

A CONVOLUTIONAL NEURAL NETWORK METHODOLOGY WITH A
MOMENTUM-FLUX-BASED LOSS FUNCTION FOR PREDICTING
AERODYNAMIC FLOW AROUND AIRFOILS

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

BY

MUSTAFA MERT DENİZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

DECEMBER 2023

Approval of the thesis:

**A CONVOLUTIONAL NEURAL NETWORK METHODOLOGY WITH A
MOMENTUM-FLUX-BASED LOSS FUNCTION FOR PREDICTING
AERODYNAMIC FLOW AROUND AIRFOILS**

submitted by **MUSTAFA MERT DENİZ** in partial fulfillment of the requirements
for the degree of **Master of Science in Mechanical Engineering, Middle East
Technical University** by,

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

Prof. Dr. M. A. Sahir Arıkan
Head of the Department, **Mechanical Engineering** _____

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

Assoc. Prof. Dr. Hande Alemdar
Co-Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Metin Yavuz
Mechanical Engineering, METU _____

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

Assoc. Prof. Dr. Hande Alemdar
Computer Engineering, METU _____

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

Assist. Prof. Dr. Ali Karakuş
Mechanical Engineering, METU _____

Date: 07.12.2023

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 Last name: MUSTAFA MERT DENİZ

Signature:

ABSTRACT

A CONVOLUTIONAL NEURAL NETWORK METHODOLOGY WITH A MOMENTUM-FLUX-BASED LOSS FUNCTION FOR PREDICTING AERODYNAMIC FLOW AROUND AIRFOILS

Deniz, Mustafa Mert
Master of Science, Mechanical Engineering
Supervisor: Assist. Prof. Dr. Özgür Uğraş Baran
Co-Supervisor: Assoc. Prof. Dr. Hande Alemdar

December 2023, 75 pages

The selection of critical components like aircraft wings can be time-consuming due to the high computational costs associated with flow simulations. Deep machine learning techniques can predict flow domain and desired parameters and coefficients at a significantly lower cost. However, A deep learning prediction often ignores the physical processes that form the flow. Suppose a deep learning network that is informed about the physics underlying the trained scenario can be developed. In that case, this algorithm can offer a tool that can more accurately predict the flow around objects. A model problem involving flow around an airfoil is proposed to test this idea. Lift and drag forces on this airfoil, together with the flow domain around the airfoil, are the prediction parameters. A convolutional neural network model is developed as the prediction tool. The loss function is enhanced by a new conservation of momentum-based loss function to improve the fidelity of the model. The addition of conservation of momentum improves the lift and drag predictions around the aircraft significantly. The flow database is prepared with compressible CFD runs. To improve the accuracy, the CFD solver and the loss function calculation utilize the same flux function. The predictions after the training showed improved

lift and drag estimations compared to simple loss functions in a fraction of the time and cost of CFD calculations.

Keywords: Deep Learning, Convolutional Neural Network, HLLC Riemann Solver, Flow Prediction, Preliminary Design.

ÖZ

UÇAK KANATLARI ETRAFINDA AERODİNAMİK AKIŞ TAHMİNİ İÇİN MOMENTUM-AKI-BAZLI KAYIP FONKSİYONLU CONVOLUTIONAL SİNİR AĞI METODOLOJİSİ

Deniz, Mustafa Mert
Yüksek Lisans, Makina Mühendisliği
Tez Yöneticisi: Dr. Özgür Uğraş Baran
Ortak Tez Yöneticisi: Doç. Dr. Hande Alemdar

Aralık 2023, 75 sayfa

Uçak kanatları gibi kritik bileşenlerin seçimi, akış simülasyonlarıyla ilgili yüksek hesaplama maliyetleri nedeniyle zaman alıcı olabilir. Derin makine öğrenimi teknikleri, akış alanını, istenen parametreleri ve aerodinamik katsayıları önemli ölçüde daha düşük bir maliyetle tahmin edebilir. Ancak derin öğrenme tahmini genellikle akışı oluşturan fiziksel süreçleri göz ardı eder. Eğitilen senaryonun altında yatan fizik hakkında bilgi sahibi olan bir derin öğrenme ağının geliştirilebileceğini varsayalım. Bu durumda bu algoritma, nesnelerin etrafındaki akışı daha doğru bir şekilde tahmin edebilecek bir araç sunabilir. Bu fikri test etmek için bir kanat profili etrafındaki akışı içeren bir model problem önerilmiştir. Bu kanat profili üzerindeki kaldırma ve sürüklenme kuvvetleri, kanat profili etrafındaki akış alanıyla birlikte geliştirilen modelin tahmin parametreleridir. Tahmin aracı olarak convolutional bir sinir ağı modeli geliştirilmiştir. Kayıp fonksiyonu, modelin doğruluğunu artırmak için momentum korunumu bazlı yeni bir kayıp fonksiyonu ile geliştirildi. Momentum korunumunun eklenmesi, uçağın etrafındaki kaldırma ve sürüklenme tahminlerini önemli ölçüde artırır. Akış veritabanı sıkıştırılabilir CFD çalışmalarıyla hazırlandı. Doğruluğu geliştirmek için CFD çözücü ve kayıp

fonksiyonu hesaplaması aynı akı fonksiyonunu kullanır. Eğitimden sonraki analizler, CFD hesaplamalarının zaman ve maliyetinden çok daha az bir sürede, basit kayıp fonksiyonlarına kıyasla daha iyi kaldırma ve sürüklenme kuvveti tahminleri gösterdi.

Anahtar Kelimeler: Derin Öğrenme, Convolutional Sinir Ağları, HLLC Riemann Çözücüsü, Akış Tahmini, Ön Tasarım

To my family

ACKNOWLEDGMENTS

I would like to thank the heroes of this major event that changed my life and perspective, my thesis advisor Assist. Prof. Dr. Özgür Uğraş BARAN, Assoc. Prof. Dr. Hande ALEMDAR. I also would like to thank Assist. Prof. Dr. Ali KARAKUŞ, who accepted me as a temporary student of his during the pandemic.

I would like to express my deepest gratitude to Dr. Cihat DURU for his guidance, encouragement, and insight throughout the research.

I feel deeply grateful to my friend Adnan Harun DOĞAN, for assisting and helping me throughout my graduate program.

Lastly, I am very grateful to my manager at Roketsan, Mr. Kaan İNAL, and to the Pyrotechnic Systems Design Team for their continuous support.

TABLE OF CONTENTS

ABSTRACT.....	v
ÖZ	vii
ACKNOWLEDGMENTS	x
TABLE OF CONTENTS.....	xi
LIST OF TABLES	xiv
LIST OF FIGURES	xv
LIST OF ABBREVIATIONS.....	xviii
CHAPTERS	
1 INTRODUCTION	1
1.1 Aim of This Study.....	4
1.2 Airfoil Physics and Performance Criteria	5
2 DATA PREPARATION	7
2.1 Problem Definition.....	7
2.2 Conservation Laws.....	8
2.3 CFD Methodology	10
3 HLLC APPROXIMATE RIEMANN SOLVER & MOMENTUM FLUX- BASED LOSS FUNCTION	13
3.1 The HLL and HLLC Riemann Solvers	13
3.1.1 The Riemann Problem	13
3.1.2 The Godunov Flux	14
3.1.3 Integral Relations & Consistency Condition	15

3.1.4	The HLLC Approximate Riemann Solver	17
3.1.5	Wave Speed Calculations	20
3.2	Calculation of Momentum Fluxes with HLLC Approximate Riemann Solver	21
4	CNN ARCHITECTURE & NETWORK TRAINING.....	25
4.1	CNN Methodology	25
4.2	Modified-CNNFOIL Network Details	26
4.3	Network Training	27
4.3.1	Numerical and Physics-based Loss Functions	28
4.3.2	Dataset Splitting and Normalization Method	29
4.3.3	Training Procedure, Number of Epochs and Accuracy Metric ...	29
5	FLOW FIELD VARIABLES, MOMENTUM AND COEFFICIENT PREDICTIONS	33
5.1	Effect of Model Training on Flow Field Predictions	33
5.2	Effects of Model Training on Momentum Conservation	45
5.3	Drag and Lift Coefficients Comparison Between CFD Results and the Ground Truth	55
5.3.1	Drag and Lift Calculation Methodology, Selection of the Rectangular Domains	55
5.3.2	Effects of Model Training on Drag & Lift Coefficients.....	63
6	CONCLUSION AND FUTURE WORKS.....	67
	REFERENCES	69
	APPENDICES	73
A.	HLLC Mass and Momentum Flux Calculation Function, MATLAB Script	73

B. Encoder-Decoder Type of Network, Python Script..... 75

LIST OF TABLES

TABLES

Table 4.1 Training Parameters and the Details	27
Table 4.2 Training Process with Both Numerical and Physics-based Loss Functions	29

LIST OF FIGURES

FIGURES

Figure 1.1. The angle of Attack Between an Airfoil and Relative Wind.....	6
Figure 2.1. Computational Domain and Mesh Structure [24].....	10
Figure 2.2. Pressure Distribution of E342 Airfoil, Rectangular Stretched Domain	12
Figure 3.1. Structure of the Solution of the Riemann Problem for Time-Dependent Euler Equations	14
Figure 3.2. New Form of the Control Region on x-t Plane.....	15
Figure 3.3. Structure of the HLLC Approximate Riemann Solver.....	18
Figure 3.4. Momentum Fluxes Representation at Each Pixel.....	23
Figure 4.1. CNN Input, Filter, and Output Representation.....	25
Figure 4.2. Mean Absolute Errors for Primitive Variables for both Training and Validation Set.....	31
Figure 5.1. NACA747a415 Pressure Predictions at 0° Angle of Attack a) non-Physics-based Training, b) Physics-based Training, c) GT, d) non-Physics-based Predictions & GT (MAPE / 100), e) Physics-based Predictions & GT (MAPE / 100)	36
Figure 5.2. NACA747a415 Pressure Predictions at 8° Angle of Attack a) non-Physics-based Training, b) Physics-based Training, c) GT, d) non-Physics-based Predictions & GT (MAPE / 100), e) Physics-based Predictions & GT (MAPE / 100)	37
Figure 5.3. NACA2421 Temperature Predictions at 4° Angle of Attack a) non-Physics-based Training, b) Physics-based Training, c) GT, d) non-Physics-based Predictions & GT (MAPE / 100), e) Physics-based Predictions & GT (MAPE / 100)	39
Figure 5.4. NACA2421 Temperature Predictions at 12° Angle of Attack a) non-Physics-based Training, b) Physics-based Training, c) GT, d) non-Physics-based Predictions & GT (MAPE / 100), e) Physics-based Predictions & GT (MAPE / 100)	40

Figure 5.5. SC1012R8 Velocity Predictions at 6° Angle of Attack a) non-Physics-based Training, b) Physics-based Training, c) GT, d) non-Physics-based Predictions & GT (MAPE / 100), e) Physics-based Predictions & GT (MAPE / 100)..... 42

Figure 5.6. SC1012R8 Velocity Predictions at 14° Angle of Attack a) non-Physics-based Training, b) Physics-based Training, c) GT, d) non-Physics-based Predictions & GT (MAPE / 100), e) Physics-based Predictions & GT (MAPE / 100)..... 43

Figure 5.7. Percentage Errors of a) Pressure in NACA747a415, b) Temperature in NACA2421, c) Velocity in SC1012r8 for Different Angles of Attack Values..... 44

Figure 5.8. NPL9626 at 0° Angle of Attack, X-Momentum Conservation Law Outputs for a) Numerical Training, b) Physics-based Training, c) GT, d) Physics-based & GT Difference (Non-Dimensionalized by $12\rho V_\infty^2$)..... 47

Figure 5.9. NPL9626 at 8° Angle of Attack, X-Momentum Conservation Law Outputs for a) Numerical Training, b) Physics-based Training, c) GT, d) Physics-based & GT Difference (Non-Dimensionalized by $12\rho V_\infty^2$)..... 48

Figure 5.10. NPL9626 at 16° Angle of Attack, X-Momentum Conservation Law Outputs for a) Numerical Training, b) Physics-based Training, c) GT, d) Physics-based & GT Difference (Non-Dimensionalized by $12\rho V_\infty^2$)..... 49

Figure 5.11. NPL9626 at 0° Angle of Attack, Y-Momentum Conservation Law Outputs for a) Numerical Training, b) Physics-based Training, c) GT, d) Physics-based & GT Difference (Non-Dimensionalized by $12\rho V_\infty^2$)..... 52

Figure 5.12. NPL9626 at 8° Angle of Attack, Y-Momentum Conservation Law Outputs for a) Numerical Training, b) Physics-based Training, c) GT, d) Physics-based & GT Difference (Non-Dimensionalized by $12\rho V_\infty^2$)..... 53

Figure 5.13. NPL9626 at 16° Angle of Attack, Y-Momentum Conservation Law Outputs for a) Numerical Training, b) Physics-based Training, c) GT, d) Physics-based & GT Difference (Non-Dimensionalized by $12\rho V_\infty^2$)..... 54

Figure 5.14. X & Y Momentum Losses of Both Training and Validation Set..... 55

Figure 5.15. The Rectangles Utilized in Drag and Lift Calculations 57

Figure 5.16. E347 at AOA=10. GT vs CFD results. a) CD, b) CL 59

Figure 5.17. a) CD, b) CL Comparison between 16 Rectangles and 60 Rectangles Averaging Operation.....	60
Figure 5.18. a) E347, b) NACA654221, c) RAF69 Rectangular Regions (Resolution) Comparisons for CD and CL	62
Figure 5.19. Drag & Lift Coefficient Predictions of FX61184, M12, and RAE5215	65
Figure 5.20. Drag Polar Curves of FX61184, M12, and RAE5215.....	66

LIST OF ABBREVIATIONS

ABBREVIATIONS

CFD	Computational Fluid Dynamics
MAE	Mean Absolute Error
HLLC	Harten–Lax–van Leer Contact
RANS	Reynolds-Averaged Navier-Stokes
AOA	Angle of Attack
CAD	Computer Aided Design
CNN	Convolutional Neural Network
MLP	Multilayer Perceptron
DNN	Deep Neural Network
ML	Machine Learning
FNN	Forward Neural Network
PINN	Physics-Informed Neural Network
DF	Distance Field
MAPE	Mean Absolute Percentage Error
PPI	Pixels per Inch

CHAPTER 1

INTRODUCTION

Airfoils are essential in aviation applications. They are shaped to direct the forces acting on objects stemming from the airflow around them in desired directions. Airfoils are designed for aircraft wings, control surfaces, and engine intake applications [1] . Selecting the airfoil shape that meets the requirements is crucial to achieving the correct magnitude and direction of aerodynamic forces. Therefore, the development of airfoil forms is of extreme significance.

Aircraft are designed to fulfill a variety of duties nowadays and operate in a variety of flight conditions. A plane operating in a different environment may experience various aerodynamic states. As a result, the shape of airfoils must be created to suit the operating conditions of aircraft. Given how important airfoil forms are, it is essential to precisely anticipate the flow they cause and, ultimately, the forces they apply to the object.

Several techniques can be used during the design phase of airfoils, including experimental studies, CFD analyses, and the increasingly popular machine learning applications. Experimental studies are a tried-and-true method that provides information on the structural and aerodynamic properties of airfoil shape. However, experimental studies are often very costly and time-consuming. On the other hand, numerical analyses are used to examine the aerodynamic behaviors of airfoils. The mathematical modeling and computer solution of flow around an object is the foundation of CFD. The studies conducted with CFD methodology could be cross-checked with the data obtained from experimental studies to ensure the accuracy of results even though they are more affordable than experimental studies. With the development of computing technology and increased processing power, CFD

applications have gained popularity and are now frequently used as tools for both academic research and commercial applications.

Machine learning applications of fluid mechanics problems have recently gained popularity, as evidenced by the large number of academic studies published [2]. Many fluid mechanics-related machine learning applications, such as predicting flow fields and aerodynamic coefficients, may involve analyzing large datasets that have been previously acquired. These applications provide alternative solvers or analysis tools for CFD and experimental studies. Given the current performance of these applications and their advantages in terms of time and cost, they can be considered in the early stages of the design process.

Numerous highly accurate machine learning models available today can be used as early design tools. Convolutional Neural Networks (CNN), a deep learning technique, was used to create some of these tools. This approach, frequently used in image processing and recognition, is modified for fluid dynamics by predicting flow patterns by assigning flow around an airfoil to pixels with a specific resolution [3]. CNN can be used to train various parameters, including flow coefficients, primitive or conservative variables, and more.

The study of A. da Silva et al. [4] presented how neural networks give promising results about the design variables, such as pressure, for both low ($0.3 < M < 0.72$) and high ($0.72 < M < 0.82$) transonic flow regimes. The study reported that, when the obtained results are considered, larger networks could be used to achieve accurate predictions for high transonic flow regimes.

Another data-driven model by Jin X et al. [5], is developed to predict the velocity field around a circular cylinder at different Reynolds numbers utilizing the pressure field as an input. Sekar et al. [6] use a two-step approach to predict the flow field around airfoils. At first, they benefit from CNN to parameterize the airfoils from airfoil images. Then, the obtained parameters are then used in an Multi Layer Perceptron (MLP) network to predict the flow field and flow coefficients. The authors reported that an overall accuracy of 97% was achieved for the test cases. Hui

et al. [7] report the performance of CNN compared to FNN, which analyzes the given image at once, especially when structured data consisting of 2D cartesian coordinates of airfoils is given, and pressure distribution over an airfoil is predicted.

CNNs' capability of predicting aerodynamic coefficients around the airfoils is investigated by Yilmaz et al. [8], Yuan et al. [9], Portal-Porras et al. [10], and Wu et al. [11] using different types of input to the networks. The authors also observed the effect of network depth utilized in the studies and reported no strict correlation between network depth and prediction accuracy. Also, Zhang et al. [12] benefit from CNNs to predict the lift coefficient of an airfoil dataset ranging between -10 to 30 degrees of angles of attack and 0.3 to 0.8 Mach number. The study also compares the results between CNN and MLP-based predictions. The study shows the capacity of CNN to provide accurate predictions with minimal constraints. Olayemi et al. [13] investigate the aerodynamic coefficients of supercritical airfoils at a transonic flow regime and discuss the input image size and the effect of the input size on the predictions.

Low Reynolds number ($Re=10$) flows of 10,000 arbitrary shapes are trained using CNN by Viquerat et al. [14] to achieve accurate drag predictions. The authors reported in the study that drag prediction errors range between 0.2 to 3.06% . Flow field maps of 2D arbitrary shapes generated with Bezier curves at low Reynolds numbers are predicted in the study of Chen J. et al. [15]. The authors benefit from one of the presented U-net-based architectures, which is a parallel U-net. Sun et al. [16] benefit from target wall Mach number distribution as an input and aim to predict target airfoil shapes. Their study is an example of how inverse design cases are investigated by introducing them into neural networks.

Another example of inverse design is the study of Sekar et al. [17], which aims to predict airfoil shapes for a given pressure distribution. The study uses different types of neural networks (CNN and other Deep Neural Networks DNNs) with two different input methods (image form in CNN, numerical form in DNN).

Singh et al. [18] studied a methodology that brings a data-driven framework and Spalart-Allmaras turbulence model of a traditional RANS solver. Their method enhances the predictive capability for the various flow conditions around the airfoils. CNNs require more data and processing than machine learning techniques, but when a suitable network is established, they can offer significant insights into general flow characteristics. Combining the fluid dynamics equations with different conditions (boundary and initial) in modern machine learning applications enables more precise time-averaged and time-dependent results [19]. Raissi et al. [20] introduced Physics-Informed Neural Networks, which are trained to solve the learning tasks while obeying the physics given as partial differential equations. Also, Li et al. [21] report that using PINNs by including physical laws can help to achieve more accurate predictions by obeying the nature of the problem definition.

Kim et al. [22] assessed the potential of CNN for the geometric modification of airfoils to obtain a higher lift/drag ratio by modifying the pressure coefficient distributions around the airfoil surface.

1.1 Aim of This Study

In this work, a convolutional encoder-decoder network inspired by CNNFOIL [23], [24] by Duru et al. is improved for better fidelity to flow physics. The authors of CNNFOIL built a CNN model in the original work that takes DF as input and predicts pressure coefficients and the Mach number. Then, predictions are utilized to calculate both lift and drag coefficients and compare them with the ground truth values. The dataset used by Duru et al. has a wide range of angle of attack values (-10 to 20 degrees) at 0.7 Mach number where shock formation can still be observed. The model has been evaluated by another airfoil from outside of the selected dataset, and the overall results are in good agreement with the CFD outputs.

In this study, a CNN-based machine-learning tool to predict the flow around airfoils and primitive variables is developed. This tool utilizes conservation of momentum

and conservation mass principles to calculate loss functions that enhance the accuracy of the predictions. A carefully crafted airfoil flow database to train the model is developed using Reynolds-Averaged Navier-Stokes Equations-based CFD analyses with the Spalart-Allmaras Turbulence Model [25] of 204 airfoils at various angles of attack. The CFD solutions and conservation law-based loss functions applied in the ML model utilize the same flux scheme to ensure the predictions are as close as possible to the CFD solutions. The airfoil shape that the developed model needs is provided by the distance fields surrounding the airfoils. The wall distance is a shape descriptor that fits the fluid flow ML models since the wall boundaries shape the flow field.

This thesis aims to prepare a trustworthy CFD database, auxiliary tools to convert this database into formats appropriate for the machine learning environment, and a CNN to interpret relationships between pixels to accomplish these goals precisely. The developed tool will be a showcase to assess the utilization of conservation laws in ML training on fluid mechanics problems. The flow over airfoil problem is chosen as the model problem. To assess the performance of the developed tool, integral quantities related to conservative variables are calculated. The conservation of momentum is Newton's second law for a control volume. Therefore, an airfoil prediction model trained with the conservation of momentum loss function should provide better drag and lift estimates. This thesis presents the force and moment estimations of airfoil lift and drag forces. This tool uses far-field flow data rather than those on the wing surface to accurately predict the drag and lift by combining these predictions with the momentum conservation loss function.

1.2 Airfoil Physics and Performance Criteria

The Bernoulli principle states that pressure decreases as flow velocity increases. When properly directed, the force that produces lift is produced by the pressure difference between high and low-pressure areas. The airfoil shape controls the lift force [26], where the cambered upper surface accelerates the flow, and the relatively

flat lower surface decelerates the flow through surfaces. This shape suggests areas of low pressure on top and relatively higher pressure on the bottom surfaces. Lift and drag forces are components of the net forces in the normal direction and parallel to the wind direction. The lift and drag forces resulting from the pressure distribution of the airfoil are called pressure lift and drag.

The Reynolds number also has a significant effect on the flow around the airfoil. The viscous force is an additional force among the pressure, the wind component of which is called the viscous drag. Viscous forces are more pronounced in areas with high-velocity gradients, such as those close to an airfoil's surface. The Mach number represents the ratio of the flow velocity to the speed of sound in that medium. In a gas, the Mach number of a flow indicates the compressibility effects. The compressible effects increase as the Mach number increases and the flow structure changes. The angle of attack is formed by the wing's chord line and the flow direction, as shown in Figure 1.1. The changes to this angle result in changes to the wing's aerodynamic properties.

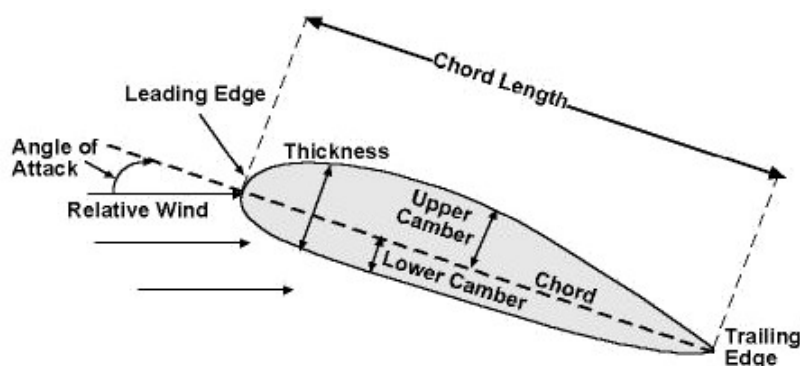


Figure 1.1. The angle of Attack Between an Airfoil and Relative Wind

CHAPTER 2

DATA PREPARATION

In this chapter, the dataset obtained from CFD outputs will be converted to a format such that the introduced CNN model can utilize the data and training sessions can be conducted. CFD outputs give an enormous amount of data in different intensities at different regions on the flow domain. The model created in this study requires some of the flow properties, such as pressure, velocity, and temperature. Therefore, the CFD outputs should be extracted from the raw outputs and converted into a format.

2.1 Problem Definition

In machine learning applications, it is possible to acquire predictions of aerodynamic flows, simulate turbulent flows by improving the accuracy of turbulence models, detect some anomalies in the fluid flows such as vortices, shocks, and separations, and reduce the number of physical experiments or CFD analysis of fluid flows in general.

While machine learning techniques are used to predict the flow fields, the forces acting on the surfaces may not be predicted or calculated accurately. However, the addition of sensors to assess fidelity of the predictions to the physics of the problem to the selected machine learning method can improve the accuracy desired on the forces acting on the domain. CFD tools are developed using conservation laws. The same conservation laws can be applied to some Neural network methods as such physics sensors. With this approach, the underlying neural network may be more aware of the physics of the problem. Hence, the prediction of the solution field will be improved. Following this idea, 2D compressible flow around airfoils is selected as the model problem. Therefore, momentum conservation laws are introduced to the utilized CNN as a loss function in this study.

Predicting the forces acting around and on the airfoil is the main aim of this study. As an example problem to conduct the study, a dataset of CFD solutions of 204 distinct airfoils at 0.7 Mach has been selected. 0.7 Mach flow speed can contain many fluid phenomena such as shocks, flow separation, and wake regions at different configurations, although the flow regime is still below the transonic regime. Also, the angles of attack values of the solutions range between -10 to 20 degrees, and the free-stream Reynolds number is $Re = 6 \times 10^6$. The high Reynolds number states that viscous forces will be less dominant in the flow and may even be neglected.

2.2 Conservation Laws

Conservation laws are the fundamental principles that refer to preserving certain quantities. The behavior of physical systems and dynamics of the natural world can be modelled and solved by these conservation laws.

Conservation of mass is the first principle of conservation laws, especially for fluid mechanics. It states that mass cannot be created or destroyed in an isolated system. The mathematical representation of the conservation of mass can be seen in Equation 2.1.

Conservation of momentum is another principle of conservation laws. Momentum conservation states that the total linear momentum of the same system remains constant only if no external forces are acting on the system. Linear momentum can be expressed as the product of mass and velocity, and the mathematical expression for the conservation of momentum can be seen in Equation 2.2.

Lastly, the conservation of energy states that the total energy of the same isolated system must remain over time only if there are no external forces applying work on the system. The mathematical expression of the conservation of energy can be given as in Equation 2.3.

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_i)}{\partial x_i} = 0 \quad 2.1$$

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_j)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} \quad 2.2$$

$$\frac{\partial(\rho e_t)}{\partial t} + \frac{\partial(\rho h u_j)}{\partial x_j} = \frac{\partial}{\partial x_j} (u_i \tau_{ij} - q_j) \quad 2.3$$

Density, pressure, velocity, temperature, total energy per unit mass, and total enthalpy are the terms that make up these equations. When combined with the equation of state, this system of equations can be closed, and this set, which has a total of six equations, can be solved for six unknowns. The equation of state for an ideal gas is given in Equation 2.4.

$$p = \rho RT \quad 2.4$$

The shear stress tensor in Equation 2.2 and Equation 2.3 can be expressed by Equation 2.5:

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \frac{\partial u_k}{\partial x_k} \delta_{ij} \right) \quad 2.5$$

Internal energy and total energy can be calculated by using Equations 2.6 and 2.7, where gas constant $R = 287.1 \text{ J / kgK}$, and specific heat ratio $\gamma = 1.4$ for air. Also, total enthalpy is calculated as in Equation 2.8:

$$e = \frac{RT}{\gamma - 1} \quad 2.6$$

$$e_t = \frac{1}{2} u_i^2 + e \quad 2.7$$

$$h = e_t + \frac{p}{\rho} \quad 2.8$$

Dynamic viscosity is determined by Sutherland's law, which is calculated in Equation 2.9:

$$\mu = \frac{1.458 \times 10^{-6} T^{3/2}}{T + 110.4} \quad 2.9$$

Lastly, the heat flux can be given as in Equation 2.10, where T is in Kelvin and k is the thermal conductivity.

$$q_j = -k \frac{\partial T}{\partial x_j} \quad 2.10$$

2.3 CFD Methodology

The equations represented in Section 2.2 can be coupled and solved in a CFD solver. The CFD solver, which the dataset used in this study is taken from, is a finite volume-based solver that can solve compressible conservation equations with the Spalart-Allmaras turbulence model. A RANS-based methodology is deemed sufficient for steady-state simulations. Also, note that using a one-equation RANS-based turbulent model is cheaper regarding time and computational power required.

CFD outputs are taken from the study of Duru et al. [24]. The authors report that the mesh used in the study has an O-topology grid, and the far field is fixed at 500 chord lengths away from the airfoil surface. One example of the computational domain and the mesh around an airfoil can be seen in Figure 2.1.

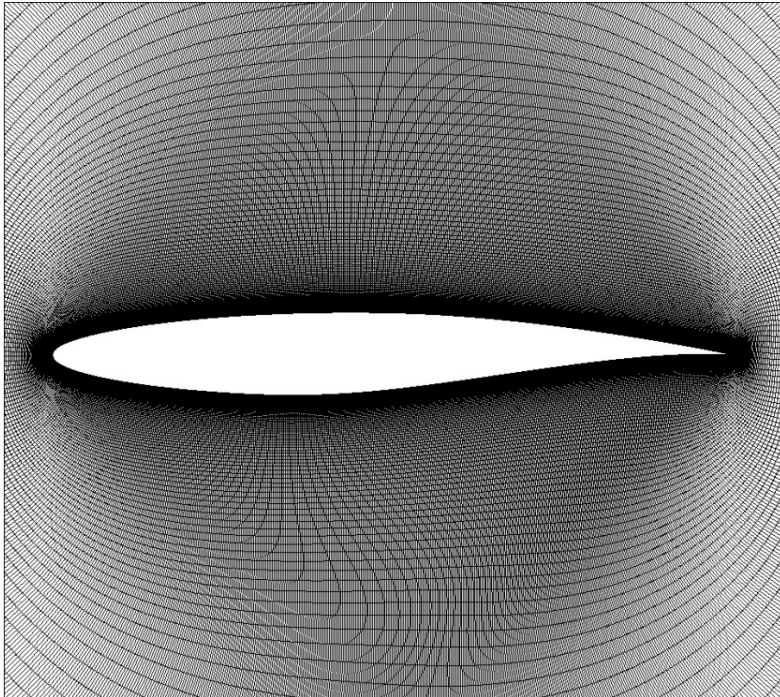


Figure 2.1. Computational Domain and Mesh Structure [24]

After this system of equations has been solved, the outputs from the CFD results must be converted into a particular format for use as training parameters in the training set. Pressure, velocity (horizontal and vertical components), and temperature are the physical parameters that need to be transformed into the appropriate format. Using an open-source data visualization program, Paraview, these four parameters are transformed into the desired format. The 'Resample to Image' filter is used to create a rectangular area with dimensions of two chords by one chord that is centered on all of the mentioned airfoils. Then, all data in this area is resized to the preferred format, particularly 256x256 pixels.

Using CNNs as a method requires $n \times m$ pixels data format and grid resolution. That is why the data extracted from the CFD solutions must be approximated using pixels with a lower resolution than the CFD mesh resolution for this method to work. The impact of these newly approximated parameters on the actual values in the study is elaborated in detail in the following sections.

The 256x256 matrices of the approximated pressure, velocity, and temperature parameters are ready to be used as the ground truths and utilized for the algorithm's loss functions. One should note that the pixels produced here are stretched horizontally by a factor of 2:1. The reason for stretching the domain in the horizontal axis is that the forces acting on the field are more dominant in the horizontal direction than the vertical direction. For example, pressure field data imported into the pixels and the stretched rectangular domain in the horizontal direction can be seen in Figure 2.2.

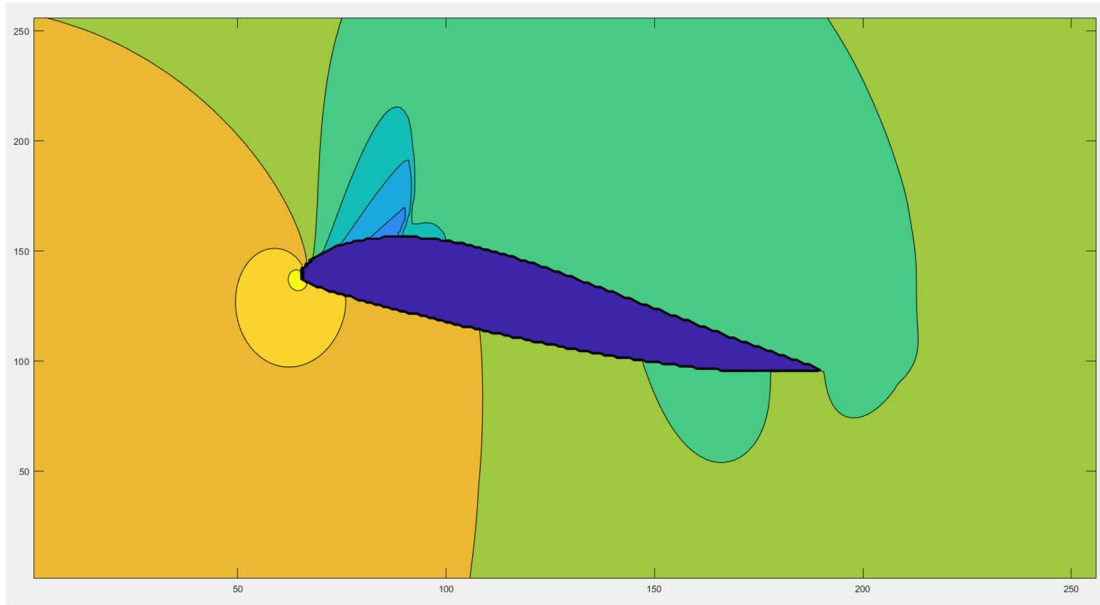


Figure 2.2. Pressure Distribution of E342 Airfoil, Rectangular Stretched Domain

Guo et al. [27] and Bhatnagar et al. [19] report that using a Distance Field matrix as an input improves the predictions. The required Distance Field matrix in this study, which will serve as input, is obtained using a MATLAB script that calculates the distance of each pixel in the selected region to the nearest point on the airfoil surface. Similar to the preparation of flow parameters, this process is computed for each airfoil at every angle.

Consequently, along with the Distance Field (DF) matrix that will be provided as input to the algorithm, the flow parameters used in the loss functions are also prepared in a 256x256 format for training.

In some of the study's parameters, the regions corresponding to each airfoil's inner portion have been swapped out for values of far-field pressure and temperature to improve training performance and prevent the occurrence of NaN values in the denominators of the employed loss functions. These numbers are 300 Kelvin for temperature and 39244.5 Pascal for pressure.

CHAPTER 3

HLLC APPROXIMATE RIEMANN SOLVER & MOMENTUM FLUX-BASED LOSS FUNCTION

This chapter introduces the HLLC Approximate Riemann Solver [28] that is used to calculate the conservative fluxes. Then, a HLLC momentum flux-based physical loss function is presented. Also, the application of HLLC as a loss function is explained.

The CFD solver, which the dataset was created from, uses the HLLC approximate Riemann solver. If the same methodology can be applied to the ML algorithm somehow, then the predictions taken from the models could be more accurate in terms of both primitive variables and the net forces acting on the domain. Also, high-gradient regions where the discontinuities may occur can be resolved more precisely.

3.1 The HLL and HLLC Riemann Solvers

3.1.1 The Riemann Problem

Riemann problem is the generalized shock tube problem that is often utilized in compressible flow solvers. The Riemann problem includes the interaction of two different states of a fluid, which is separated by a discontinuity. These states have different flow variables, such as pressure, density, temperature, velocity, etc. The structure of the solution on an $x - t$ plane can be seen in Figure 3.1.

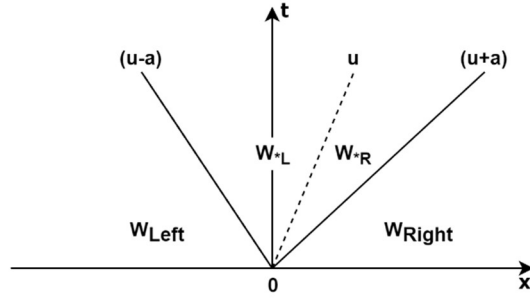


Figure 3.1. Structure of the Solution of the Riemann Problem for Time-Dependent Euler Equations

The solution has three waves, and these waves are associated with the eigenvalues of $\lambda_1 = u - a$, $\lambda_2 = u$ and $\lambda_3 = u + a$. Four different states occur, which are separated by these three waves. States are named as W_L , W_{*L} , W_{*R} , and W_R . The middle wave (λ_2) separates W_{*L} and W_{*R} , and this wave is called contact discontinuity. On the other hand, λ_1, λ_3 waves can be either shock waves or rarefaction waves. Therefore, four possible scenarios are possible.

3.1.2 The Godunov Flux

The Riemann Problem with its initial and boundary conditions was given as:

$$\begin{aligned} \text{PDEs: } & U_t + F(U)_x = 0 \\ \text{IC: } & U(x, 0) = U^{(0)}(x) \\ \text{BCs: } & U(0, t) = U_L(t), \quad U(L, t) = U_R(t) \end{aligned} \quad 3.1$$

Another representation of Equation 3.1 is by using an explicit conservative formula:

$$U_i^{n+1} = U_i^n + \frac{\Delta t}{\Delta x} \left[F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}} \right] \quad 3.2$$

Godunov's flux method states that $F_{i+\frac{1}{2}} = F(U_{i+\frac{1}{2}}(0))$, where $U_{i+\frac{1}{2}}(0)$ is taken as equal to $U_{i+\frac{1}{2}}(x/t)$, which is an exact similarity solution to the Riemann Problem.

For the 3D case, U and F vectors are given as:

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix}, F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(E + p) \end{bmatrix} \quad 3.3$$

The primitive variables of the states belonging to x_L and x_R are:

$$w_L = \begin{bmatrix} \rho_L \\ u_L \\ v_L \\ w_L \\ p_L \end{bmatrix}, w_R = \begin{bmatrix} \rho_R \\ u_R \\ v_R \\ w_R \\ p_R \end{bmatrix} \quad 3.4$$

The state at the interface should be determined for calculating the fluxes, and state variables should be put into conservation equations.

3.1.3 Integral Relations & Consistency Condition

We can apply some changes to our control region on $x - t$ plane. First, the control region is set as $V = [x_L, x_R] \times [0, T]$. The new form of the selected control region and regarding waves can be seen in Figure 3.2.

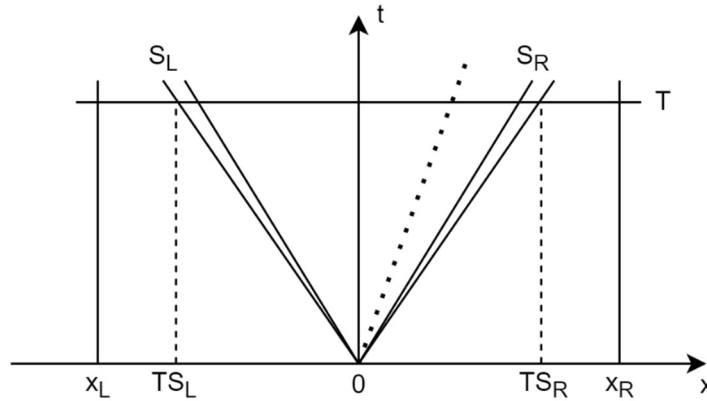


Figure 3.2. New Form of the Control Region on $x-t$ Plane.

The integral form of conservation law can be expressed in Equation 3.5 in this control region.

$$\int_{x_L}^{x_R} U(x, T) dx = \int_{x_L}^{x_R} U(x, 0) dx + \int_0^T F(U(x_L, t)) dt - \int_0^T F(U(x_R, t)) dt \quad 3.5$$

One can simplify the right-hand side of the equation, and Equation 3.5 becomes,

$$\int_{x_L}^{x_R} U(x, T) dx = x_R U_R - x_L U_L + T(F_L - F_R) \quad 3.6$$

The obtained equation is named the consistency condition. Furthermore, we can now divide the left-hand side of Equation 3.5 into three different integrals with the help of our new control volume given in Figure 3.2.

$$\int_{x_L}^{x_R} U(x, T) dx = \int_{x_L}^{TS_L} U(x, T) dx + \int_{TS_L}^{TS_R} U(x, T) dx + \int_{TS_R}^{x_R} U(x, T) dx \quad 3.7$$

In this equation, S_L and S_R are the waves from the solution of the Riemann Problem. In Equation 3.7, the first and the third terms on the right-hand side represent two rectangular regions. They can be evaluated easily, and the equation becomes,

$$\int_{x_L}^{x_R} U(x, T) dx = \int_{TS_L}^{TS_R} U(x, T) dx + (TS_L - x_L)U_L + (x_R - TS_R)U_R \quad 3.8$$

The right-hand side of this equation will be imported to the left-hand side of Equation 3.6.

$$\int_{TS_L}^{TS_R} U(x, T) dx + (TS_L - x_L)U_L + (x_R - TS_R)U_R = x_R U_R - x_L U_L + T(F_L - F_R) \quad 3.9$$

After arranging the resultant equation:

$$\int_{TS_L}^{TS_R} U(x, T) dx = T(S_R U_R - S_L U_L + F_L - F_R) \quad 3.10$$

At this stage of the procedure, we can divide the equation by $T(S_R - S_L)$. This is the width of the wave system on the horizontal axis given in Figure 3.2. The final form of the equation can be expressed as:

$$\frac{1}{T(S_R - S_L)} \int_{TS_L}^{TS_R} U(x, T) dx = \frac{S_R U_R - S_L U_L + F_L - F_R}{S_R - S_L} \quad 3.11$$

If S_L and S_R are known (or can be calculated) initially, then the right-hand side of Equation 3.11 can be considered as a known parameter. Therefore, the right-hand side of the equation can be denoted by U_{hl} .

$$U_{hl} = \frac{S_R U_R - S_L U_L + F_L - F_R}{S_R - S_L} \quad 3.12$$

As a last step, let us divide our control volume into two parts, which are the left-hand side of the t -axis and the right-hand side of the t -axis. First, we consider the left-hand side and apply the necessary changes in Equation 3.10.

$$\int_{TS_L}^0 U(x, T) dx = -TS_L U_L + T(F_L - F_{0L}) \quad 3.13$$

Then, we need to solve for F_{0L} , and what we achieve is:

$$F_{0L} = F_L - S_L U_L - \frac{1}{T} \int_{TS_L}^0 U(x, T) dx \quad 3.14$$

The same procedure can be conducted for the right-hand side of the t -axis to find F_{0R} as:

$$F_{0R} = F_R - S_R U_R - \frac{1}{T} \int_0^{TS_R} U(x, T) dx \quad 3.15$$

Combining Equation 3.11, Equation 3.14, and Equation 3.15 yields:

$$F_{0R} = F_{0L} \quad 3.16$$

which is where the consistency condition comes from in Equation 3.6.

3.1.4 The HLLC Approximate Riemann Solver

HLLC (Harten, Lax, van Leer Contact) approximate Riemann solver is a numerical method used in CFD to solve the Riemann problem for compressible flow. HLLC methodology was developed to address the limitation of previous solvers struggling to solve contact discontinuities. The HLLC methodology solves this limitation considering an approximate contact discontinuity wave between two sides (states) of the fluid motion. This scheme is a modified and improved version of the HLL scheme. The main difference is the consideration of the contact wave in the HLLC scheme. Therefore, it can be utilized more accurately in a wider range of systems. The structure of the HLLC scheme can be seen in Figure 3.3.

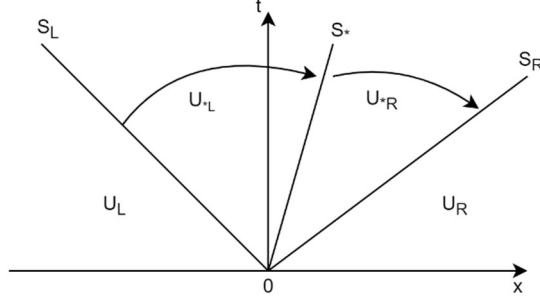


Figure 3.3. Structure of the HLLC Approximate Riemann Solver

In HLLC approximate Riemann Solver, S_L and S_R terms will represent the slowest and the fastest wave speeds. In addition, S_* will take place as a middle wave speed. To begin the procedure, we will benefit from Equation 3.11 and divide this equation's left-hand side into two parts, as follows,

$$\frac{1}{T(S_R - S_L)} \int_{TS_L}^{TS_R} U(x, T) dx = \frac{1}{T(S_R - S_L)} \int_{TS_L}^{TS_*} U(x, T) dx + \frac{1}{T(S_R - S_L)} \int_{TS_*}^{TS_R} U(x, T) dx \quad 3.17$$

In this equation, we can express the right-hand side in two different definitions as:

$$U_{*L} = \frac{1}{T(S_* - S_L)} \int_{TS_L}^{TS_*} U(x, T) dx \quad 3.18$$

$$U_{*R} = \frac{1}{T(S_R - S_*)} \int_{TS_*}^{TS_R} U(x, T) dx \quad 3.19$$

Equation 3.18 and Equation 3.19 are substituted into Equation 3.17 and put the resultant equation into the left-hand side of Equation 3.11, and the new equation becomes:

$$U^{hll} = \left(\frac{S_* - S_L}{S_R - S_L} \right) U_{*L} + \left(\frac{S_R - S_*}{S_R - S_L} \right) U_{*R} \quad 3.20$$

where U^{hll} is given in Equation 3.12. Finally, we can express \tilde{U} as:

$$\tilde{U} = \begin{cases} U_L & \text{if } \frac{x}{t} \leq S_L \\ U_{*L} & \text{if } S_L \leq \frac{x}{t} \leq S_* \\ U_{*R} & \text{if } S_* \leq \frac{x}{t} \leq S_R \\ U_R & \text{if } \frac{x}{t} \geq S_R \end{cases} \quad 3.21$$

Previously, we have calculated F_{0L} and F_{0R} in Equation 3.14 and Equation 3.15. Similarly, for the HLLC Approximate Riemann Solver, we can find expressions for F_{*L} and F_{*R} . This time, we may also represent these two in terms of each other, which Equation 3.23 stands for.

$$F_{*L} = F_L + S_L(U_{*L} - U_L) \quad 3.22$$

$$F_{*R} = F_{*L} + S_*(U_{*R} - U_{*L}) \quad 3.23$$

$$F_{*R} = F_R + S_R(U_{*R} - U_R) \quad 3.24$$

It can be observed from these three equations that there are four unknowns for three equations. One can argue that if we write F_{*L} in terms of F_{*R} , there will be four equations. However, that equation and Equation 3.23 will not be independent equations. Therefore, for the given unknowns, F_{*R} , F_{*L} , U_{*R} and U_{*L} , vectors U_{*R} , and U_{*L} are required to be calculated. Then, fluxes can be calculated from previously given equations. In this case, Equation 3.23 will not be needed to close the equation set.

In HLLC Approximate Riemann Solver, to find the vectors U_{*R} , and U_{*L} , some conditions should take place. Imposed conditions can be given as:

$$\left\{ \begin{array}{l} u_{*L} = u_{*R} = u_* \\ \rho_{*L} = \rho_{*R} = \rho_* \\ v_{*L} = v_L, v_{*R} = v_R \\ w_{*L} = w_L, w_{*R} = w_R \end{array} \right. \quad 3.25$$

We just take the main components of velocity and the pressure term and equate them to each other for the left and the right sides of the star region. In addition to these adjustments, Toro also suggests setting $S_* = u_*$. Therefore, Equation 3.22 and Equation 3.24 are rearranged. New form of the equations is:

$$S_L U_{*L} - F_{*L} = S_L U_L - F_L = Q_L \quad 3.26$$

$$S_R U_{*R} - F_{*R} = S_R U_R - F_R = Q_R \quad 3.27$$

At this point, a quick reminder of U and F vectors will be very beneficial to understand the solution itself.

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix}, F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(E + p) \end{bmatrix} \quad 3.28$$

Solving Equation 3.26 and Equation 3.27 with previously given conditions yields:

$$U_{*N} = \rho_N \left(\frac{S_N - u_N}{S_N - S_*} \right) \begin{bmatrix} 1 \\ S_* \\ v_N \\ w_N \\ \frac{E_N}{\rho_N} + (S_* - u_N) \left[S_x + \frac{p_N}{\rho_N(S_N - u_N)} \right] \end{bmatrix} \quad 3.29$$

Here, N stands for both left (L) and right (R). For instance, N=L will give F_{*L} when obtained U_{*L} is used in Equation 3.22. A similar approach can be applied to Equation 3.24. Therefore, all unknowns can be obtained by applying the given steps.

To summarize the flux vector not only in the starred region but also for the left and right sides of the control volume, it can be written as:

$$F_{i+\frac{1}{2}}^{hllc} = \begin{cases} F_L & \text{if } 0 \leq S_L \\ F_{*L} = F_L + S_L(U_{*L} - U_L) & \text{if } S_L \leq 0 \leq S_* \\ F_{*R} = F_{*L} + S_*(U_{*R} - U_{*L}) & \text{if } S_* \leq 0 \leq S_R \\ F_R & \text{if } 0 \geq S_R \end{cases} \quad 3.30$$

3.1.5 Wave Speed Calculations

Up to this point, we assumed that the terms like S_L , S_R , S_* are known parameters. However, they should be obtained with appropriate methods. Note that the S_* wave does not exist in HLL due to the two-wave model. On the other hand, HLLC, which stands for Harten, Lax, and van Leer with Contact, considers a three-wave model and S_* , which stands for middle wave, should be taken into consideration as well.

In the direct wave speed method, S.F. Davis [29] suggested that $S_L = u_L - a_L$, and $S_R = u_R - a_R$, where a term is the speed of sound. A better selection of S_L , and S_R will be given as follows:

$$S_L = \min\{u_L - a_L, u_R - a_R\}, S_R = \max\{u_L + a_L, u_R + a_R\} \quad 3.31$$

In addition, S.F. Davis, and B. Einfeldt [30] suggest using Roe average eigenvalues for S_L , and S_R , which can be expressed as:

$$S_L = \tilde{u} - \tilde{a}, S_R = \tilde{u} + \tilde{a} \quad 3.32$$

where \tilde{u} and \tilde{a} are the Roe-average particle and sound speeds, and they can be calculated as:

$$\tilde{u} = \frac{\sqrt{\rho_L}u_L + \sqrt{\rho_R}u_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \tilde{a} = \left[(\gamma - 1) \left(\tilde{H} - \frac{1}{2}\tilde{u}^2 \right) \right]^{\frac{1}{2}} \quad 3.33$$

Equation 3.33 contains many new terms, so a quick reminder of energy-enthalpy-internal energy and equation of state concepts would be very beneficial. Enthalpy can be written as in Equation 2.8. Another term is the internal energy term. Internal energy can be calculated as in Equation 2.6 for ideal gases. Including the equation of state equation in Equation 2.4, when all these equations are combined and solved for the enthalpy term H , the obtained equation is as follows:

$$H = \frac{RT\gamma}{(\gamma-1)} + \frac{1}{2}V^2 \quad 3.34$$

Before using H , let us see how \tilde{H} is defined by using Roe-averaging.

$$\tilde{H} = \frac{\sqrt{\rho_L}H_L + \sqrt{\rho_R}H_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \quad 3.35$$

Equation 3.34 can be utilized to find H_R , and H_L so that they can be used in Equation 3.35 to obtain \tilde{H} .

3.2 Calculation of Momentum Fluxes with HLLC Approximate Riemann Solver

The HLLC momentum flux-based physics-based loss function will be presented in this part. Note that, previously explained Roe-Averaged properties are used in this calculation, such as velocity, enthalpy, speed of sound, etc.

The CFD solver, FlowPsi [31], benefits from a preconditioning HLLC flux option. Chorin-Turkel [32] preconditioning is implemented to obtain accurate and converged solutions, especially for low-speed flows. This methodology is especially beneficial for RANS flow computations. Preconditioning rescales the wave speeds, resulting in improved iterative convergence to the solution. Also, preconditioning reduces the dissipation of HLLC fluxes as well.

There are some variables used in the HLLC flux calculations that come from the preconditioning method, such as η_P , M_r , and σ . They are the preconditioning parameter, given in Equation 3.36, the local Mach number, and a term calculated by using some flow variables given in Equation 3.37.

$$\eta_P = M_r^2 / (1 - M_r^2) \quad 3.36$$

$$\sigma = \sqrt{u_n^2 (1 - \eta_P)^2 + 4\eta_P a^2} \quad 3.37$$

In general, the code that calculates the mass and the momentum fluxes based on the HLLC Approximate Riemann Solver has the key components and the following steps:

1. The code takes the input variables from right and left states based on the pressure, velocity components, and temperature.
2. Characteristic variables and speed are calculated as parts of the HLLC solver for left and right states.
3. Wave speeds (S_L, S_R, S_M) are calculated based on the preconditioning terms and the velocity components of the states.
4. The mass and the momentum fluxes are calculated through the interface, considering different possible wave configurations such as contact wave, rarefaction wave, or shock wave.
5. Then, the variables in the intermediate state between the right and the left states are divided by contact wave and these two regions are also named as starred region.

6. Lastly, corrections are made in the flux calculation, which considers the influence of the star region.
7. Mass and momentum fluxes on each side of a pixel are found using neighboring pixels around the targeted pixel.

After finding the fluxes, an HLLC momentum flux-based physical loss function should be evaluated. Predicting the fluxes at each interface is not different from directly predicting the flow variables using the numerical Mean Absolute Error (MAE) loss function. The reason behind this is that fluxes are already calculated based on these variables. Therefore, instead of predicting all the momentum fluxes of a pixel individually, their summation is predicted in this study. Figure 3.4 shows the momentum fluxes calculated at each pixel.

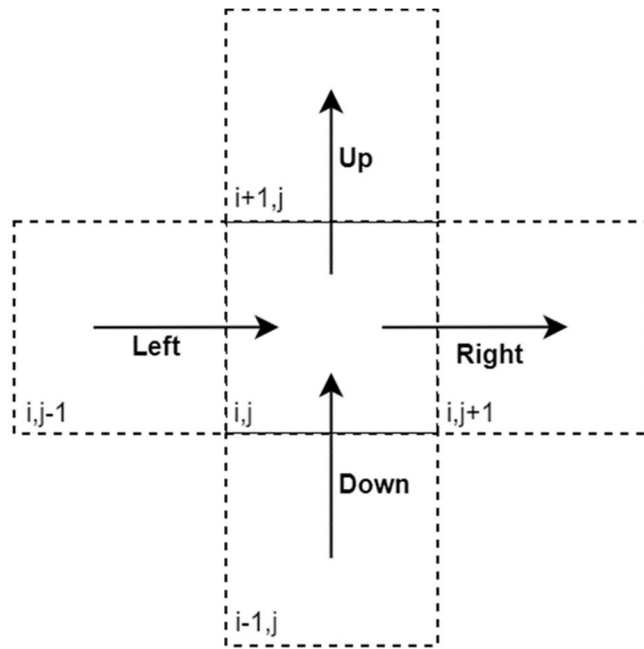


Figure 3.4. Momentum Fluxes Representation at Each Pixel

The summation of momentum fluxes at each pixel is also trained by using MAE formulation, given in Equation 4.4. Note that this MAE loss function involves the HLLC-based momentum flux instead of primitive variables used in the numerical training session. The details will be given in Chapter 4.

The momentum summation at each pixel has two components, one of which belongs to the net force acting in the horizontal direction, and the other one belongs to the net force acting in the vertical direction. In the model training sessions, vertical and horizontal components of these forces are trained together and equally weighted. These terms will appear in the HLLC momentum flux-based loss function, which is given in Equation 4.3.

CHAPTER 4

CNN ARCHITECTURE & NETWORK TRAINING

A convolutional neural network (CNN) is an artificial neural network used in deep learning that specializes in analyzing gridded imagery such as images, videos, and audio signals. CNNs are also referred to as successors of fully connected neural networks (FCNN) because CNNs incorporate shift invariance through the use of shared-weight convolution kernels or filters. These filters slide over the input features, generating translation-equivariant feature maps [33].

4.1 CNN Methodology

CNNs consist of convolutional layers, especially in the body. These layers can be considered an affine transformation on its input and output (Euclidean) spaces. This operation is accomplished by a kernel of filters striding over the input grid. Stride is a cardinal value by which the filter is jumped in any direction over the input grid. Lastly, padding is used to make input and output shapes coherent. The relation between the input, the filter, and the output spaces is given in Figure 4.1.

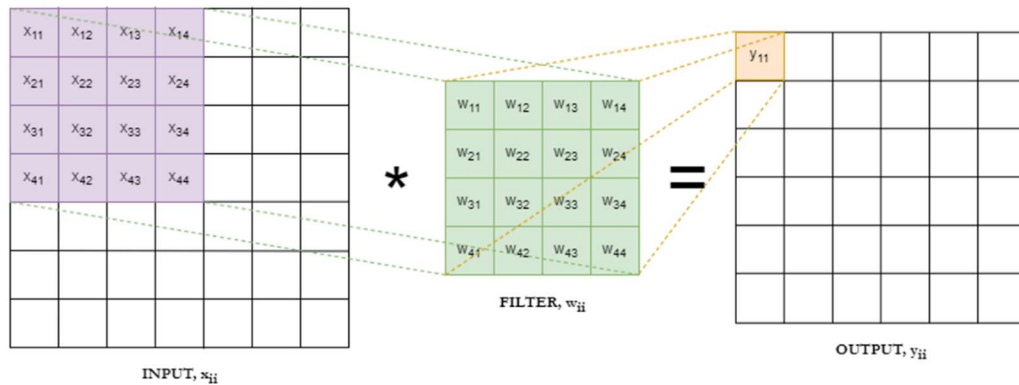


Figure 4.1. CNN Input, Filter, and Output Representation

For example, let an input image have a shape $(3, W_i, H_i)$, where the leading dimension is usually denoted as the “input channels” of that image grid. Therefore, Equation 4.1 is applied to calculate the output image shape.

$$W_o = (W_i - Filter + 2 \times Padding) / Stride + 1 \quad 4.1$$

Let our filters have the shape of $(3 \times 2 \times 2)$. The leading dimensions must be the same as that of the input grid, which stands for “input channels”. Also note that “output channels” are the number of filters in the kernel. Therefore, for the i^{th} output channel, we have Equation 4.2 as:

$$Y^{(i)} = f(X * F^{(i)} + b) \quad 4.2$$

where X is the input image, $F^{(i)}$ is a filter grid, b is a bias term, and f is a non-linearity (activation) function like Rectified Linear Unit (ReLU) or Hyperbolic Tangent (TanH).

Usually, the outputs of these filters are then passed through pooling layers, which reduce the spatial dimensionality of the data. Finally, the output is fed into one or more fully connected layers for classification or regression. However, since these are not used in our network, they will be skipped for simplicity.

4.2 Modified-CNNFOIL Network Details

In this work, we present a convolutional encoder-decoder network inspired by CNNFOIL [24] developed by Duru et al. The authors of CNNFOIL conduct the training sessions to predict pressure coefficient and Mach number fields. However, our network is a modified version of CNNFOIL. Our model predicts four flow variables: pressure, velocity, and temperature around an airfoil. The network used in this study is pre-trained with Mean Absolute Error for the primitive variables individually. In total, four different channels are trained numerically. Then, the model is improved by adding a momentum conservation-based loss function. The momentum fluxes in all domains except for inner and outer boundaries share the same flux function, HLLC, with the CFD solver. Outer boundaries, which are the

edges and corners of the rectangular domain, and the inner boundary, which is the airfoil surface, are not involved in HLLC calculation.

Modified-CNNFOIL used in this study is an encoder-decoder CNN. There are three encoder parts (with kernel sizes of 7, 6, and 4, respectively) consisting of 10 blocks, each with a convolutional layer and a batch normalization layer. Also, there are 1, 1, and two strides on the encoder section, respectively, and only one padding on the last encoder. The network takes 256×256 distance field images representing the airfoil geometry and outputs to a $256 \times 2 \times 2$ latent (feature) space. The decoder consists of 6 blocks in total but without batch normalization layers this time. The decoder part maps this space to four different channels: temperature, x-velocity, y-velocity, and pressure. Transpose convolutional blocks have (4×4) kernels, two strides, and one padding.

4.3 Network Training

A dataset whose details are given in the previous chapters, with a Mach number of 0.7 and a free stream Reynolds number of 6,000,000, is trained by the employed neural network architecture to predict the Reynolds-averaged flow field parameters for angles ranging from -10 to 20 degrees. The details is given in Table 4.1.

Table 4.1 Training Parameters and the Details

Training Parameters	Details
Activation Function	Exponential Linear Unit (ELU)
Optimizer	Adam Optimizer
Normalization Method	Mean-Std Normalization
Numerical Loss Function	Mean Absolute Error (Primitive Variables-Based)
Physics-based Loss Function	HLLC Momentum-based Flux Conservation

4.3.1 Numerical and Physics-based Loss Functions

Loss functions, which quantify the difference between the predictions and the ground truth, are one of the key components in training sessions. The aim of the training session is to reduce the amount of these losses. After calculating the losses at each epoch, computation of the gradient of the loss has been calculated with respect to model parameters. The chain rule of calculus is utilized to calculate all necessary gradients at each layer. Lastly, the optimizer adjusts the model parameters based on these gradients to reduce the losses in the predictions. The loss functions can be in different forms, such as MSE, MAE, and cross-entropy. However, the loss function can also be related to the model in terms of the physics behind the trained case.

The model is first pre-trained with MAE loss function three times with different learning rates. This numerical loss function focuses on the primitive variable itself. Then, the obtained model is subjected to the HLLC momentum flux-based loss function with different learning rates. The HLLC momentum flux-based loss function takes the predicted primitive variables and calculates momentum fluxes. Then, calculated momentum fluxes are put into the same Mean Absolute Error methodology. The total loss is calculated with the combination of primitive variable-based MAE errors and HLLC momentum flux-based MAE errors, as shown in Equation 4.3. The dominance of the physics-based loss error is emphasized as a result of lessening the function weight, and the training is continued. The weights are balanced during this process to prevent the model from forgetting the data it learned during initial training. These values are set as $\alpha=0.3$ and $\mu=0.7$. Note that momentum in the horizontal and vertical directions have the same functional weight.

$$Total\ Loss = [\alpha \times MAE_{Primitive\ Variables}] + [\mu \times MAE_{HLLC\text{-}Based\ Momentum\ Flux}] \quad 4.3$$

The Mean Absolute Error loss function can be expressed as shown in Equation 4.4. Here, the term y_j represents the ground truth, the term \hat{y}_j represents the model outputs, and the term n represents the number of data points in the flow field.

$$MAE = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j| \quad 4.4$$

4.3.2 Dataset Splitting and Normalization Method

The set of 204 airfoils has been randomly split into training, validation and test sets with an 85-8-7 split percentage for all angles of attack. After the division, airfoils are kept the same as all angles are trained separately. In other words, if an airfoil is in the test dataset, it will remain there for all angles of attack values. This approach ensures that the choice of airfoil profiles does not introduce bias in the training conducted from various angles. The mean-standard deviation normalization method has been used to normalize the primitive variables around the airfoils. However, when the physics-based loss function is used at each epoch, these data are denormalized once more. This denormalization process preserves the parameters' significance and physical significance during training.

4.3.3 Training Procedure, Number of Epochs and Accuracy Metric

A sequential view of those experiments applied to the model is as in Table 4.2:

Table 4.2 Training Process with Both Numerical and Physics-based Loss Functions

Training Method	Learning Rate	Number of Epoch
MAE	3E-3	150
MAE	1E-3	150
MAE	3E-4	150
HLLC + MAE	3E-4	150
HLLC + MAE	1E-4	100

Until Mean Absolute Error reached a steady state, the training process was continued. Using the parameters listed in Table 4.2, all channels, including pressure, velocity components, and temperature, were trained. 450 epochs in total were needed

to reach a steady-state condition. The training and validation loss graphs produced by the dataset's training sessions with a 0° angle of attack are shown in Figure 4.2. Note that Figure 4.2 shows only the losses of the numerical training sessions. Physical training session losses are shown in Section 5.2, which focuses on the momentum flux findings and the convergence of momentum conservation.

The physics-based loss function was added to the model after this numerical, non-physics-based training phase, affecting the total loss function by following the function weights given in Equation 4.3. This factor helped to refine the previously trained primitive variables, which improved the flow's momentum field and made it possible to predict the drag and lift coefficients with greater accuracy.

The Mean Absolute Percentage Error (MAPE) metric was used to evaluate the Flow field parameters after all training sessions were finished, and the model outputs were compared to the ground truth. This metric was used to compare models that had been trained using both MAE and HLLC momentum flux-based losses, and it looked at how the addition of physics-based loss function affected either improvements or decreases. Since the primary objective is to enhance momentum conservation and drag-lift coefficient predictions, a small increase in growing errors in primitive variables can be regarded in this study. The Mean Absolute Percentage Error (MAPE) metric can be defined in Equation 4.5:

$$MAPE = \frac{1}{N} \sum_{k=1}^n \left| \frac{GT_k - Pr_k}{GT_k} \right| \times 100 \quad 4.5$$

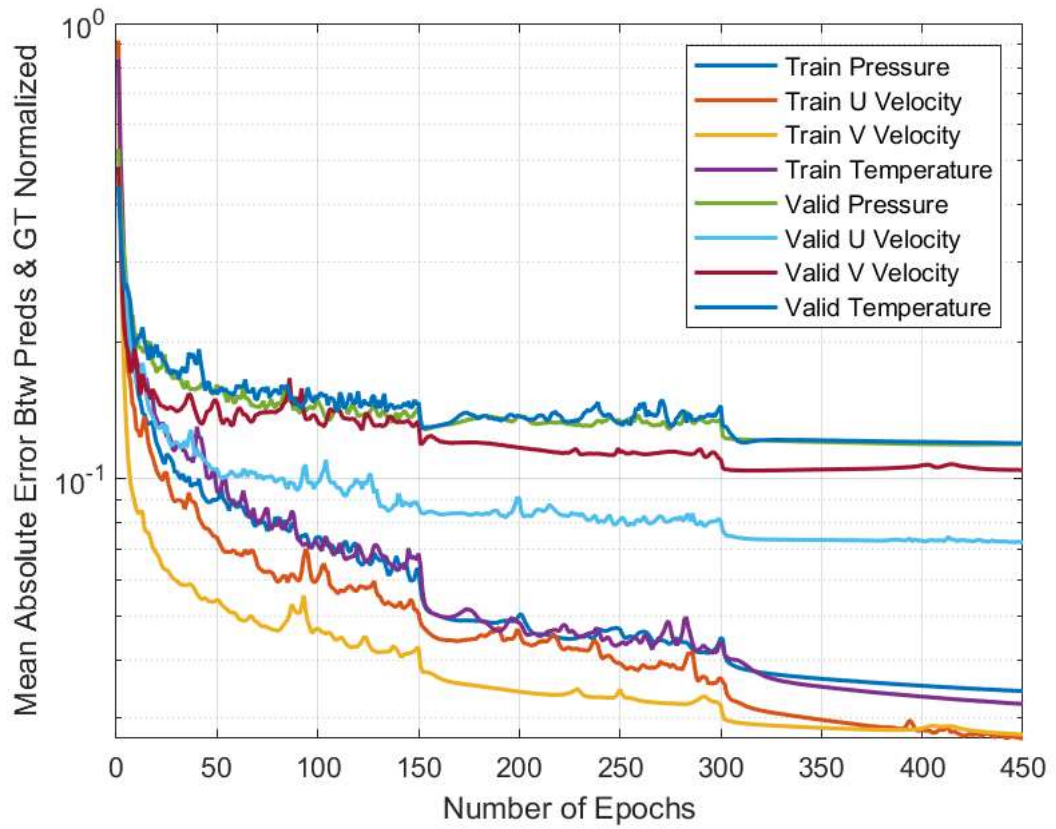


Figure 4.2. Mean Absolute Errors for Primitive Variables for both Training and Validation Set

CHAPTER 5

FLOW FIELD VARIABLES, MOMENTUM AND COEFFICIENT PREDICTIONS

This section deals with the variables that will be considered when assessing the effectiveness of the trained models. The main goal of the study is to make our ML methodology understand and predict the behavior of the overall flow and the net forces acting on the domain by enhancing them with an HLLC momentum flux-based loss function. Establishing a link between adjacent pixels in terms of momentum fluxes, using CNN's momentum conservation loss function, is required to accomplish more accurate predictions, and understanding of the flow field. Neighboring neurons can be adjusted collectively during training thanks to this connection. The experiments showed improvements in both the pixel-by-pixel calculation of momentum conservation and the fine-tuning of primitive variables. Additionally, a clear improvement in the control volume method's drag-lift coefficients, which use momentum fluxes unique to particular pixels, was seen compared to the results of a numerical training study. Therefore, the obtained results will be examined in three subcategories. These categories can be defined as the comparison of primitive variables with ground truths through numerical and physics-based training, the comparison of the momentum conservation field with ground truths through numerical and HLLC momentum flux-based physics-based training, and the comparison of drag-lift coefficients with ground truths and CFD solutions through numerical and physics-based training.

5.1 Effect of Model Training on Flow Field Predictions

As was already mentioned, a subset of the dataset's test airfoils, distributed in an 85-8-7 ratio, have been chosen for comparisons in this section to assess the degree of the model's learning. Randomly selected airfoils are RAE5215, NACA2421, and

NACA747A415, respectively, and these airfoils have not been seen by the model previously. This section focuses on the model's predictions for the pressure, temperature, and velocity parameters for these three airfoils at various angles.

The maximum thickness and camber of the NACA747A415 airfoil are 15% at 40.2% chord and 3% at 35% chord, respectively. This airfoil was kept entirely separate from the model during training and used in the test dataset after training to verify the model's performance. Figure 5.1 and Figure 5.2 compare pressure parameters for the angle of attack values of 0° and 8° . The mean errors shown in the difference plots correspond to the version of MAPE that is not multiplied by 100.

When the pressure predictions are considered, it can be said that both numerically trained models and models trained with physics-based loss function throughout the flow have promising predictions. The improved communication between nearby pixels because of the physics-based training can be used to explain the softer transitions in contour plots across the entire flow. However, some degradation in the predictions in areas with shock formations could still occur even when using the physics-based loss function. While this degradation may appear quite severe in contour plots, it is clear from the mean percentage error for the entire flow that it doesn't exceed 1–1.5%. Pixel-based errors of up to 20% are possible in shock regions where the maximum error is observed, especially at low angles of attack (like $AOA = 0^\circ$). Although it would be theoretically expected that the momentum conservation loss function calculated and trained with the HLLC method would improve the predictions in regions with shocks, the anticipated effect couldn't be observed on primitive variables. Despite a large network, the lack of pixel density in this region and the effort of the method used to improve the overall flow behavior might be considered the main reasons for this issue. As a solution, the number of pixels can be increased by two or four times by increasing the PPI. However, a significant increase in the computational time and a negligible increase in overall accuracy across the entire domain suggest that there is not much potential to solve these close-boundary problems using two or four times as many pixels.

Note that a key objective of using a momentum flux-based physics-based loss function is to improve momentum conservation across the entire flow and smooth transitions between neighboring pixels. Therefore, the fact that pressure values do not significantly degrade across the flow is deemed sufficient for this study.

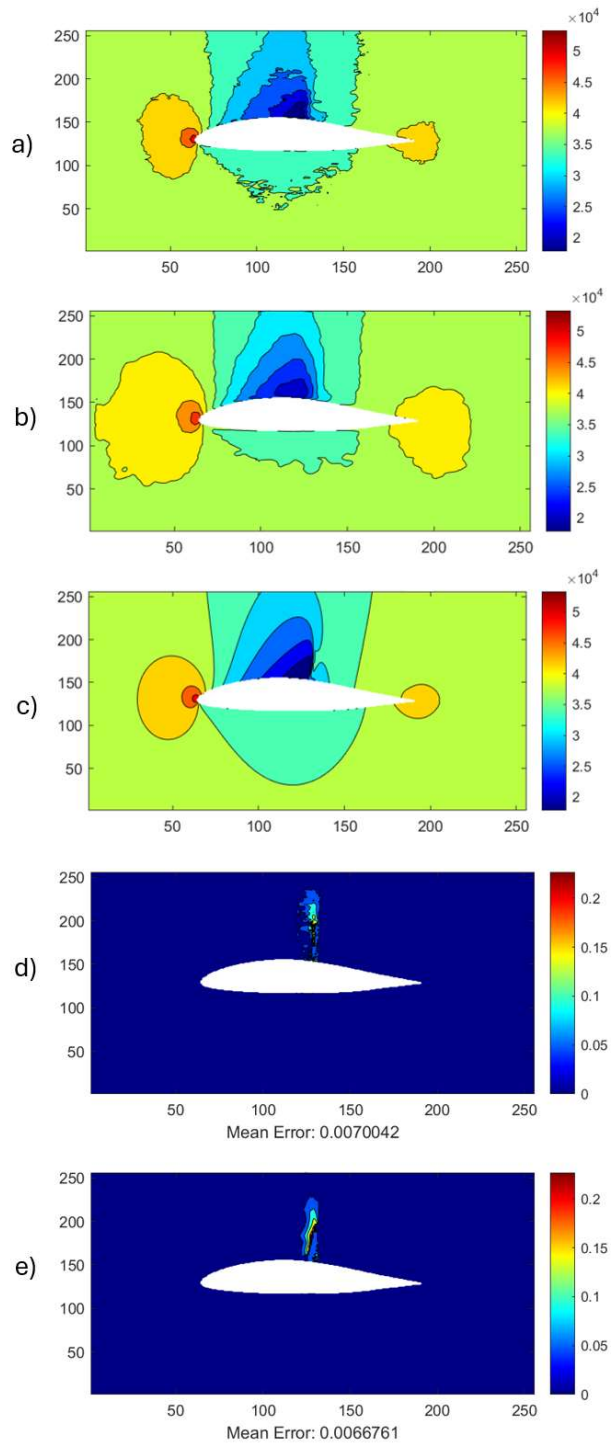


Figure 5.1. NACA747a415 Pressure Predictions at 0° Angle of Attack
a) non-Physics-based Training, b) Physics-based Training, c) GT, d) non-Physics-based Predictions & GT (MAPE / 100), e) Physics-based Predictions & GT (MAPE / 100)

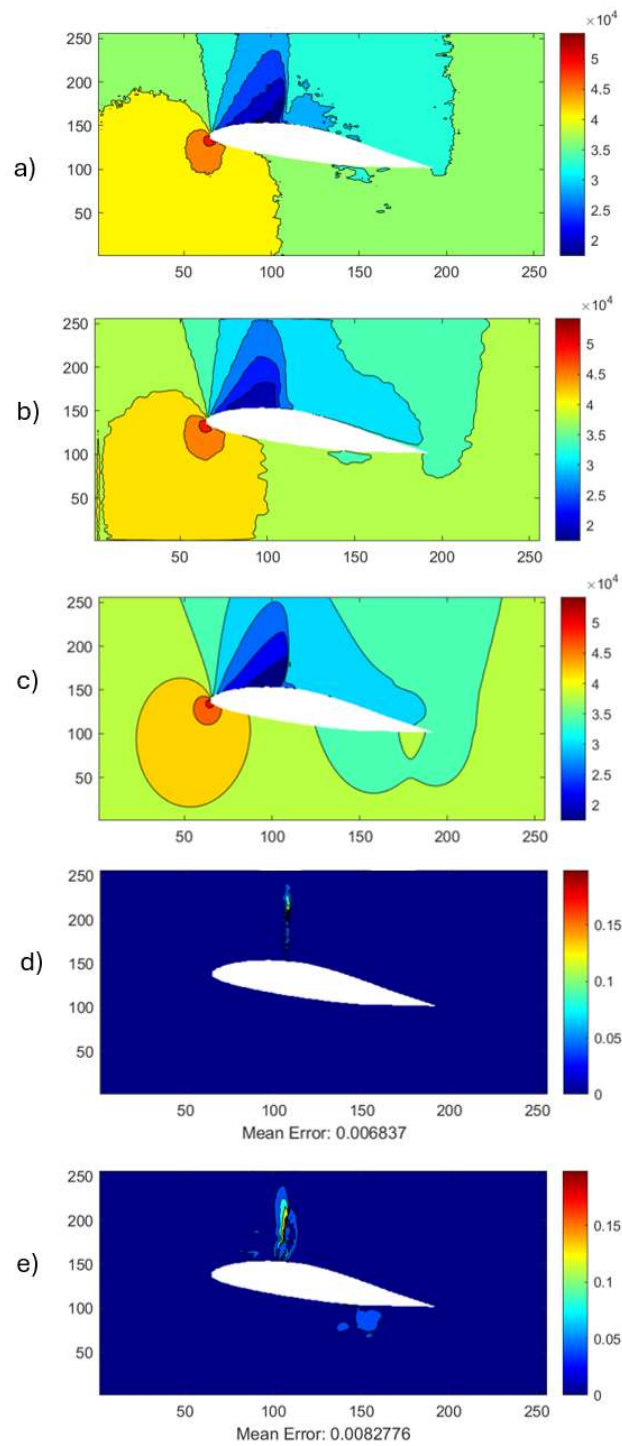


Figure 5.2. NACA747a415 Pressure Predictions at 8° Angle of Attack
a) non-Physics-based Training, b) Physics-based Training, c) GT, d) non-Physics-based Predictions & GT (MAPE / 100), e) Physics-based Predictions & GT (MAPE / 100)

The NACA2421 airfoil has a maximum thickness of 21% at 30% chord and a maximum camber of 2% at 40% chord. Similar to the NACA747A415 airfoil, this airfoil was kept entirely separate from the model during training and was used in the test dataset after training to verify the model's performance. Figure 5.3 and Figure 5.4 compare the temperature parameters for the angle of attack values of 4° and 12° . Note that the mean errors in the difference plots represent the non-scaled version of MAPE.

Training for temperature parameters has been a relatively easier process throughout the study compared to training pressure and velocity components. This is because the desired temperature profiles are generally closer in data, with fewer outliers and gradients, making the training more successful in numerically and physics-based trained segments. In this training, the average errors range from 0.1% to 0.3%. Therefore, achieving a reliable observation and comment on the difference between numerically and physics-based trained models is not easy. Nonetheless, upon examining the model's behavior, it can be concluded that errors increase in shock locations and the wake region at higher angles (where gradients between the wake region and free stream are higher). Similarly, maximum errors have also been observed in these locations.

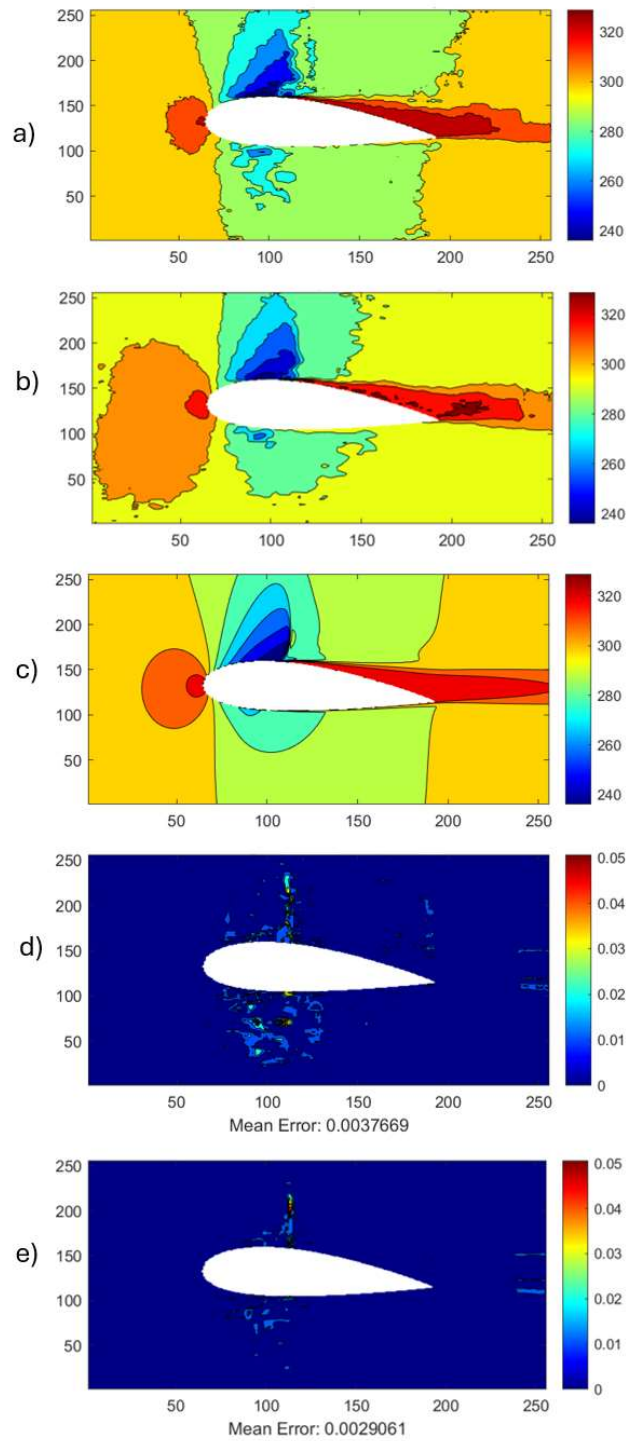


Figure 5.3. NACA2421 Temperature Predictions at 4° Angle of Attack
a) non-Physics-based Training, b) Physics-based Training, c) GT, d) non-Physics-based Predictions & GT (MAPE / 100), e) Physics-based Predictions & GT (MAPE / 100)

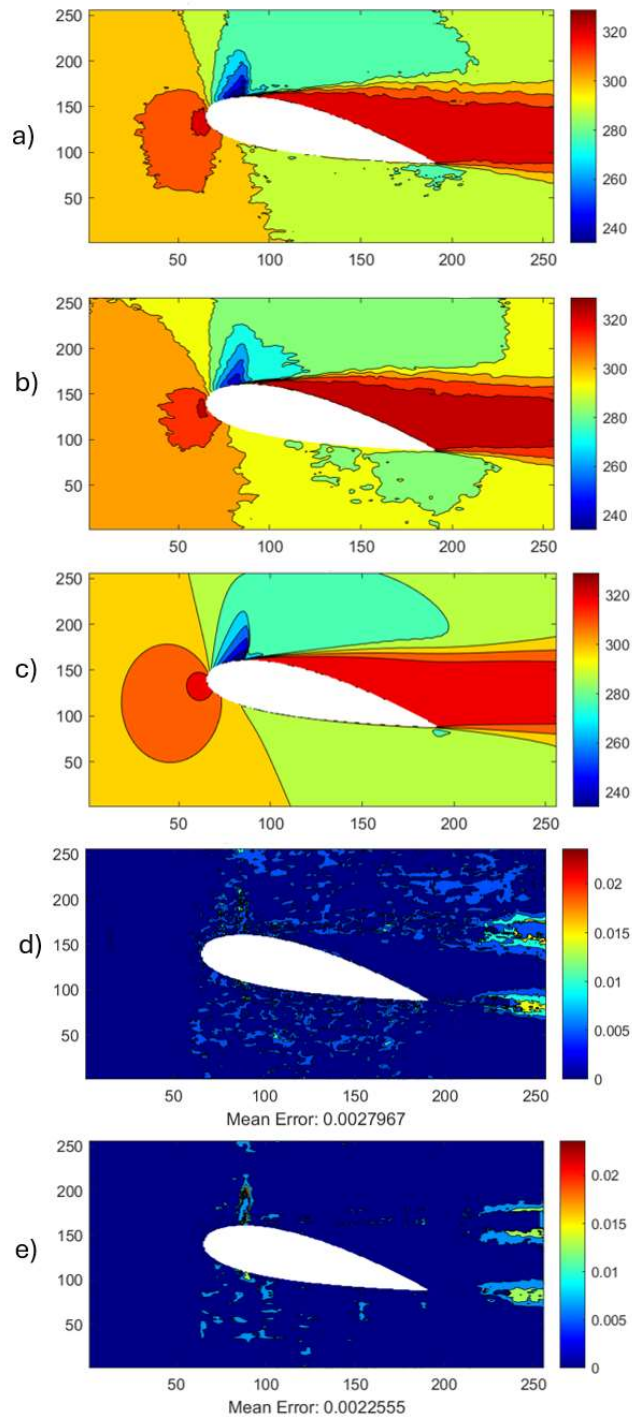


Figure 5.4. NACA2421 Temperature Predictions at 12° Angle of Attack
a) non-Physics-based Training, b) Physics-based Training, c) GT, d) non-Physics-based Predictions & GT (MAPE / 100), e) Physics-based Predictions & GT (MAPE / 100)

The SC1012R8 airfoil has a maximum thickness of 12% at 27.8% chord and a maximum camber of 2.7% at 21.8% chord. Like the other airfoils, this airfoil was kept separate from the model during training and was used in the test dataset after training to verify the model's performance. Figure 5.5 and Figure 5.6 compare the velocity parameters for the angle of attack values of 6° and 14° . Note that the mean errors in the difference plots represent the non-scaled version of MAPE.

Although the velocity components were trained in separate channels, the presentation here is provided by taking the magnitudes to make more accurate observations about the behavior of flow prediction. In the results for velocity channels, errors have dramatically increased, especially in the wake region corresponding to the boundaries of the 256×256 grid. In addition to that, similar to pressure and temperature parameters, errors are observed to increase in shock regions and wake region formations at higher angles. Generally, prediction errors are most pronounced in regions where gradients are highest. For low angles, the model has been less accurate in making predictions before and after the shock location. Furthermore, as a positive impact of the physics-based loss function, smoother transitions between contours can be observed in Figure 5.5 and Figure 5.6.

Throughout the training sessions aimed at observing the behavior of primitive variables, it was observed that the uniform image resolution in the CNN causes challenges in making accurate predictions in regions with large pressure gradients. Especially considering the mesh density near the airfoil surfaces in CFD solutions of the airfoils, it can be stated that the model focuses on overall improvements rather than regional focus, resulting in difficulties in making accurate predictions in these mentioned regions.

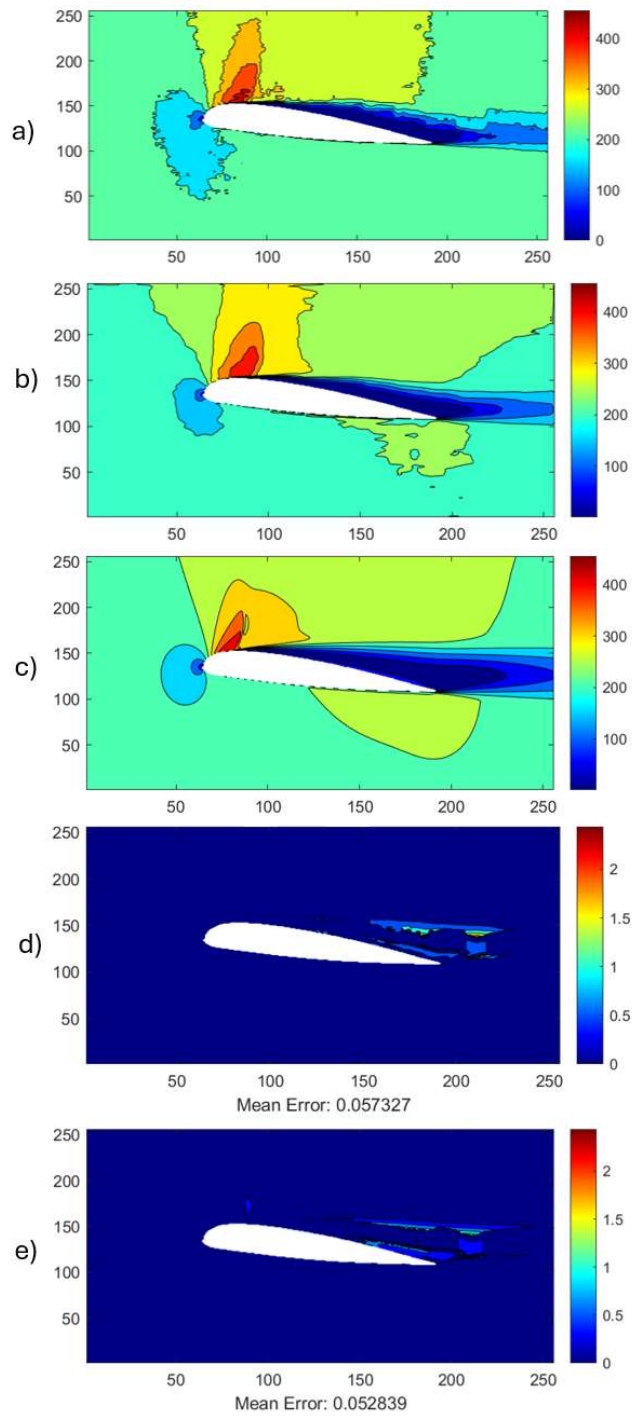


Figure 5.5. SC1012R8 Velocity Predictions at 6° Angle of Attack
a) non-Physics-based Training, b) Physics-based Training, c) GT, d) non-Physics-based Predictions & GT (MAPE / 100), e) Physics-based Predictions & GT (MAPE / 100)

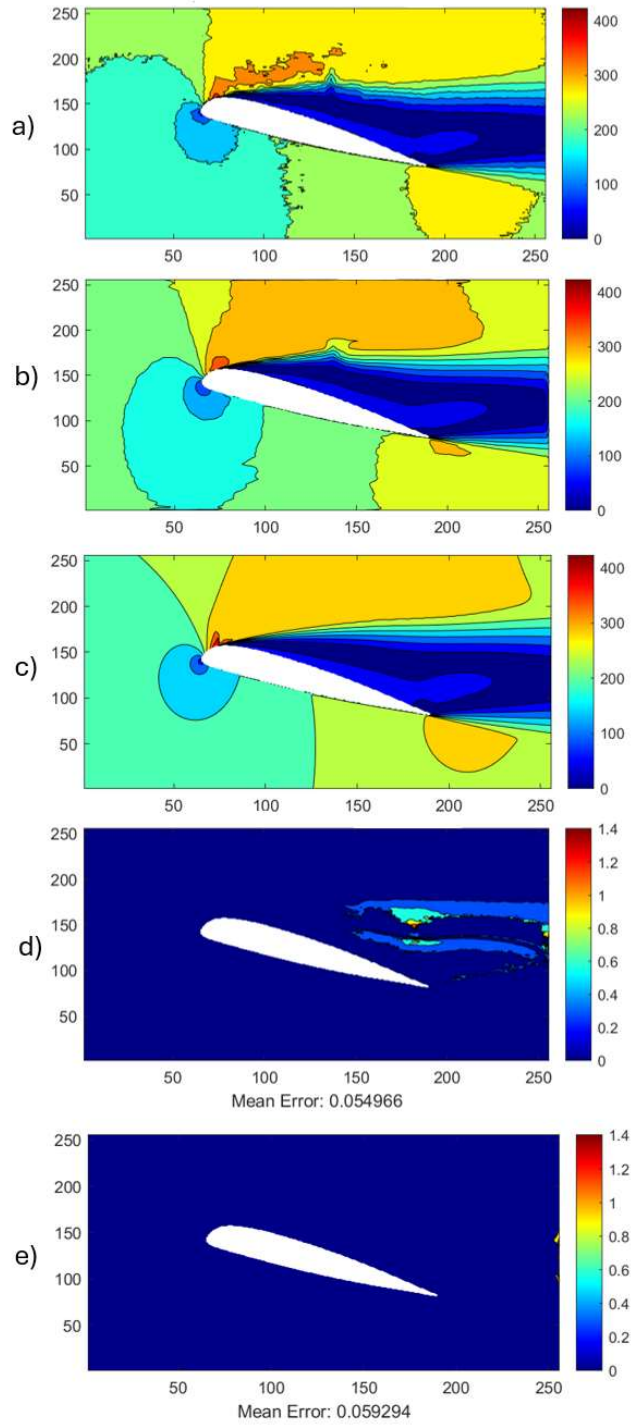


Figure 5.6. SC1012R8 Velocity Predictions at 14° Angle of Attack
a) non-Physics-based Training, b) Physics-based Training, c) GT, d) non-Physics-based Predictions & GT (MAPE / 100), e) Physics-based Predictions & GT (MAPE / 100)

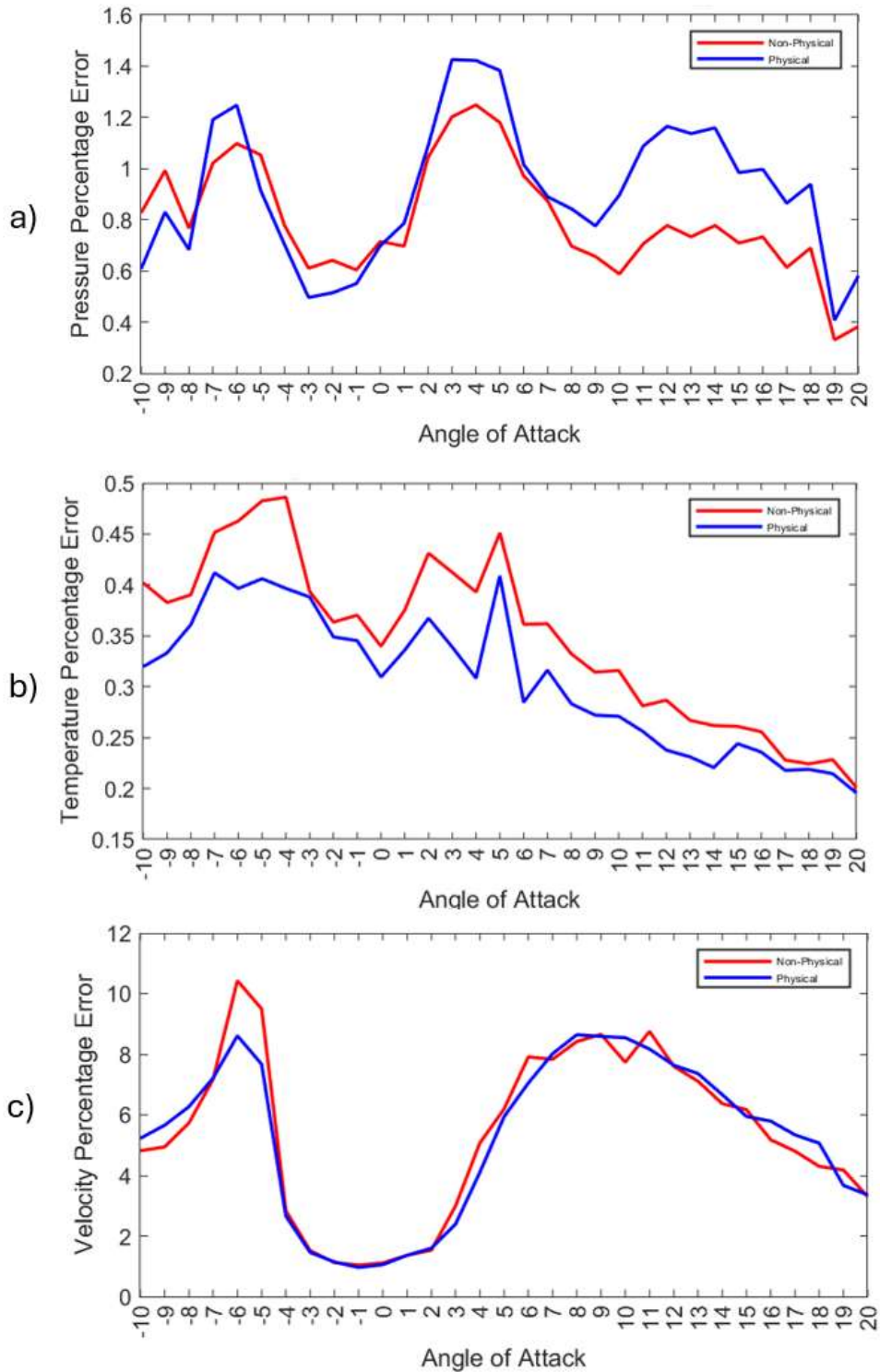


Figure 5.7. Percentage Errors of a) Pressure in NACA747a415, b) Temperature in NACA2421, c) Velocity in SC1012r8 for Different Angles of Attack Values

As can be observed in Figure 5.7, activating the physics-based loss function has a slightly negative impact rather than improving the primitive variables. However, it should be noted that the percentage errors are not excessively high in both methods, and both models can make predictions with sufficient accuracy for future use in the study.

5.2 Effects of Model Training on Momentum Conservation

In this subsection, which demonstrates the impact of HLLC momentum flux-based physics-based loss function, questions the overall momentum conservation of the flow, and compares it with the ground truth momentum conservation, the horizontal and vertical momentum conservation will be examined. When considering the momentum conservation in the horizontal direction, the influence of pressure and the horizontal component of velocity is quite significant. Therefore, when interpreting the results, the conclusion will be that 'the more finely-tuned the pressure and horizontal velocity components are, the more momentum conservation can be improved. A similar inference can be made for vertical momentum conservation, focusing on the pressure and vertical velocity components.

This training session was carried out with 150 Epochs and a learning rate of $3e-4$, followed by another 100 Epochs of training with a learning rate of $1e-4$ to complete the process. This approach deliberately kept the learning rates lower than those in the numerical part where primitive variables were trained, attempting to prevent the degradation of the trained parameters from the numerical section during this training.

The NPL9626 airfoil has a maximum thickness of 12.1% at 30.9% chord and a maximum camber of 0.7% at 30.9% chord. This airfoil was kept separate from the model during training and was used in the test dataset after training to verify the model's performance. Figure 5.8, Figure 5.9, Figure 5.10, Figure 5.11, Figure 5.12, and Figure 5.13 give the horizontal and vertical momentum conservation fields for the selected airfoil at 0° , 8° , and 16° angle of attack values. Note that there is no

comparison between numerical training and the ground truth because numerical training outputs have almost no physical meaning. The numbers on the scales represent the forces divided by the dynamic pressure ($\frac{1}{2}\rho V_\infty^2$) to be in a non-dimensional form, and the mean errors under the difference plots represent the non-multiplied form of MAPE.

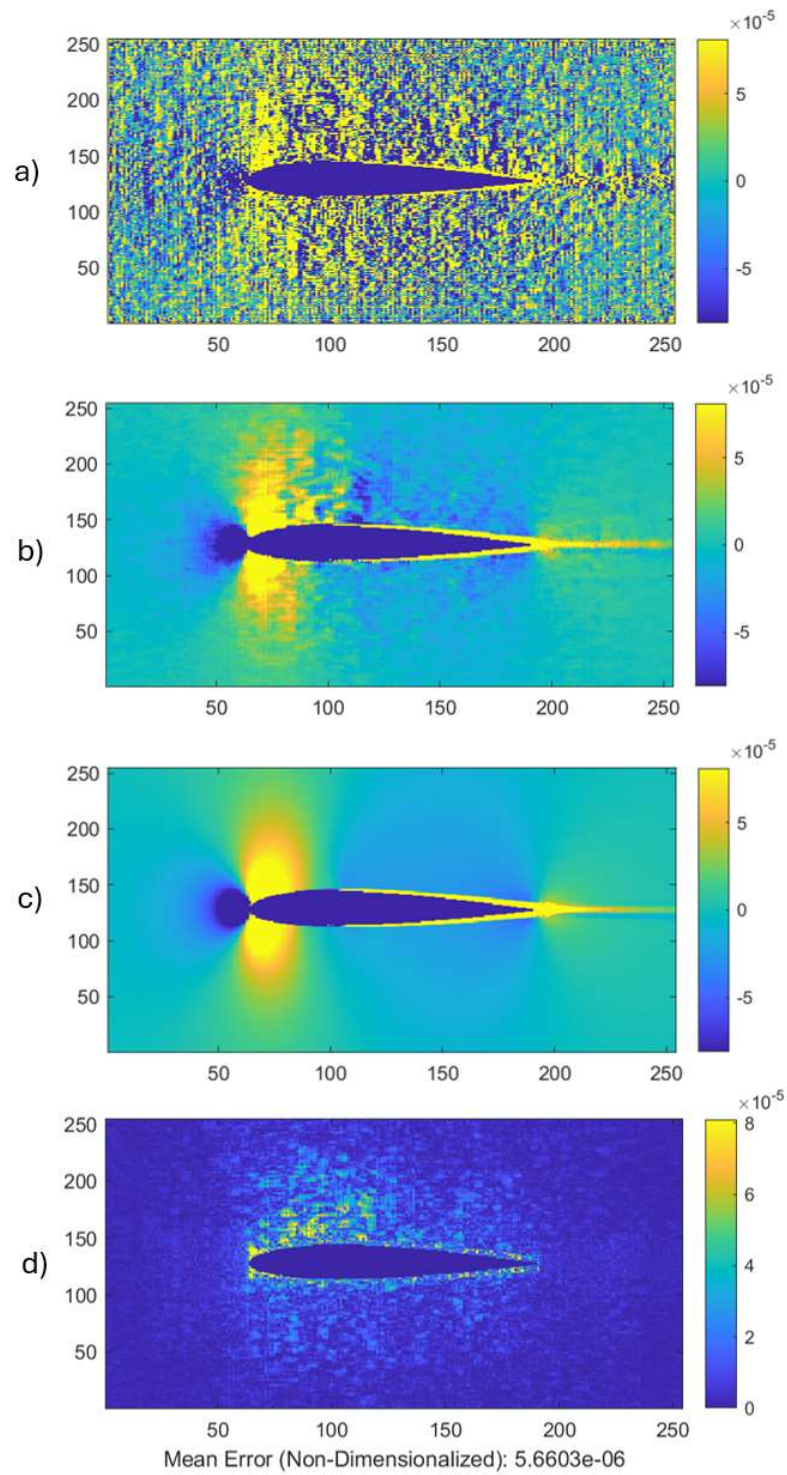


Figure 5.8. NPL9626 at 0° Angle of Attack, X-Momentum Conservation Law Outputs for a) Numerical Training, b) Physics-based Training, c) GT, d) Physics-based & GT Difference (Non-Dimensionalized by $\frac{1}{2}\rho V_{\infty}^2$)

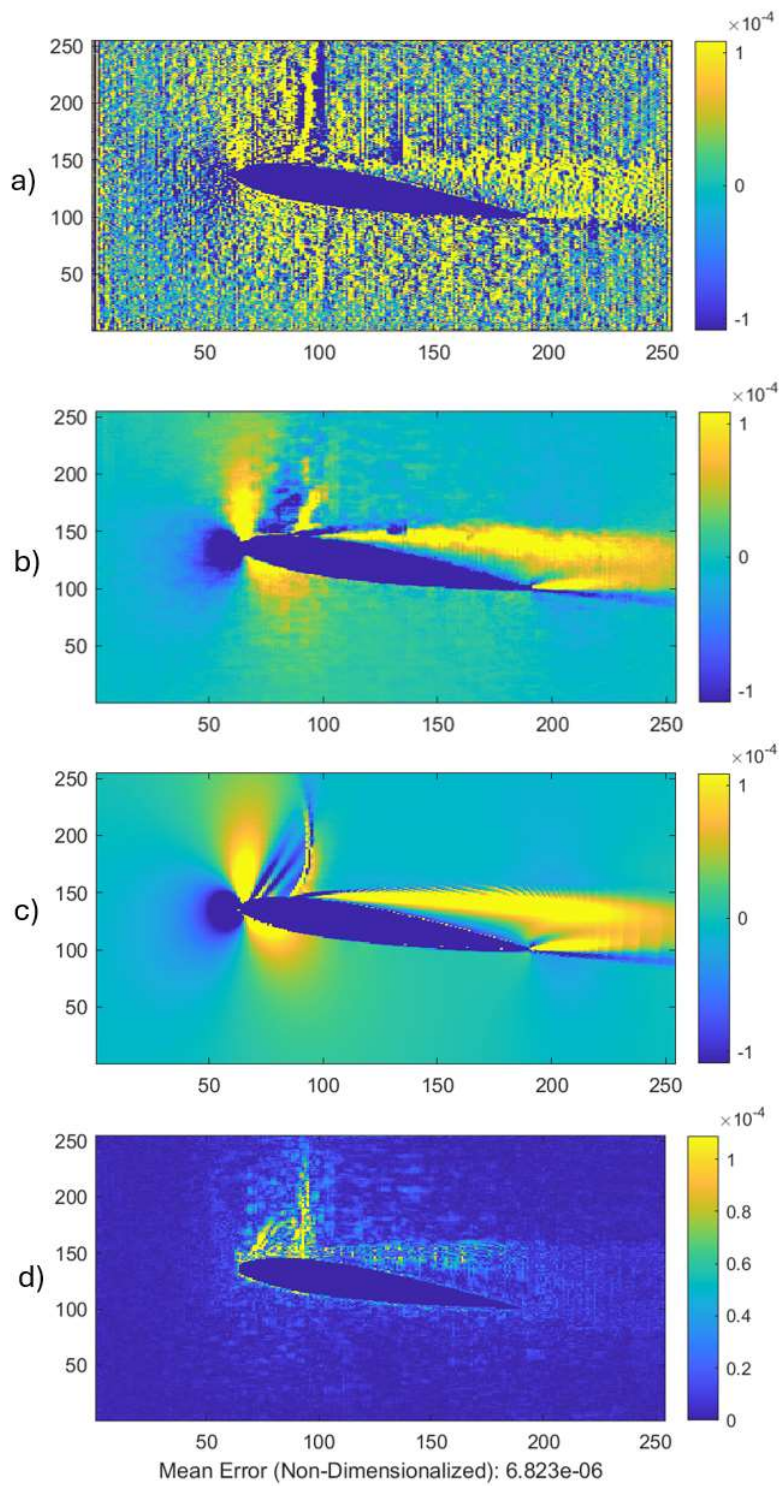


Figure 5.9. NPL9626 at 8° Angle of Attack, X-Momentum Conservation Law Outputs for a) Numerical Training, b) Physics-based Training, c) GT, d) Physics-based & GT Difference (Non-Dimensionalized by $\frac{1}{2}\rho V_{\infty}^2$)

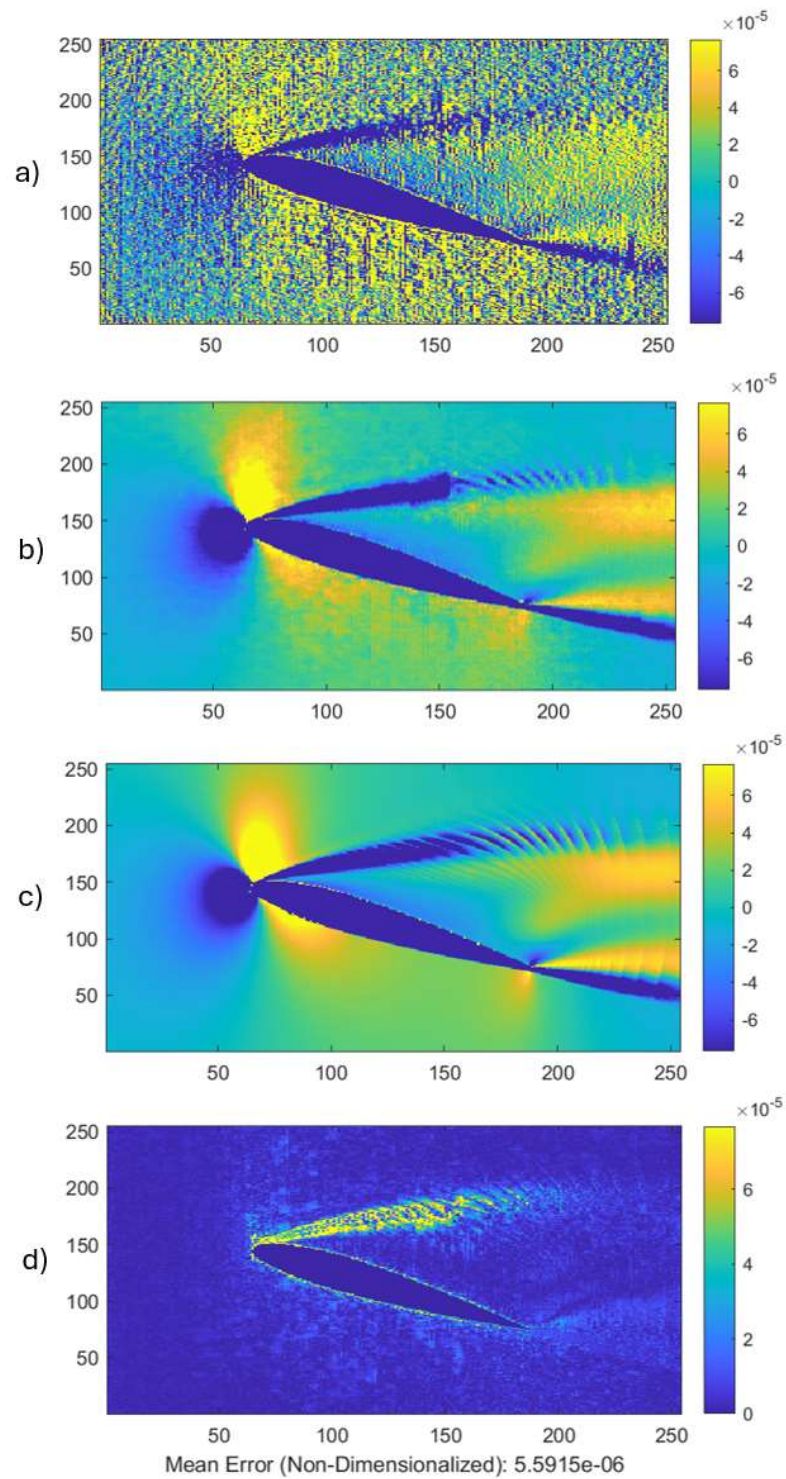


Figure 5.10. NPL9626 at 16° Angle of Attack, X-Momentum Conservation Law Outputs for a) Numerical Training, b) Physics-based Training, c) GT, d) Physics-based & GT Difference (Non-Dimensionalized by $\frac{1}{2}\rho V_\infty^2$)

As evident from Figure 5.8, Figure 5.9, Figure 5.10, Figure 5.11, Figure 5.12, and Figure 5.13, notable errors are observed in the predictions of shock formation and wake region at high angles of attack, like in primitive variables. The model trained using the HLLC momentum flux-based loss function shows significant errors in predicting the magnitudes in shock formation regions. However, it accurately predicts the shock location, which is close to the actual solution. Therefore, it can be inferred that the model successfully understands and predicts the flow trend, provided that magnitudes of the momentum flux around high gradient areas are not considered. Numerical training outputs show large errors all around the field. Therefore, the model does not know much information about the physics behind the problem domain. Adding a physical momentum-based loss function and relating the neighboring pixels with each other in terms of momentum fluxes makes the ML architecture realize more about the flow and forces acting on the field.

As the angle of attack increases, the accumulation of errors in downstream transition regions can easily be observed. While the flow patterns in these regions are relatively captured, there are still differences between the predictions and the actual solutions when compared.

It should be noted that throughout the training, the weights of each control volume, which corresponds to each pixel, were kept the same with respect to each other, meaning that every pixel's contribution to momentum conservation was treated equally. Despite introducing proximity to the airfoil as a parameter to the model through distance functions or similar filtering methods for close boundary solutions, the most accurate results were obtained when no weight distribution was applied throughout the domain.

In regions far from the airfoil, the effect of the HLLC momentum flux-based loss function was maximized and managed to minimize the errors (compared to the numerical training). As mentioned in the next subsection, the ability to calculate drag and lift coefficients accurately and away from the airfoil played a significant role in achieving such low errors in the results of this study.

When examining the loss plots in Figure 5.14, the Mean Absolute Error values for X and Y momentum conservation during the physics-based training sequence following the numerical training can be observed. Since the flow around the airfoil is mostly dominated by the horizontal velocity component, the error in the X momentum conservation component is initially higher. As a result, the model exhibited behavior to preserve the X momentum, which had a higher initial error. Similar loss plots were obtained for all angle of attack values, and the momentum conservation components converged to values that were not significantly different from each other. The fact that the validation dataset shows similar behavior and converges with the training dataset can be considered a promising sign of the training's reliability.

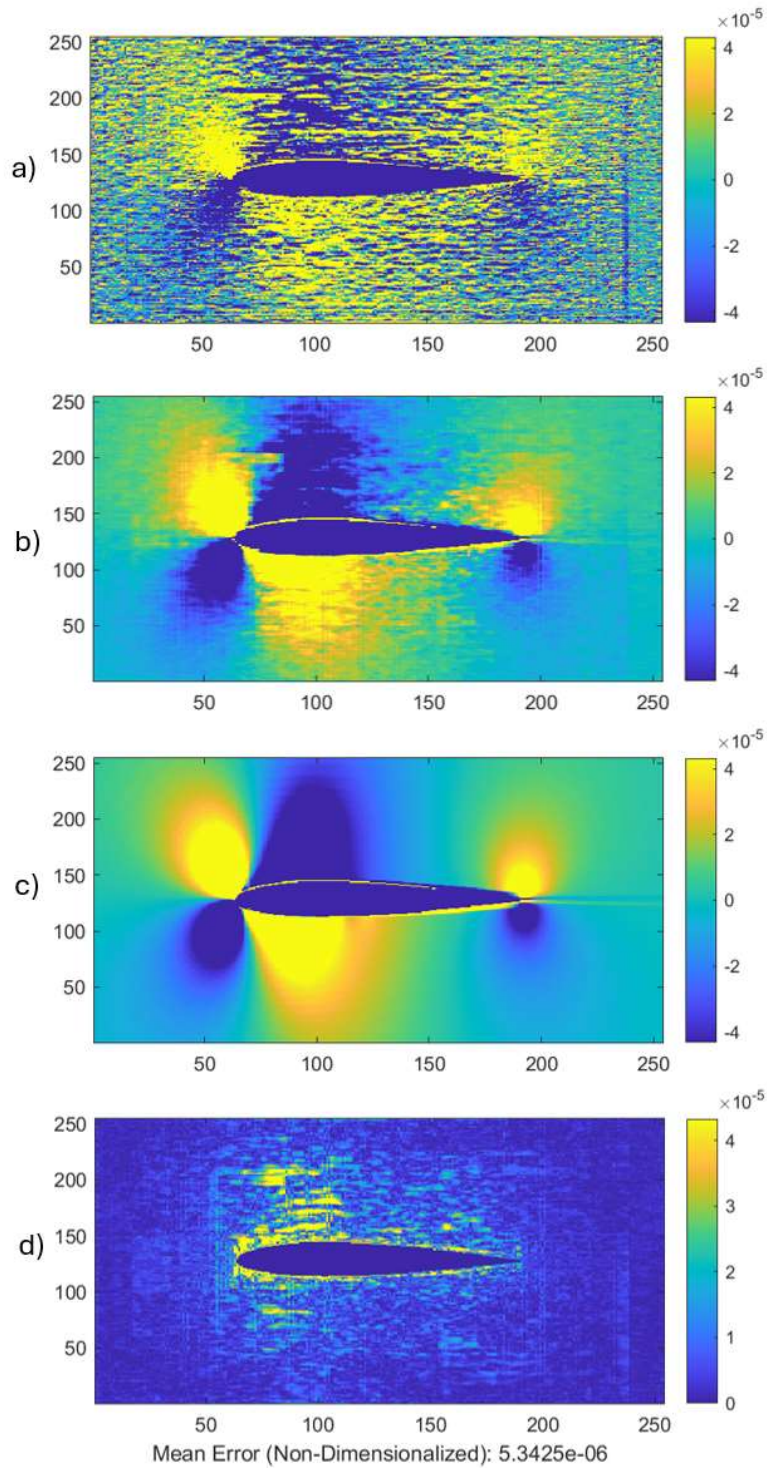


Figure 5.11. NPL9626 at 0° Angle of Attack, Y-Momentum Conservation Law Outputs for a) Numerical Training, b) Physics-based Training, c) GT, d) Physics-based & GT Difference (Non-Dimensionalized by $\frac{1}{2}\rho V_{\infty}^2$)

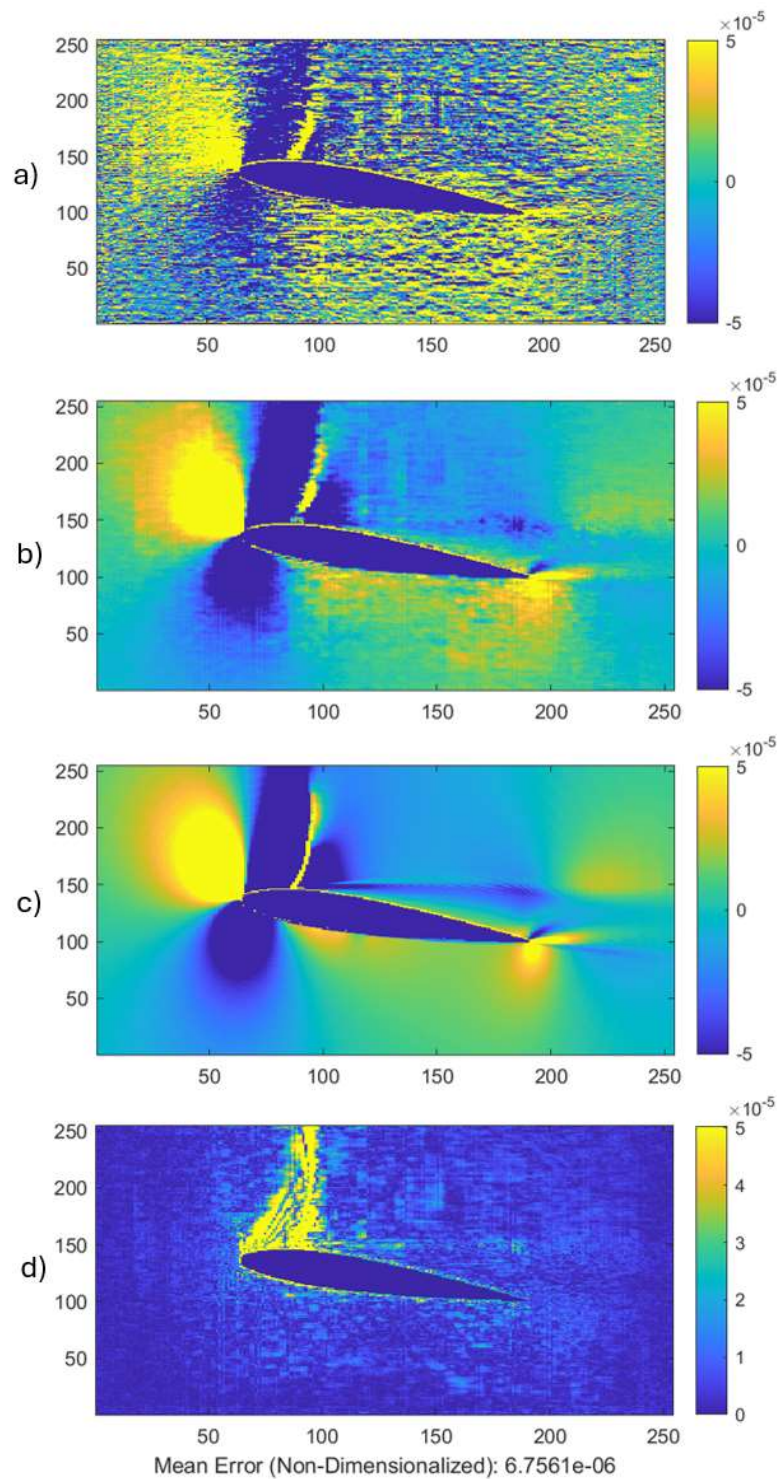


Figure 5.12. NPL9626 at 8° Angle of Attack, Y-Momentum Conservation Law Outputs for a) Numerical Training, b) Physics-based Training, c) GT, d) Physics-based & GT Difference (Non-Dimensionalized by $\frac{1}{2}\rho V_\infty^2$)

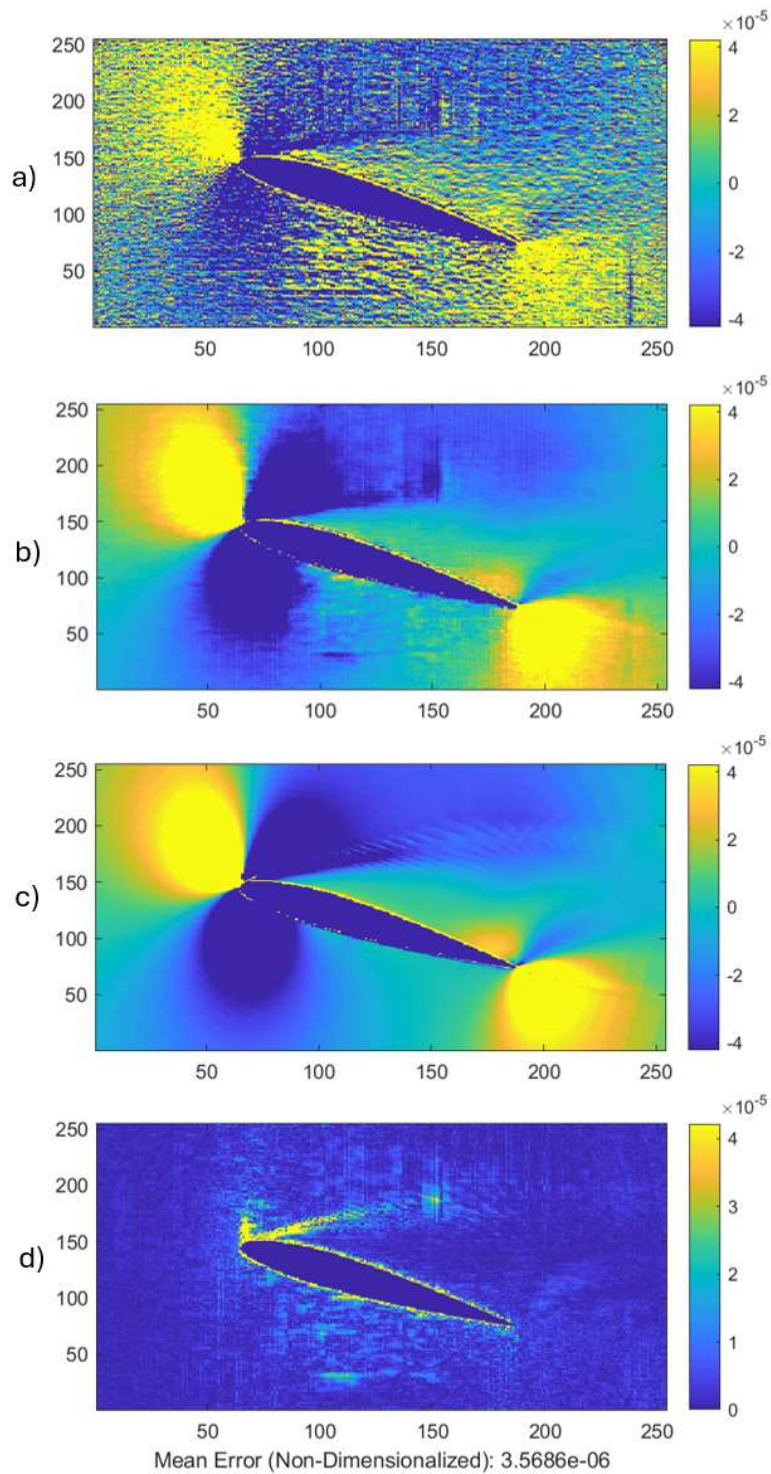


Figure 5.13. NPL9626 at 16° Angle of Attack, Y-Momentum Conservation Law Outputs for a) Numerical Training, b) Physics-based Training, c) GT, d) Physics-based & GT Difference (Non-Dimensionalized by $\frac{1}{2}\rho V_{\infty}^2$)

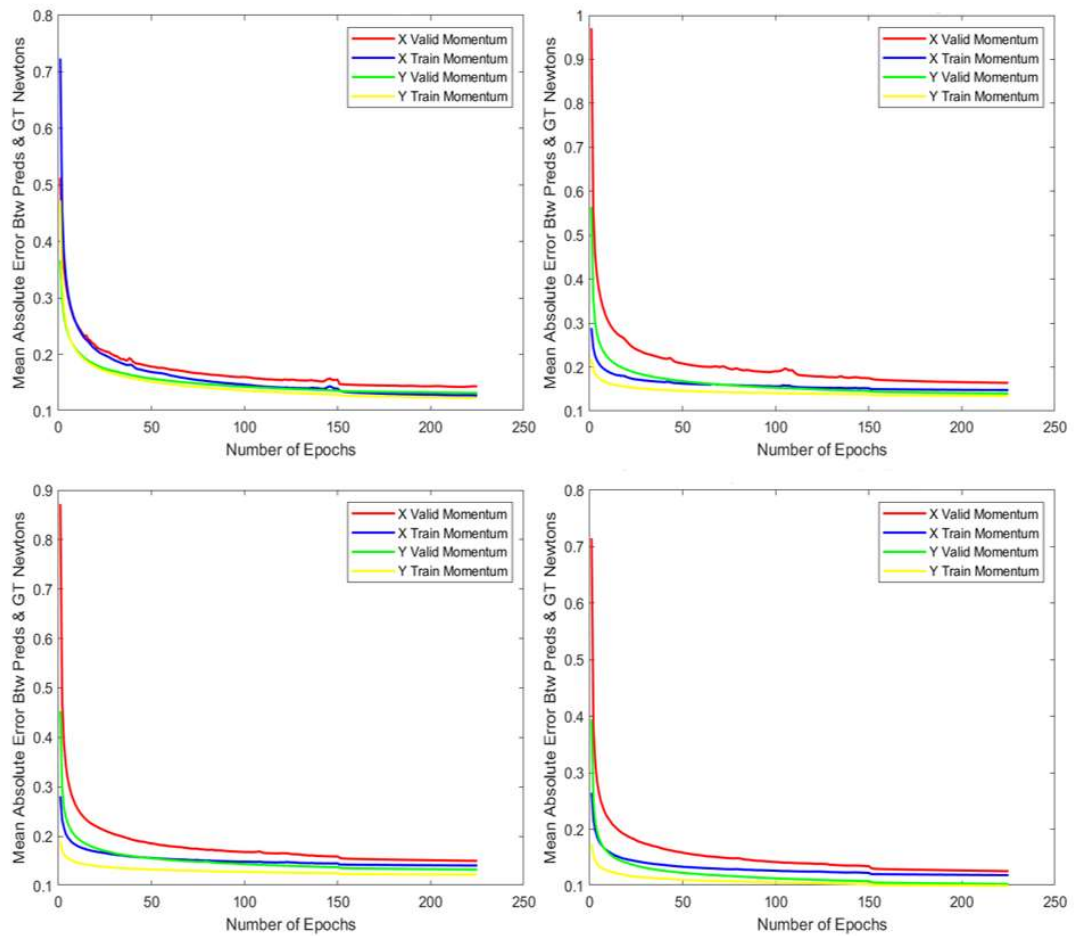


Figure 5.14. X & Y Momentum Losses of Both Training and Validation Set

5.3 Drag and Lift Coefficients Comparison Between CFD Results and the Ground Truth

5.3.1 Drag and Lift Calculation Methodology, Selection of the Rectangular Domains

In our study, CFD results are processed through Paraview's Resample to Image filter in a selected rectangular region. This gives us a pixelated domain and prepares them for the training in the deep learning section. However, during this operation, primitive variables are approximated into pixels, which have bigger sizes than any

CFD mesh, and this causes some errors. Even though these errors are relatively small, their effect on the whole domain should be observed. In this sub-section, the focus will be on how the selected rectangular region affects the ground truth values. In this section, different sizes of the rectangular region will be investigated by holding the pixels per inch (PPI) constant. Later, PPI will be increased, and its effect will be observed.

While designing an airfoil, there are crucial concepts such as lift coefficient (CL), drag coefficient (CD), and lift-to-drag ratio. Our study's most important objective is to achieve accurate CD and CL estimations with the improved model. Drag and lift forces can be expressed in terms of the pressure forces acting on the airfoil. Pressure forces act perpendicular to the surface, and the sum of these forces in the wind and normal to the wind directions yield drag and lift forces.

Drag and lift forces have pressure and viscous components. The viscous component is significant at locations near the wall. Therefore, neglecting the viscous drag, the pressure drag and lift are calculated by evaluating the momentum flux and integrating the pressure on the control surface. The perpendicular component to the flow direction of the net force, \vec{F} , is called the lift force, and the component parallel to the flow direction is named as the drag force.

$$\vec{F} = \oint p \vec{n} dA \quad 5.1$$

In our study, calculating these coefficients using the line integral method is not applicable because our pixelized domain has a shallow resolution along the airfoil surface compared to the CFD mesh. Moreover, the approximations made in the data preparation process might increase the error dramatically when we consider the low number of pixels used in this calculation. Therefore, the control volume methods and the momentum conservation equations can be combined to calculate the net forces acting on the airfoil. This process can be done by benefitting from the HLLC approximate Riemann Solver. Thus, the drag and the lift coefficients can be calculated more accurately away from the airfoil surface. Another advantage is the selection of the rectangular domain can be evaluated with the errors made in CD and

CL. This process aims to calculate net forces acting (using momentum conservation) on the pixels that are far from the airfoil and sum them up to calculate the net forces acting on the airfoil. One important note is that viscous effects are ignored in these calculations. A demonstration of the rectangular domains can be seen in Figure 5.15.

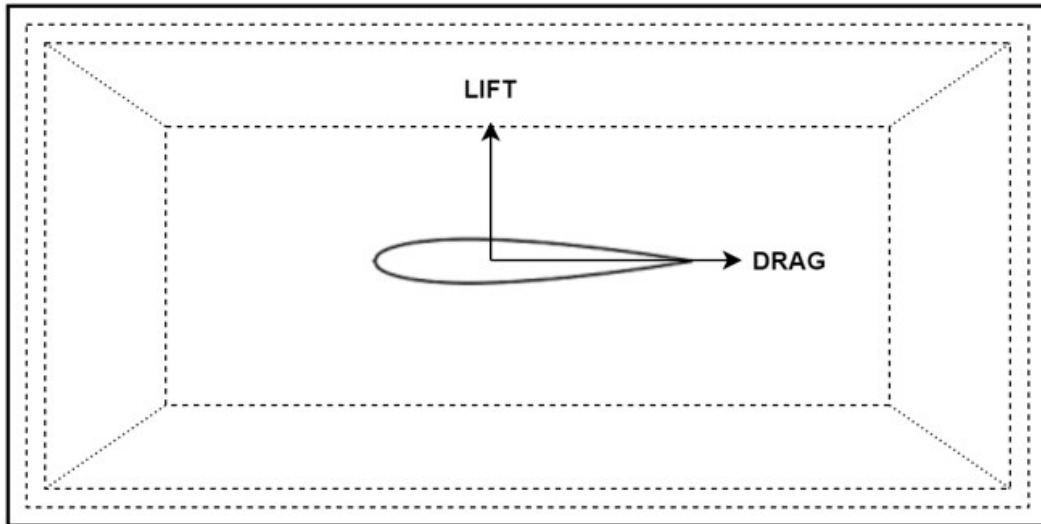


Figure 5.15. The Rectangles Utilized in Drag and Lift Calculations

For this part of the study, three different airfoils (E347, NACA654221, and RAF69) are chosen with three different angles of attack (AOA = 0, 5, 10, 15, and 20 degrees) configurations. Note that all these setups are trained in the deep learning part as well, meaning that they are taken from our training dataset.

Momentum equations can be utilized to calculate drag and lift forces. An important point is that the viscous effects become less effective as the distance to the airfoil increases. That is why the distance to the airfoil should be far enough to calculate drag and lift forces accurately, ignoring the viscous effects. Therefore, in addition to the current selection of domain (256x256 2:1 chord length), various test cases are observed, such as 512x512 4:2 chord length and 1024x1024 8:4 chord length. Lastly, PPI is increased, and the 1024x1024 4:2 chord length case is investigated as well. In the current selection of domain, the control volume are squeezed from the left and the right 64 times, starting from the outer sections until it hits the airfoil surface. That is, the control volume method is applied to a series of sub-rectangular sections of the

selected rectangular domain. In the end, there are 64 values for both drag and lift coefficients, as seen in Figure 5.16a and Figure 5.16b. The effect of the distance to the airfoil surface on both drag and lift coefficients can be observed from the figures as an increase in errors. One idea is to average these values of CD and CL for each configuration. Thus, the absolute average error of CD and CL can be calculated. However, closer regions have bigger errors on both drag and lift. The averaging operation can be done by either taking all the values or selecting some of them, especially those far away. Thus, for the rest of this section, these averaging operations will include two parts, which are the first 60 rectangles averaged and the first 16 rectangles averaged. The reason behind number 16 is that the first 16 values of CD and CL are much closer to the CFD values for most of the test cases.

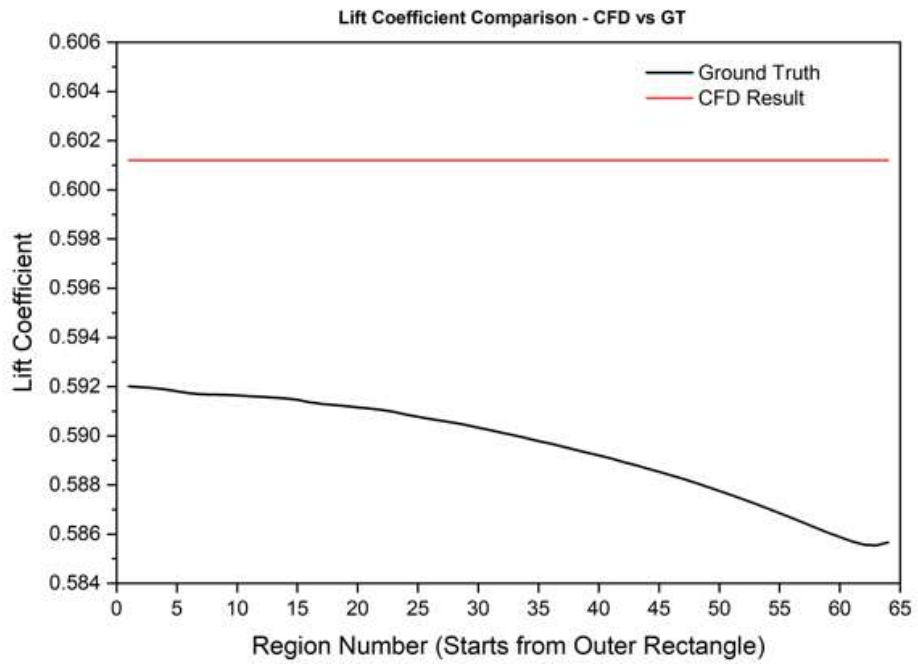
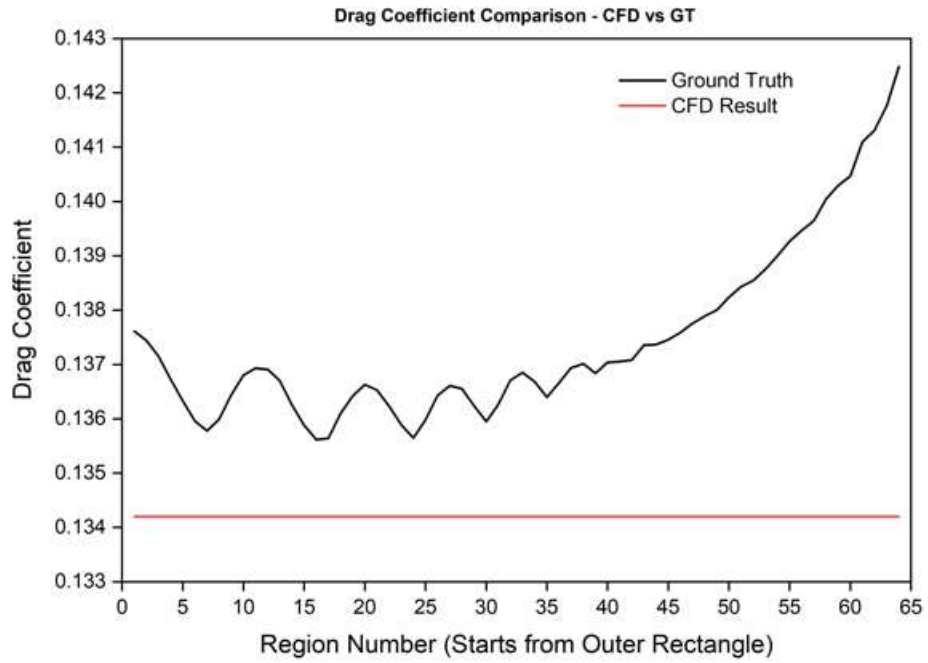


Figure 5.16. E347 at AOA=10. GT vs CFD results. a) CD, b) CL

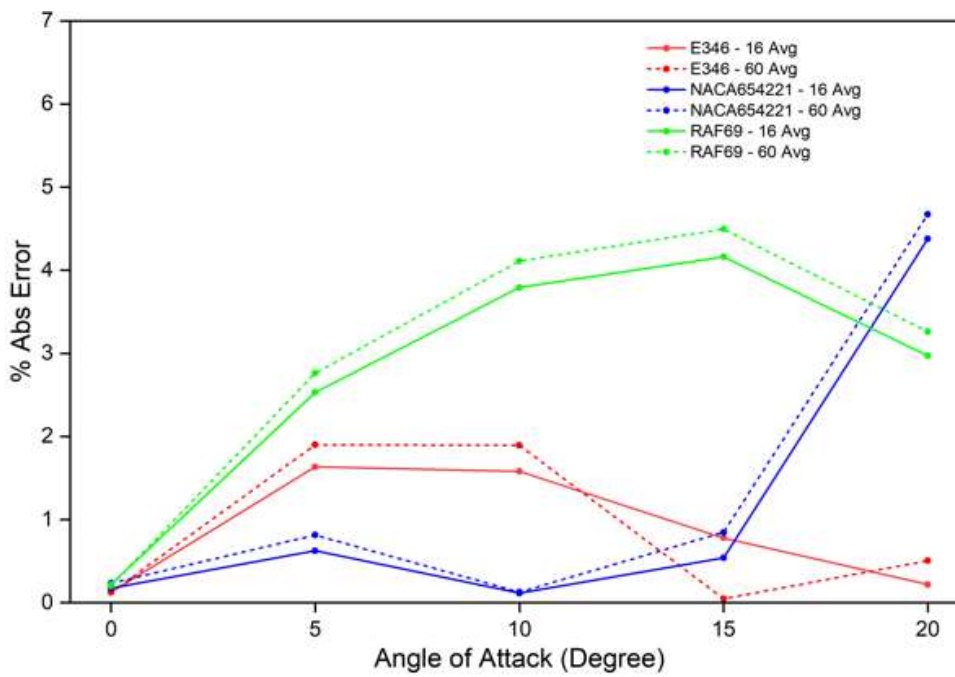
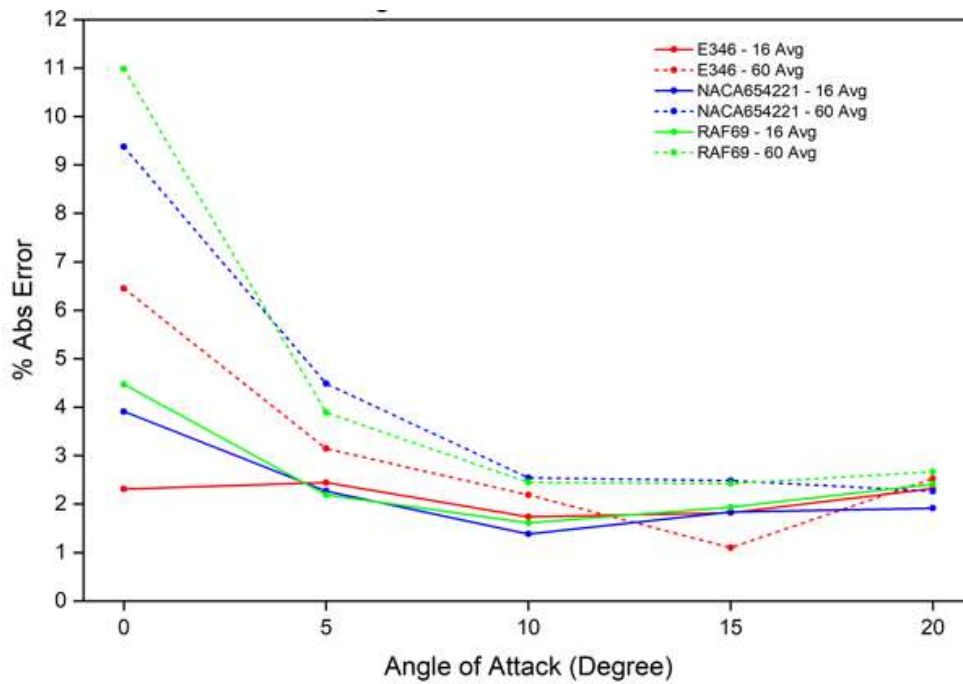


Figure 5.17. a) CD, b) CL Comparison between 16 Rectangles and 60 Rectangles Averaging Operation

Figure 5.17a and Figure 5.17b show that for both drag and lift coefficients, averaging the first 16 rectangles causes less ground truth error than 60 rectangles. Thus, the rest

of the study focuses on the version where 16 rectangles are averaged. Another important thing is that the absolute error percentage for the drag coefficient is higher than the lift coefficient's absolute error percentage. This trend can be seen in the whole dataset, and the reason behind it might be the viscous effects affecting the drag forces.

At this point, averaging the first 16 values of coefficients (outer rectangles) will be the default methodology while evaluating the rectangular region selection. Figure 5.18a, Figure 5.18b, and Figure 5.18c show that absolute errors for both drag and lift coefficients do not give a clear idea about how the domain selection should be.

Figure 5.18 gives exciting clues about the behaviors of the procedure. For instance, the 2:1C 256x256 domain (green lines) looks preferable to the others. Although the 8:4C 1024x1024 field (cyan lines) looks like giving more accurate results in some cases, the 2:1C 256x256 domain should be the selection when the dataset size is considered, as 1024x1024 pixels would mean a bigger data size. Another point is that errors usually drop when the angle of attack increases. For low degrees of angle of attack, the actual values of C_D and C_L are smaller compared to high numbers of the angle of attack. That is why the selection should be made by focusing more on the low angle of attack values. From this perspective, the 2:1C 256x256 field is still a reasonable selection. The last thing to note is that increasing the PPI for the 4:2C region (blue lines vs. red lines) does not decrease absolute errors considerably. It causes more errors for drag coefficients while changing almost nothing for the lift coefficients.

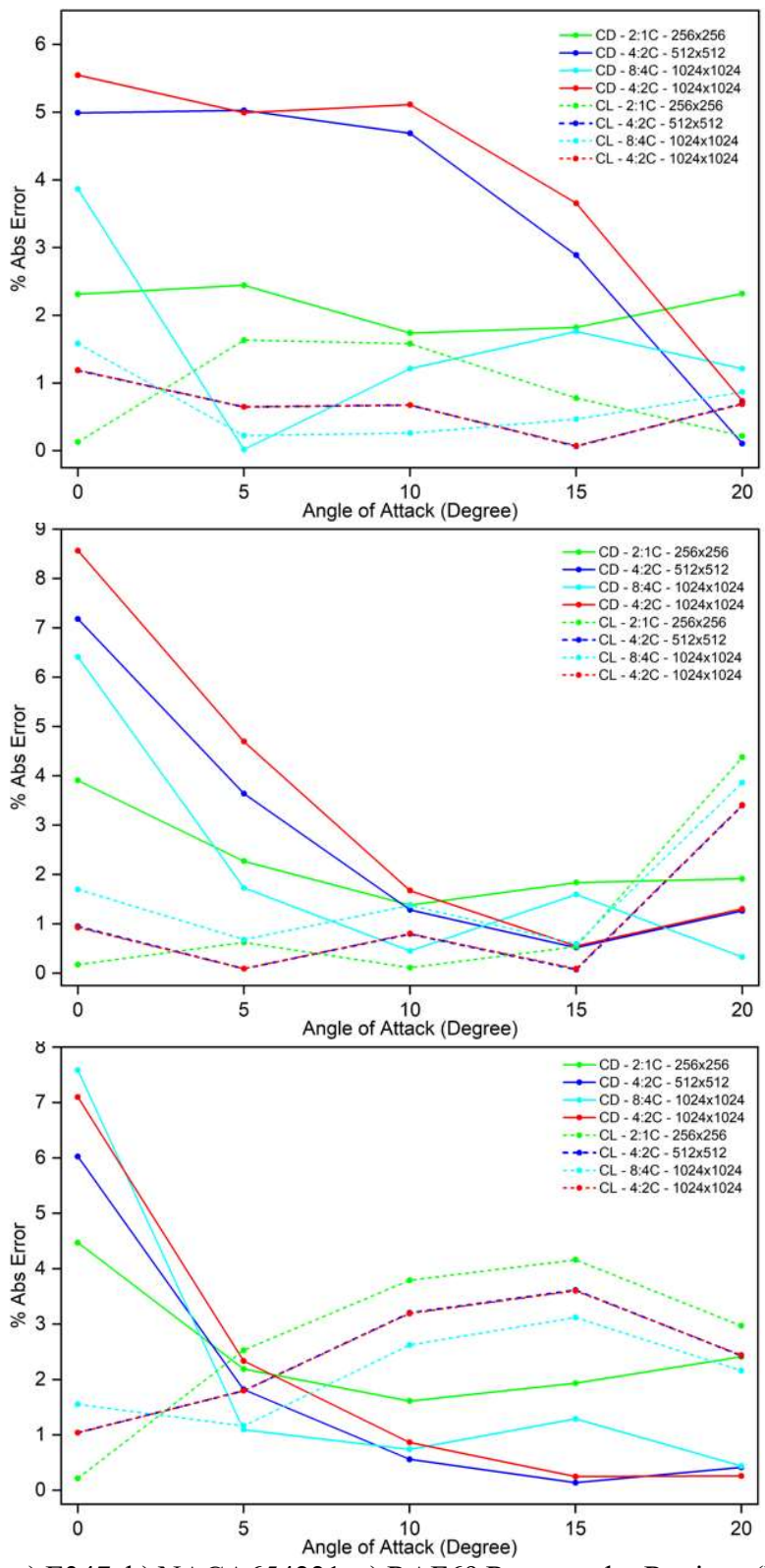


Figure 5.18. a) E347, b) NACA654221, c) RAF69 Rectangular Regions (Resolution) Comparisons for CD and CL

5.3.2 Effects of Model Training on Drag & Lift Coefficients

In this section, the drag and lift predictions for the FX61-184, M12, and RAE5215 airfoils, which are part of the test dataset and have not been seen by the model before, will be examined. Comments will be made on how the numerical and physics-based training yield results compared to the ground truth. The drag and lift coefficients are calculated based on momentum conservation within rectangular control volumes located throughout the domain, with the farthest rectangles from the airfoil being considered. These values are calculated within these rectangular areas and then reduced to a single value by averaging the values calculated over multiple control volumes. In Figure 5.19, drag and lift predictions for the mentioned three airfoils are presented for angle of attack values between -10 and 20 degrees of angle of attack. In addition to the ground truth data, drag and lift coefficients calculated directly from CFD solutions are also included in Figure 5.19. As mentioned in the data preparation section, since the CFD and ground truth coefficient values are quite close to each other, they appear almost identical in the line graphs.

Unfortunately, the results obtained through numerical training do not reach the level of accuracy that would contribute significantly to the design phase. However, the high accuracy of the coefficient predictions calculated away from the airfoil surface could still contribute significantly to the design phase, even though the model may not accurately predict the flow near the airfoil. Therefore, the fact that the gradients are lower, and the viscous effects are minimal away from the airfoil surface has formed the main motivation for accurately predicting these coefficients through physics-based training. When the HLLC momentum flux-based loss function is applied to the mentioned rectangular control volumes, the horizontal and vertical forces acting on the airfoil are obtained. These forces are then converted into drag and lift coefficients, given in Equation 5.2 and Equation 5.3, where D is the drag force, L is the lift force, and S is the reference area, which is equal to one chord.

$$C_D = \frac{D}{\frac{1}{2}\rho V_\infty^2 S} \quad 5.2$$

$$C_L = \frac{L}{\frac{1}{2}\rho V_\infty^2 S} \quad 5.3$$

Considering the overall behavior of drag and lift coefficient predictions, it can be observed that lift coefficients are more accurately predicted in numerical training. The lower Mean Absolute Error of Y momentum conservation compared to X momentum conservation before physics-based training with loss function can be accepted as a reasonable explanation for this trend. On the other hand, after training with the HLLC momentum flux-based physics-based loss function, the results appear much closer to the values obtained from ground truth and CFD solutions. The fact that drag and lift coefficients in these sessions, where each angle is trained separately, provide results so close to the desired values suggests that training with HLLC momentum flux-based physics-based loss function could yield even more successful outcomes with larger datasets.

When examining the drag polar curves created for these three airfoils in the test dataset in Figure 5.20, it can be observed that errors after numerical training are concentrated on the horizontal axis, specifically on the drag coefficient side. In contrast, with training using physics-based loss function, despite neglecting friction drag and viscous effects within the formulation, drag and lift coefficients can be predicted with a maximum error of 3-4%.

There are some challenges of defining the airfoil shape using pixels. For instance, the need for high resolution, the presence of high gradients due to boundary layer formation, and factors such as separation and turbulence. These cause to make predictions difficult in close boundaries. That is why, this study was conducted using the control volume method in the far field, for aerodynamic flow coefficient calculations. This methodology can be considered as an alternative to calculate the coefficients by utilizing pressure coefficient distribution through the pixels on the wing itself.

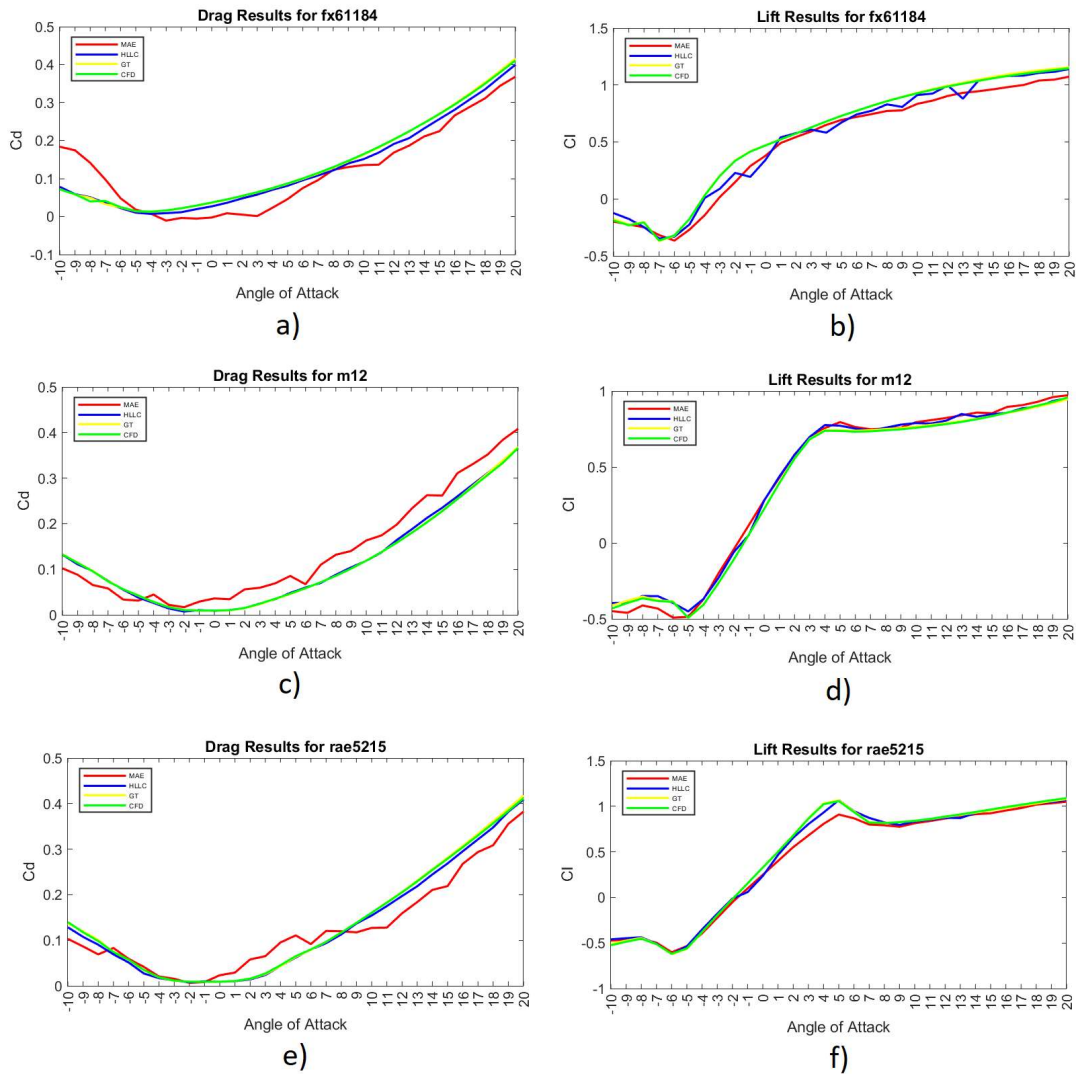


Figure 5.19. Drag & Lift Coefficient Predictions of FX61184, M12, and RAE5215

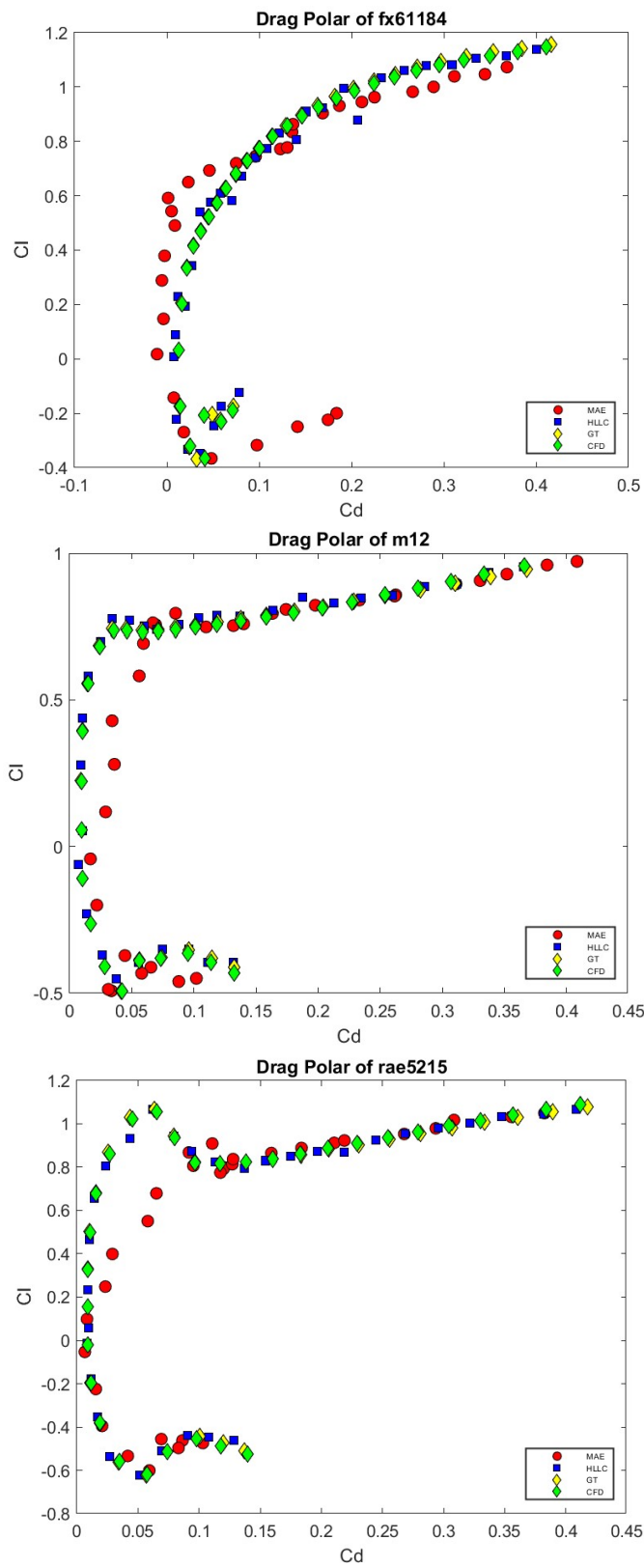


Figure 5.20. Drag Polar Curves of FX61184, M12, and RAE5215

CHAPTER 6

CONCLUSION AND FUTURE WORKS

Using the HLLC Riemann Solver and the Spalart-Allmaras Turbulence model, data from 204 different airfoil cases at 0.7 Mach and 6 million Reynolds Number were obtained for this study. These data were preprocessed to create a format suitable for deep learning training. Numerical simulations were first performed, and then, in addition to the typical numerical loss functions used in these networks, the HLLC momentum flux-based physics-based loss function was included in the Convolutional Neural Networks. With this addition, we aimed to enhance predictions for both primitive parameters in the specified flow region and predictions for net forces acting on each pixel across the domain. These predictions were then used to obtain data on the drag and lift coefficients. Both the results of the numerical training and physics-based training solutions were compared with the ground truth data. The outcomes show that models trained with the additional HLLC momentum flux-based physics-based loss function can produce better predictions of net forces acting on the pixels and more accurate drag and lift coefficient estimations.

Momentum fluxes and the net forces are calculated at each pixel with the HLLC momentum flux-based physics-based loss function. Then, different fluxes to and out of the pixel are summed up. This summation is tried to be conserved in the ML architecture.

The models trained with the HLLC momentum flux-based physics-based loss function had trouble making predictions in regions with high gradients, such as shock formation zones and wake regions, as shown by the obtained results of primitive variables and momentum conservation. However, when considering the entire domain, it is possible to see an overall beneficial effect of adding this HLLC momentum flux-based loss function. Errors in drag and lift coefficients, as well as

horizontal and vertical momentum conservation, have all decreased throughout the flow.

Drag and lift coefficient calculations are made in the control volumes, rectangles, away from the airfoil. This results in the usage of more pixel information, as more pixels are utilized when rectangle size increases, and also lets us neglect the viscous effects, which increase dramatically around the airfoil surface.

Airfoils in this study were trained separately for each angle. As a result, the dataset between -10 and 20 degrees can be trained collectively to extend the study and produce more precise results. This method can also be seen as a study targeting predictions at non-integer angles. The network presented in this study could be replaced with another neural network that aims to produce even more accurate results. One other improvement that can be applied is that another loss function with physical meaning can be introduced to the model. Importing an entropy-based loss function may improve the predictions, especially where the shock formations occur. Lastly, the importance of the outer pixels can be increased, and more precise results can be obtained in the outer rectangles in terms of drag and lift coefficients.

REFERENCES

- [1] I. Abbott and A. Von Doenhoff. Theory of Wing Sections. *New York: Dover Publications*, 1959.
- [2] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.
- [3] 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.
- [4] A. da Silva, J. Cavalcanti, and F. Catalano. Prediction of pressure distribution on wings using neural networks. in *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2008.
- [5] Jin X, Cheng P, Chen WL, Li H. Prediction model of velocity field around circular cylinder over various Reynolds numbers by fusion convolutional neural networks based on pressure on the cylinder. *Phys Fluids* 30(4):047105, 2018.
- [6] Sekar V, Jiang Q, Shu C, Khoo BC. Fast flow field prediction over airfoils using deep learning approach. *Phys Fluids* 31(5):057103, 2019.
- [7] X. Hui, J. Bai, H. Wang, and Y. Zhang. Fast pressure distribution prediction of airfoils using deep learning. *Aerospace Science and Technology*, vol. 105, 2020.
- [8] Yilmaz E, German B. A convolutional neural network approach to training predictors for airfoil performance. In: *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, p 3660, 2017.
- [9] Yuan Z, Wang Y, Qiu Y, Bai J, Chen G. Aerodynamic coefficient prediction of airfoils with convolutional neural network. In: *Asia-Pacific International Symposium on Aerospace Technology*. Springer, pp 34–46, 2018.
- [10] Portal-Porras, Koldo & Fernandez-Gamiz, Unai & Zulueta, Ekaitz & Ballesteros-Coll, Alejandro & Zulueta, Asier, CNN-based flow control device modelling on aerodynamic airfoils. *Scientific Reports*. 12. 10.1038/s41598-022-12157-w, 2022
- [11] Wu, Ming-Yu & Wu, Yan & Yuan, Xin-Yi & Chen, Zhihua & Wu, Wei-Tao & Aubry, Nadine, Fast Prediction of Flow Field around Airfoils Based on Deep Convolutional Neural Network. *Applied Sciences*. 12. 12075. 10.3390/app122312075, 2022.
- [12] Zhang Y, Sung WJ, Mavris DN. Application of convolutional neural network to predict airfoil lift coefficient. In: *AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, p 1903, 2018.
- [13] Olayemia, Olalekan & Salakoa, Oluwadolapo & Jinadu, Abdulbaqi & Obalalu, Adebowale & Anyaegbuna, Benjamin. Aerodynamic lift coefficient prediction

- of supercritical airfoils at transonic flow regime using convolutional neural networks (CNNs) and multi-layer perceptions (MLPs). *Al-Qadisiyah Journal for Engineering Sciences*. 16. 108-115. 10.30772/qjes.v16i2.955, 2023.
- [14] Viquerat, Jonathan, 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.
- [15] Chen J, Viquerat J, Hachem E. U-net architectures for fast prediction of incompressible laminar flows. *arXiv preprint*, 2019.
- [16] G. Sun, Y. Sun, and S. Wang. Artificial neural network based inverse design: Airfoils and Wings. *Aerospace Science and Technology*, vol. 42, pp. 415-428, 2015.
- [17] V. Sekar, M. Zhang, C. Shu, and B. Khoo. Inverse design of airfoil using a deep convolutional neural network. *AIAA Journal*, vol. 57, 2019.
- [18] Singh AP, Medida S, Duraisamy K. Machine-learning augmented predictive modeling of turbulent separated flows over airfoils. *AIAA J* 55:2215–2227, 2017.
- [19] 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.
- [20] Raissi, Maziar & Perdikaris, Paris & Karniadakis, George, Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations, *Journal of Computational Physics*. Volume 378, 2018.
- [21] Li, Jichao & Du, Xiaosong & Martins, Joaquim. Machine learning in aerodynamic shape optimization. *Progress in Aerospace Sciences*. 134. 100849. 10.1016/j.paerosci.2022.100849, 2022.
- [22] Kim, Min & Yoon, Hyun, Geometric modification for the enhancement of an airfoil performance using deep CNN. *Ocean Engineering*. 266. 113000. 2022.
- [23] Duru, C., Alemdar, H. & Baran, Ö.U. A deep learning approach for the transonic flow field predictions around airfoils. *Computers and Fluids*, vol 236. 2022.
- [24] Duru, C., Alemdar, H. & Baran, Ö.U. CNNFOIL: Convolutional encoder-decoder modeling for pressure fields around airfoils. *Neural Computing and Applications* 33.12: 6835-6849, 2021.
- [25] P.R. Spalart and S.R. Allmaras. A One-Equation Turbulence Model for Aerodynamics Flows. *AIAA Paper*, vol. 92, 1992.
- [26] D.P. Raymer. Aircraft Design: A Conceptual Approach. *AIAA Education Series*, 1992.

- [27] 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.
- [28] Toro, Eleuterio & Chakraborty, A., Development of a Riemann solver for the steady supersonic Euler equations. *The Aeronautical Journal*. 98. 325-339. 10.1017/S0001924000026890, 1994.
- [29] S. F. Davis. Simplified Second-Order Godunov-Type Methods. *SIAM J. Sci. Stat. Comput.*, 9:445–473, 1988.
- [30] B. Einfeldt. On Godunov-Type Methods for Gas Dynamics. *SIAM J. Numer. Anal.*, 25(2):294–318, 1988.
- [31] FlowPsi Guide Documentation, retrieved from: <https://github.com/libm31/FlowPsi/tree/master/guide>, 2023.
- [32] Turkel, Eli. Turkel, E.: Preconditioned Methods for Solving the Incompressible and Low Speed Compressible Equations. *Journal of Computational Physics* 72, 277-298. *Journal of Computational Physics*. 72. 277-298. 10.1016/0021-9991(87)90084-2, 1987.
- [33] M.V. Valueva and N.N. Nagarnov. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*, vol. 177, 2020.

APPENDICES

A. HLLC Mass and Momentum Flux Calculation Function, MATLAB Script

```
r1 = (pgl+pambient)/(Rt*Tl);
rr = (pgr+pambient)/(Rt*Tr);
h0l = Rt*Tl*gamma/gm1 + 0.5*(U1(1)^2 +U1(2)^2);
h0r = Rt*Tr*gamma/gm1 + 0.5*(Ur(1)^2 +Ur(2)^2);
EtaP = Eta+1.;
EtaM = Eta-1.;
srl = sqrt(r1);
srr = sqrt(rr);
roe_factor = 1./(srl+srr);
U_ = ((U1)*srl+(Ur)*srr)*roe_factor;
ut_ = (U_(1) *an(1)+U_(2) *an(2))-Us_n;
h0_ = (((h0l)*srl+(h0r)*srr)*roe_factor);
a_ = sqrt(max(gm1*(h0_-0.5*(U_(1)^2 +U_(2)^2)),0.0));
sigma = sqrt(max(ut_*ut_*EtaM*EtaM+4.*Eta*a_*a_,1e-30));
evl_ = 0.5*(ut_*EtaP - sigma);
evr_ = 0.5*(ut_*EtaP + sigma);
utla =U1(1) *an(1)+U1(2) *an(2);
utra =Ur(1) *an(1)+Ur(2) *an(2);
utl = utla-Us_n;
utr = utra-Us_n;
al2 = gamma*Rt*Tl;
ar2 = gamma*Rt*Tr;
sigmar = sqrt(max(utr*utr*EtaM*EtaM+4.*Eta*ar2,0.0));
sigmal = sqrt(max(utl*utl*EtaM*EtaM+4.*Eta*al2,0.0));
SL = min(evl_,.5*(utl*EtaP - sigmal));
SR = max(evr_,.5*(utr*EtaP + sigmar));
lmdot = r1*utl;
rmdot = rr*utr;
coef = 0.5*area;
if(SL>0)
    if(SR<0)
        AL=1;
    else
        AL=2.;
    end
else
    if(SR<0)
        AL=0.;
    else
        AL=1.;
    end
end
if(SR<0)
    if(SL>0)
        AR=1;
```

```

    else
        AR=2.;
    end
else
    if(SL>0)
        AR=0.;
    else
        AR=1.;
    end
end
if SL > 0 || SR < 0
    AM = 0;
else
    AM = 1;
end
AL= coef*AL;
AR= coef*AR;
AM= coef*AM;
massFlux = AL*lmdot+AR*rmdot;
massFlux = massFlux -AM*(SL*r1 + SR*rr);
momflux(1) = AL*lmdot*U1(1) + AR*rmdot*Ur(1) + an(1)*(AL*pgl+AR*pgr);
momflux(2) = AL*lmdot*U1(2) + AR*rmdot*Ur(2) + an(2)*(AL*pgl+AR*pgr);
momflux(1) = momflux(1)- AM*(SL*r1*U1(1) + SR*rr*Ur(1));
momflux(2) = momflux(2)- AM*(SL*r1*U1(2) + SR*rr*Ur(2));

SM = ((rr*utr*(SR-utr)-r1*utl*(SL-utl)+pgl-pgr)/...
      (rr*(SR-utr)-r1*(SL-utl)));
r1_star = r1*(SL-utl)/(SL-SM);
rr_star = rr*(SR-utr)/(SR-SM);
pg_star = r1*(utl-SL)*(utl-SM)+pgl;

rul_star = ((SL-utl)*r1*U1(1) + (pg_star-pgl)*an(1))/(SL-SM);
rvl_star = ((SL-utl)*r1*U1(2) + (pg_star-pgl)*an(2))/(SL-SM);
nur_star = ((SR-utr)*rr*Ur(1) + (pg_star-pgr)*an(1))/(SR-SM);
rvr_star = ((SR-utr)*rr*Ur(2) + (pg_star-pgr)*an(2))/(SR-SM);

CR = SR-abs(SM);
CL = SL+abs(SM);
massFlux = massFlux + AM*(CR*rr_star + CL*r1_star);
momflux(1) = momflux(1) + AM*(CR*rur_star + CL*rul_star);
momflux(2) = momflux(2) + AM*(CR*rvr_star + CL*rvl_star);

```

B. Encoder-Decoder Type of Network, Python Script

```
def CNNFOIL256():
    return nn.Sequential(
        OrderedDict(
            encoder1=models.Encoder(1, 2,
                                    kernel_size=7, stride=1, padding=0,
                                    norm=True, activation=nn.ELU()),
            encoder2=models.Encoder(2, 4, 8,
                                    kernel_size=6, stride=1, padding=0,
                                    norm=True, activation=nn.ELU()),
            encoder3=models.Encoder(8, 16, 32, 64, 128,
                                    kernel_size=4, stride=2, padding=1,
                                    norm=True, activation=nn.ELU()),
            decoder1=models.Decoder(128, 64, 32, 16, 4, 1,
                                    kernel_size=4, stride=2, padding=1,
                                    activation=nn.ELU()),
            pool=nn.AdaptiveAvgPool2d((256, 256))
        )
    )
```