Project EUROVAL*, 1992, 1990.

[62] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[63] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[64] Michael Selig. Uiuc airfoil coordinates database. *UIUC Applied Aerodynamics Group, Urbana, IL, accessed May*, 18:2016, 2016.

[65] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[66] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[67] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[68] Robert J Renka. Interpolation of data on the surface of a sphere. *ACM Transactions on Mathematical Software (TOMS)*, 10(4):417–436, 1984.

[69] Der-Tsai Lee and Bruce J Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242, 1980.

[70] Cihat Duru, Hande Alemdar, and Özgür Uğraş Baran. Cnnfoil: convolutional encoder decoder modeling for pressure fields around airfoils. *Neural Computing and Applications*, pages 1–15, 2020.

[71] Ryohei Kuga, Asako Kanezaki, Masaki Samejima, Yusuke Sugano, and Yasuyuki Matsushita. Multi-task learning using multi-modal encoder-decoder networks with shared skip connections. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 403–411, 2017.

[72] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

[73] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5162–5170, 2015.

[74] Frank M White and Isla Corfield. *Viscous fluid flow*, volume 3. McGraw-Hill New York, 2006.

CURRICULUM VITAE

EDUCATION

Degree	Instituon	Year of Graduation
Ph.D.	METU-Mechanical Eng.	2021
M.Sc.	TOBB ETU-Mechanical Eng.	2015
B.Sc.	Hacettepe U.- Mechanical/Automotive Eng.	2012
High School	Aksaray Anatolian Teacher High School	2006

EXPERIENCE

Year	Place	Enrollment
Sep. 2017 – Oct. 2021	METU	Coordinator Assistant
Nov. 2013 – Oct. 2021	METU	Teaching Assistant
Oct. 2016 – Oct. 2017	TürkTraktör A.Ş.	Consultant
Sep. 2012 – Nov. 2013	TOBB ETU	Teaching Assistant
Jul. 2011 – Aug. 2011	Mercedes Benz Türk A.Ş.	Summer Intern
Jun. 2011 – Jul. 2011	Mercedes Benz Türk A.Ş.	Summer Intern
Aug. 2010 – Sep. 2010	Mercedes Benz Türk A.Ş.	Summer Intern
Aug. 2009 – Aug. 2009	Mercedes Benz Türk A.Ş.	Summer Intern

PUBLICATIONS

Peer-Reviewed Journal Articles

- Duru, C., Alemdar, H., Baran, Ö. U., "*A deep learning approach for the transonic flow field predictions around airfoils*", (Under Review)

- Duru, C., Alemdar, H., Baran, Ö. U., "*CNNFOIL: convolutional encoder decoder modeling for pressure fields around airfoils*", Neural Computing and Applications, 2020.

Conference Proceedings

- Doğan, A.,Duru, C., Alemdar, H., Baran, Ö. U., "*The effect of loss functions on the deep learning modeling for the flow field predictions around airfoils*", 11st Ankara International Aerospace Conference, Sep. 8-10, 2021, Ankara, Turkey.
- Duru, C., Aktaş, M. K., "*Control of heat transfer in a water filled enclosure with a vibrating side wall*", Proceedings of CONV-14: International Symposium on Convective Heat and Mass Transfer, June 8-13, 2014, Kuşadası, Turkey.

ATTENDED ACTIVITIES

- Lecture Series on "Machine Learning for Fluid Mechanics: Analysis, Modeling, Control and Closures", Feb. 24-28, 2020, von Karman Instıtute for Fluid Dynamics.
- "1st Workshop on High Performance Computing and Applications", Dec. 21, 2019, METU.