TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Maschinenwesen, Institut für Strömungsmechanik, Professur für Strömungsmechanik

Research Project

Analysis of neural network architectures for predicting time-series data from simulations

submitted in partial fulfillment of the requirements for the degree "Master of Science"

Name	**Sumanth Reddy Yeddula**
born on	15/07/1998
in	India
submission date	13/02/2025
1st reviewer	Prof. Dr.-Ing. habil. Jochen Fröhlich
2nd reviewer	Dr.-Ing. Andre Weiner
Supervisor	M.Sc. Janis Geise

TECHNISCHE UNIVERSITÄT DRESDEN

B - 2448

Fakultät Maschinenwesen Institut für Strömungsmechanik, Professur für Strömungsmechanik

Task definition for the Research Project

Name: **Sumanth Reddy Yeddula** **Matr.-No.: 5124111**

Field of study Computational Modeling and Simulation

Title: **Analysis of neural network architectures for predicting time-series data from simulations**

Analyse von Architekturen neuronaler Netze zur Vorhersage von Zeitreihendaten aus Simulationen

Description:

Deep reinforcement learning (DRL) shows great potential for efficiently finding optimal control laws in high-dimensional state-action spaces. In fluid mechanics, training a controller (agent) using DRL is achieved by repeated execution of CFD simulations. The overall execution time of the training can be reduced significantly by replacing some of the simulations with surrogate models. Deep neural networks are common choice for building such models. The data prediction for training the DRL agent requires robust and accurate models to mitigate the effects of error propagation. This research project concerns the following tasks:

- Literature review of suitable network types for predicting time series data
- Perform simulations of an actuated 2D flow past a cylinder at varying Reynolds numbers
- Assess the prediction accuracy of force coefficients and pressure probes using fully-connected neural networks (FCNN)
- Train different model types using supervised learning and compare their prediction accuracy to the FCNN
- Analyze the prediction error depending on the hyperparameters of the chosen model

This project should conclude with a recommendation of suitable network architectures for predicting time series data from simulations, along with the influence of their hyperparameters on the prediction accuracy.

First supervisor: M.Sc. Janis Geise
Second supervisor: Dr.-Ing. Andre Weiner

Handed out: 23.10.2024
Delivery date: 12.02.2024

Prof. Dr.-Ing. habil. J. Fröhlich
Supervising Professor

Abstract

Analysis of neural network architectures for predicting time-series data from simulations

This study investigates the application of advanced neural network frameworks specifically FCNNs and LSTM networks in predicting critical aerodynamic parameters, including drag coefficient, lift coefficient, and pressure distribution, around an oscillating cylinder. These models aim to capture the complex, non-linear dynamics and temporal correlations inherent in fluid-structure interactions. High-fidelity CFD simulations using OpenFOAM are conducted to generate time-series training data across varying oscillation frequencies and amplitudes.A surrogate model is developed to replace traditional CFD simulations, significantly reducing computational costs while maintaining predictive accuracy. This surrogate model, trained on CFD generated data, enables rapid inference of aerodynamic forces, making it a viable alternative for applications requiring real-time predictions. The surrogate model is suitable to be further integrated into a DRL framework, where it assists a reinforcement learning-based controller in predicting realtime aerodynamic forces and optimizing flow control strategies. FCNNs are employed as a baseline model to capture the spatially distributed aerodynamic parameters, offering insights into the ability of neural networks to predict steady-state or low-frequency flow behaviors.The predictive accuracy of both models is assessed using Mean Squared Error (MSE), Mean Absolute Error (MAE), and temporal consistency with reference CFD data, ensuring robust model evaluation. The study successfully demonstrates the feasibility of using neural network-based surrogate models to replace computationally expensive CFD simulations. The FCNN and LSTM models were trained on high-fidelity CFD-generated time-series data, effectively capturing the essential aerodynamic parameters around an oscillating cylinder. LSTMs exhibited superior performance in high-frequency oscillatory conditions, accurately predicting phase shifts, periodic patterns, and transient force variations. FCNNs proved effective in low-frequency or steady-state conditions, offering a computationally efficient approach for cases where temporal dependencies are minimal. The surrogate model significantly reduced computational costs while maintaining high accuracy, enabling near real-time predictions for aerodynamic force estimations. Future work will focus on expanding the surrogate model to three-dimensional flows and using hybrid models to enhance generalizability and interpretability. This research underscores the capability of data-driven models to revolutionize aerodynamic analysis by providing fast, accurate, and computationally efficient alternatives to conventional CFD simulations.

Contents

Acknowledgments

I would like to express my deepest gratitude to my supervisor,**M.Sc. Janis Geise**, for his invaluable guidance, encouragement, and unwavering support throughout the course of this research. His expertise and constructive feedback have been instrumental in shaping this work and refining my understanding of the subject matter.

I am profoundly grateful to my supervising professor, **Dr.-Ing. Andre Weiner**, for his insightful advice, mentorship, and continuous encouragement. His vast knowledge and research acumen have greatly influenced my academic growth, and his guidance has played a crucial role in the successful completion of this study.

Additionally, I extend my sincere appreciation to Prof.**Dr.-Ing. habil. Jochen Fröhlich** for giving me this opportunity and his invaluable support and for fostering a research environment that encourages learning, innovation, and excellence. His expertise and leadership have provided me with a strong foundation in my research endeavors. Beyond my academic mentors, I am deeply thankful to my family for their unwavering support, patience, and motivation throughout my research journey. Their encouragement, especially during challenging times, has kept me focused and determined to achieve my goals.

Finally, I would like to acknowledge all those who have contributed to this research in various ways. Their encouragement and assistance have been an integral part of this journey, and I am truly grateful for their support.

Nomenclature

Symbols	Description
$\mathbf{z}^{(l)}$	Weighted sum of inputs at layer l
$\mathbf{W}^{(l)}$	Weight matrix at layer l
$\mathbf{h}^{(l-1)}$	Output (or activation) from the previous layer
$\mathbf{b}^{(l)}$	Bias vector at layer l
∇	Nabla (operator used in vector calculus).

Latin Symbols	Description
i_t	Represents input gate.
f_t	Represents forget gate.
o_t	Represents output gate.
σ	Represents sigmoid function.
w_x	Weight for the respective gate (x) neurons.
h_{t-1}	Output of the previous LSTM block (at timestamp $t-1$).
x_t	Input at current timestamp.
b_x	Biases for the respective gates (x).
C_d	Drag coefficient.
C_l	Lift coefficient.
p_{11}	Pressure probe 11.
f	Frequency.
A	Amplitude.
u	Velocity (flow quantity).
p	Pressure (flow quantity).

Abbrevations

MSE	Mean Squared Error.
MAE	Mean Absolute Error.
LSTM	Long Short-Term Memory.
ReLU	Rectified Linear Unit.
FCNNs	Fully Connected Neural Networks.
Re	Reynolds number.
RNN	Recurrent Neural Network.
CFD	Computational Fluid Dynamics.
DRL	Deep reinforcement Learning.
DNN	Deep Neural Network.
CFL	Courant-Friedrichs-Lewy
FVM	Finite Volume Method

1 Introduction

The problem addressed in this project is the inefficiency and high computational cost associated with training controllers in DRL using traditional simulation methods, particularly in fluid mechanics. Current training techniques rely heavily on repeated execution of CFD simulations, which are time-intensive. The challenge lies in reducing the time and resources required for model training while maintaining accuracy. To address this, surrogate models, particularly DNNs, will be explored as an alternative to directly executing CFD simulations [10]. The primary objective is to assess the potential of FCNN and other machine learning models in accurately predicting time-series data, such as force coefficients and pressure readings, in place of expensive simulations. The project aims to identify the most suitable network architectures and hyperparameters to achieve accurate, efficient predictions for time-series data from simulations.This project focuses on the application of deep neural networks, specifically FCNN and LSTM networks, for predicting time-series data derived from simulations in fluid mechanics. The central challenge in the field of fluid dynamics is the computational cost and time associated with training controllers using traditional methods like CFD simulations. The goal of this project is to explore how surrogate models, in the form of FCNN and LSTM networks, can efficiently predict time-series data, such as force coefficients and pressure probes, to reduce the need for repeated CFD simulations. This reduction in simulation execution time is crucial for optimizing training processes, especially in deep reinforcement learning (DRL) applications [3].

The training process will focus on two primary models: FCNN and LSTM. FCNN will be employed to explore how traditional feedforward neural networks perform in predicting time-series data, while LSTM will be used due to its proven ability to handle time-dependent and sequential data, making it particularly suited for time-series prediction tasks. Both models will be trained on the generated simulation data, and their performance will be evaluated based on their ability to predict key features accurately [7].

Once trained, the models will be assessed by comparing their predicted outputs with the ground truth data obtained from the CFD simulations. The accuracy of each model in predicting force coefficients and pressure probes will be analyzed, providing insights into the efficiency of FCNN and LSTM networks for this type of prediction task. Special attention will be given to evaluating the performance in terms of prediction error, and the models will be further fine-tuned through hyperparameter optimization to improve their accuracy [9].

The project will also include an exploration of the impact of different hyperparameters, such as the number of layers, the number of neurons per layer, the learning rate, and the activation functions, on the performance of the models. This optimization process aims to identify the best configuration that minimizes prediction errors and improves the models' generalization capability [12].

The implementation of the project will be carried out using popular machine learning frameworks such as PyTorch. This framework offers the necessary tools to design, train, and evaluate the FCNN and LSTM models, ensuring flexibility in model development and optimization. The data generated from the 34 simulations will be split into training and validation sets to ensure the models generalize well to unseen data, avoiding overfitting while improving predictive accuracy [11].

By the end of the project, a set of recommendations will be provided, focusing on the most effective FCNN and LSTM architectures for time-series prediction in fluid dynamics applications. The recommendations will include insights on how hyperparameters affect prediction accuracy and offer guidance on configuring neural networks to achieve optimal results in similar simulation-driven tasks [5].

In conclusion, this project aims to explore the potential of FCNN and LSTM networks for replacing traditional CFD simulations with more efficient surrogate models. By optimizing these models for time-series prediction tasks, the project seeks to reduce the computational burden associated with training in fluid dynamics, ultimately contributing to the broader goal of improving the efficiency and scalability of simulation-driven tasks in various engineering and scientific domains.

1.1 Objective

The objective of this project is to explore and analyze neural network architectures for the prediction of time-series data generated from simulations. The project begins with a comprehensive literature review to identify the most suitable network types for time-series prediction. Simulations of an actuated 2D flow past a cylinder at varying Reynolds numbers will be performed to generate the necessary training data. The prediction accuracy of force coefficients and pressure probes will be evaluated using FCNN. Additionally, LSTM model will be trained and their prediction accuracies compared to that of the FCNN. The project will also involve an analysis of how different hyperparameters influence prediction errors. Finally, the project aims to provide recommendations for the most appropriate network architectures for time-series prediction, emphasizing the role of hyperparameters in optimizing prediction accuracy.The scope of this project encompasses the evaluation and application of deep neural networks for predicting time-series data in fluid mechanics simulations. It will involve performing detailed simulations of a 2D flow past a cylinder across single Reynolds number to generate datasets used for model training. The project will focus on investigating the effectiveness of FCNN and LSTM in accurately predicting force coefficients and pressure probes. Additionally, the scope includes an analysis of the impact of different hyperparameters on model performance. By comparing multiple architectures, the project aims to determine the optimal approach for prediction tasks and offer recommendations for future implementations in simulation-driven predictions. This research will contribute to the development of more efficient

models for predicting dynamic fluid behaviors with reduced reliance on computationally expensive simulations.

1.2 Motivation

The motivation behind this project stems from the growing need for efficient methods to predict time-series data, particularly in the field of fluid mechanics. Traditional simulation-based training, such as CFD, is time-consuming and computationally expensive. By leveraging DRL and surrogate models, the training process can be significantly expedited, making it more feasible for real-world applications [1]. Neural networks, with their ability to model complex, high-dimensional data, provide a promising solution for replacing some of these simulations. This project aims to explore and optimize neural network architectures to improve the accuracy and efficiency of time-series predictions, which is crucial for advancing research in fluid dynamics and similar areas where real-time predictions are needed but traditional methods fall short [6].

2 Numerical Setup

Accurate aerodynamic modeling is necessary to forecast crucial properties including drag coefficient, lift coefficient, and pressure distribution, all of which are essential for the design and optimization of engineering systems, particularly in the fields of fluid mechanics and aerodynamics. Conventional CFD simulations provide high-fidelity solutions for fluid flow analysis, but they often need significant computer resources, especially when simulating unsteady and nonlinear flow events. These challenges may become even more apparent when working with complex geometries or flows that have high Reynolds numbers. Researchers are now investigating other approaches as a result of this limitation, such as substituting machine learning-based models for CFD simulations. These machine learning techniques enable quick aerodynamic predictions without sacrificing precision, perhaps improving the overall efficacy of the design process [11].

2.1 Neural Networks for Aerodynamic Modeling

Neural networks have garnered significant attention in fluid dynamics research due to their capacity to learn complex, non-linear correlations from large datasets. Unlike traditional methods that require solving the governing equations of fluid dynamics at each time step, neural networks can be trained to approximate the relationship between input parameters (such as velocity and pressure) and output parameters (such as drag and lift coefficients). The need for neural networks arises from the high computational cost and time required by traditional CFD simulations. By offering fast yet accurate machine learning models, neural networks provide a promising approach for surrogate modeling, potentially replacing expensive CFD simulations with more efficient alternatives.

2.1.1 FCNNs

Because FCNNs can simulate intricate interactions between inputs and outputs and approximate nonlinear functions, they are frequently utilized for regression tasks in fluid dynamics [11]. Aerodynamic coefficients and flow field attributes are two examples of steady-state flow issues where FCNNs perform especially well when the input data is spatially structured. FCNNs are useful tools for predicting aerodynamic coefficients from CFD data because they can effectively capture the underlying patterns in the data by learning the mapping from input characteristics to the target outputs.

2.1.2 LSTMs

A particular kind of RNN called an LSTM network is made to process sequential data. They are suitable for time-dependent prediction challenges because they may be used to represent unstable aerodynamic flows in which the condition of the system at a particular moment is dependent on previous inputs [5]. In contrast to conventional RNNs, LSTMs may learn temporal patterns over long periods of time because of their architecture, which stores long-term dependencies in their memory cells. This capability is especially useful for modeling aerodynamic phenomena where past flow states significantly influence future behavior, such as wake dynamics, pressure variations, and vortex shedding. The time-dependent aerodynamic characteristics of flows surrounding oscillating structures, like the forces acting on a vibrating cylinder, are captured in this study using LSTMs.

This study attempts to create an effective surrogate model that can forecast aerodynamic forces acting on an oscillating cylinder by contrasting FCNNs and LSTMs. Achieving great precision while lowering the computational expense usually connected with CFD simulations is the aim. A promising path toward improving real-time flow analysis and facilitating more effective design procedures in engineering applications is the hybridization of machine learning models for aerodynamic forecasts.

2.2 Flow Conditions and Numerical Setup for Oscillating Cylinder Simulation

This work focuses on a canonical benchmark problem in fluid dynamics: the two-dimensional **laminar flow** past an oscillating circular cylinder at a Reynolds number of **100**. This setup has been widely studied in literature and serves as a standard test case for validating computational and machine learning models for aerodynamic predictions[1]. Vortex shedding, periodic wake creation, and changing pressure distributions are the main causes of an oscillating cylinder's unsteady flow characteristics. These phenomena are highly nonlinear and difficult to model since they depend on variables such as the **Re**, **f**, and **A**. Consequently, the analysis of this flow configuration provides a valuable benchmark for assessing the predictive capabilities of machine learning algorithms in capturing aerodynamic features and understanding the force dynamics on bluff bodies.

The Reynolds number, which characterizes the ratio of inertial to viscous forces, is defined as:

$$Re = \frac{UD}{\nu} \tag{2.1}$$

where:

- U is the free-stream velocity,
- D is the cylinder diameter,
- ν is the kinematic viscosity of the fluid.

The aerodynamic forces acting on the cylinder are described using the C_d and C_l, which are given by:

$$C_d = \frac{2F_d}{\rho U^2 D} \tag{2.2}$$

$$C_l = \frac{2F_l}{\rho U^2 D} \tag{2.3}$$

where:

- F_d and F_l represent the drag and lift forces, respectively,
- ρ is the fluid density,
- U is the free-stream velocity,
- D is the cylinder diameter.

By focusing on this well-established benchmark case, this study evaluates the effectiveness of machine learning-based surrogate models in replacing traditional CFD simulations. High-fidelity CFD simulations are performed using OpenFOAM's PimpleFoam solver, with initialization through PotentialFoam, ensuring an accurate and stable numerical setup. The surrogate models trained on this data enable rapid aerodynamic force prediction, supporting applications in real-time flow control and aerodynamic optimization. Future research will aim to extend this work to higher Reynolds number turbulent flows and three-dimensional configurations, further enhancing the applicability of data-driven aerodynamic modeling.

2.2.1 Governing Equations

The Navier-Stokes equations, which describe the motion of an incompressible, viscous fluid, serve as the foundation for the governing equations for the flow around the oscillating cylinder. These equations, which control flow behavior under different circumstances, are essential to fluid dynamics. The following are the equations:

$$\frac{\partial u}{\partial t} + u \cdot \nabla u = -\frac{1}{\rho}\nabla p + \nu \nabla^2 u \tag{2.4}$$

$$\nabla \cdot u = 0 \tag{2.5}$$

where u is the velocity field, p is the pressure, ρ is the fluid density, and ν is the kinematic viscosity. These equations describe the fluid's velocity field and pressure distribution, which are influenced by the oscillating motion of the cylinder, vortex shedding, and the dynamic interaction between the fluid and the structure.

2.2.2 Simulation Domain and Boundary Conditions

For the numerical simulation of the oscillating cylinder flow, a structured computational domain is employed, ensuring that high-resolution grids are used near the cylinder surface and in the wake region. This resolution is necessary to accurately capture the boundary layer and the intricate wake dynamics. The boundary conditions for the CFD simulation are defined as follows:

- **Inlet boundary**: A uniform velocity profile is applied, corresponding to a fixed Reynolds number to simulate steady inflow conditions.
- **Outlet boundary**: A convective boundary condition is used at the outlet to allow smooth exit of the flow without introducing artificial reflections.
- **Cylinder surface**: A no-slip boundary condition is applied, meaning that the velocity at the cylinder surface is zero.
- **Far-field boundaries**: These boundaries are chosen to prevent reflections from the domain edges, ensuring that the flow characteristics remain undisturbed in the computational domain.

2.2.3 Cylinder Motion

The motion of the cylinder is modeled as a transverse oscillation, which is described by the following equation:

$$y(t) = A \sin(2\pi f t) \tag{2.6}$$

where A is the oscillation amplitude, f is the oscillation frequency, and t is time. The oscillating motion of the cylinder introduces dynamic effects that influence the vortex shedding and pressure distribution around the body. By varying the amplitude-to-diameter ratio (A/D) and frequency-to-natural frequency ratio (f/f_n), different flow regimes and aerodynamic forces can be explored.

2.2.4 Mesh Resolution and Temporal Discretization

A high-resolution structured grid is created in order to precisely capture the intricate flow characteristics surrounding the oscillating cylinder. This grid guarantees that wake dynamics and boundary layer effects are adequately handled.

2.3 Setup of Experiment and Simulations

The setup for this study involves a two-dimensional computational domain featuring an oscillating circular cylinder, subjected to controlled actuation via variations in amplitude and frequency. The oscillatory motion of the cylinder generates fluid-structure interactions, leading to vortex shedding, drag fluctuations, and dynamic lift forces. These phenomena are key to understanding the aerodynamic forces acting on the cylinder. To simulate these interactions, CFD simulations are conducted using OpenFOAM (version 2312), an open-source CFD software. The simulations are designed to generate time-series data for key aerodynamic parameters, such as drag and lift coefficients, as well as pressure distributions. The simulations are governed by the incompressible, transient Navier-Stokes equations, which are solved using the *PimpleFoam* solver. This solver is ideal for handling unsteady flows, such as those generated by the oscillating cylinder. The system is initialized using the *PotentialFoam* solver to obtain an initial steady-state velocity field, which ensures smooth convergence and helps prevent numerical instabilities during the transition into the unsteady regime.

The FVM is employed for numerical discretization on a structured computational mesh. A time step of Delta t= 0.01s is used to adhere to the CFL condition, ensuring numerical stability while accurately capturing transient aerodynamic effects.

2.3.1 Computational Domain

The computational domain is a rectangular region in the x-y plane, with the cylinder placed at the center. The dimensions of the domain are as follows:

- **Width** (x-direction): 22 x/d
- **Height** (y-direction): 4.1 y/d

The cylinder has a diameter of 0.1m and is located at coordinates $x = 1x/d$ and $y = 2y/d$. The flow enters from the left side of the domain (inlet) and exits from the right side (outlet). The placement of the cylinder and the flow boundaries are shown in the following diagram.

In this diagram, the cylinder is placed in the center of the computational domain near the inlet as

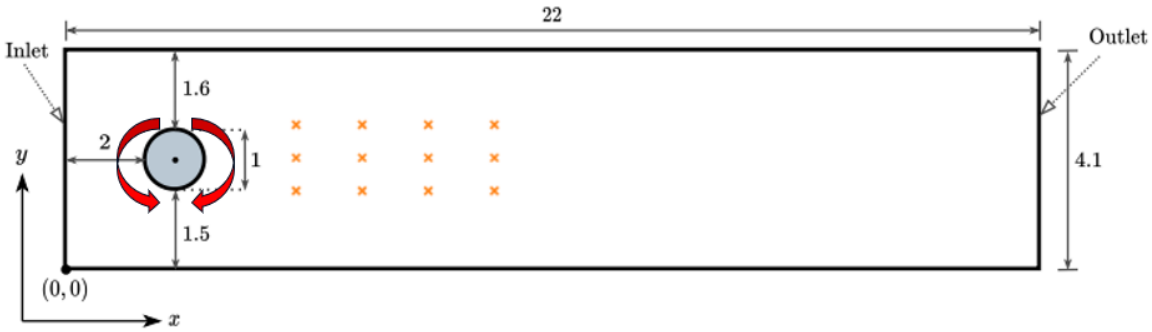

Figure 2.1: Computational domain setup showing the oscillating cylinder and flow boundaries.

per the mentioned dimensions, with the inlet and outlet boundaries clearly marked. The orange crosses placed in te domain are the pressure probes to capture the pressure distribution in the domain. The flow direction is shown, with the fluid entering from the left and exiting from the right. The oscillatory motion of the cylinder induces vortex shedding, which significantly affects the wake region and the aerodynamic forces acting on the cylinder.

2.3.2 Numerical Mesh

A structured mesh is used to discretize the computational domain. The mesh resolution is crucial for accurately capturing the boundary layers and wake structures, which are essential for modeling vortex shedding. The mesh used in the simulations consists of 39,200 cells.

This mesh structure ensures accurate resolution in regions critical to vortex shedding, such as the boundary layer around the cylinder and the wake behind it. The mesh is finer around the cylinder to capture the flow characteristics near the surface, and it becomes coarser further from the cylinder, which helps optimize computational resources.

This figure presents the structured mesh around the cylinder, with a finer mesh near the cylinder to accurately capture the boundary layer effects, while the coarser mesh farther from the cylinder is used to model the larger flow structures in the wake. The mesh resolution is carefully selected to balance capturing vortex shedding dynamics and maintaining computational efficiency.

2.3.3 Flow Conditions and Simulation Parameters

The simulation parameters are carefully selected to replicate realistic oscillatory flow conditions. These parameters are represented as dimensionless numbers to ensure scalability and comparability across different flow conditions. LHS sampling is done for parameter variation to find the optimal amplitude and frequency for the actuation of the cylinder.

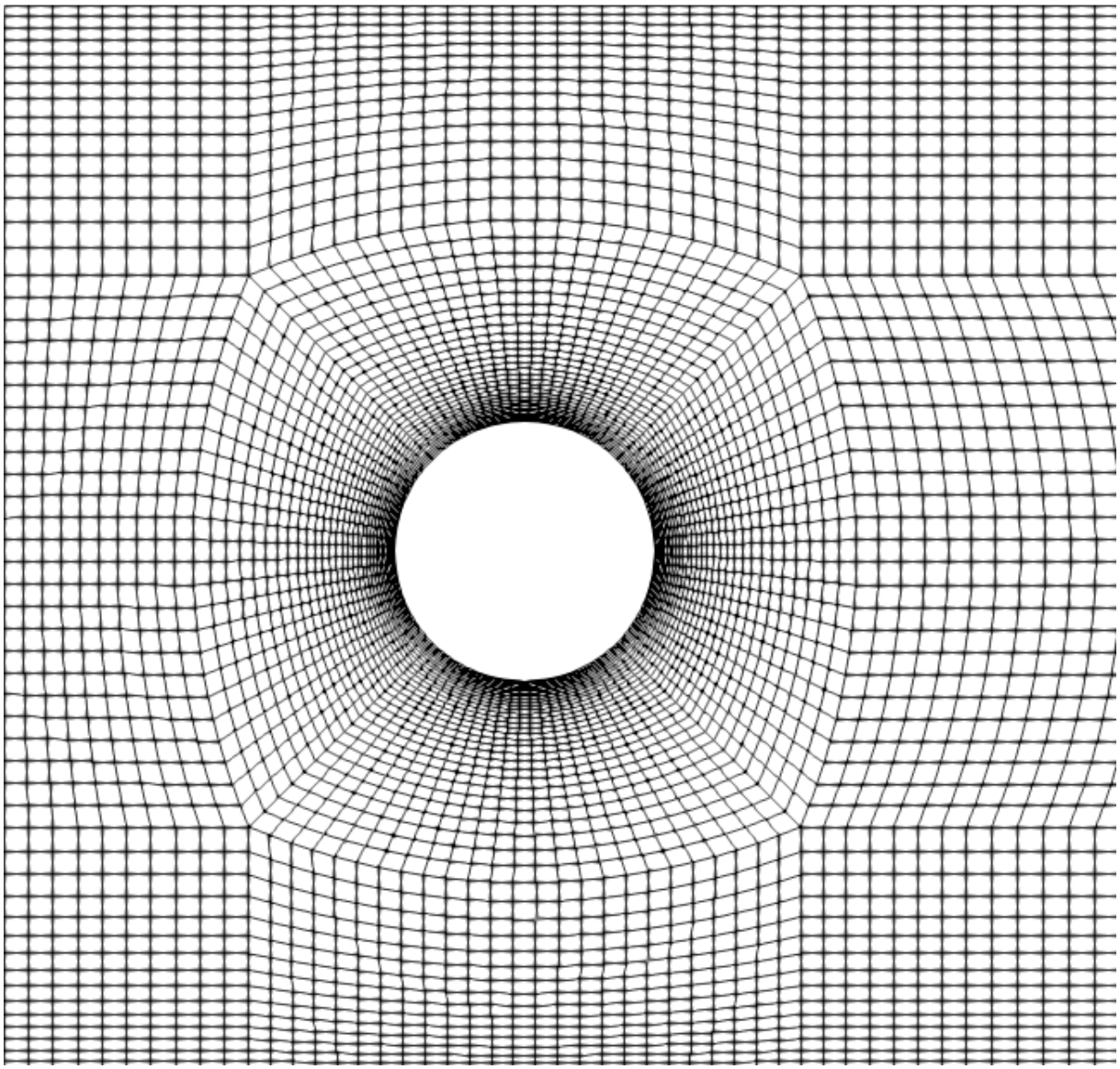

Figure 2.2: Enlarged view of mesh around the oscillating cylinder.

- **CFD Simulations and Parameter Variation and Amplitude-to-Diameter Ratio (A/D)**: A total of 34 CFD simulations are conducted, varying the amplitude-to-diameter ratio (A/D) and the frequency of the cylinder's oscillatory motion. The amplitude-to-diameter ratio, A/D, governs the intensity of the oscillations, with lower values (e.g., $A/D = 0.5$) resulting in smaller oscillations and a relatively stable wake, while higher values (e.g., $A/D = 3.5$) lead to stronger oscillations that significantly influence vortex formation and shedding behavior. The range $A/D \in [0.5, 3.5]$ is selected to capture both low and high-intensity oscillations, providing a comprehensive understanding of how variations in amplitude affect the wake dynamics and vortex shedding.

The actuation parameters, including A/D and frequency, are varied using Latin Hypercube Sampling (LHS), a statistical technique that efficiently samples the parameter space. This method ensures that the simulations encompass a wide range of parameter values, allowing for a detailed analysis of vortex shedding and aerodynamic forces.

- **Dimensionless Frequency (f/f_n)**: The dimensionless frequency is defined as the ratio of the oscillation frequency f to the natural shedding frequency f_n of the cylinder. The chosen frequency range is $f/f_n \in [3.0, 5.0]$, encompassing both subcritical and supercritical vortex shedding regimes. This range effectively captures a variety of flow structures, including shear layer instability, periodic vortex shedding, and flow reattachment phenomena.

By varying both the amplitude and frequency, the simulations cover a wide range of flow behaviors, from laminar to turbulent conditions, ensuring that the surrogate models can generalize across different wake transition regimes.

2.3.4 Surrogate Model Development

Once the time-series data is generated from the CFD simulations, the next step is to develop surrogate models to predict aerodynamic forces more efficiently. FCNNs and LSTM networks are trained on the CFD-generated data to predict aerodynamic forces such as drag and lift, based on the cylinder's oscillatory motion. These machine learning models offer a computationally efficient alternative to running full CFD simulations.

By using these surrogate models, the study aims to reduce the computational cost associated with CFD simulations while retaining high prediction accuracy. The models are trained to predict key parameters based on the time-series data of amplitude and frequency variations, helping to provide real-time predictions for engineering applications involving oscillating cylinders.

2.3.5 Vortex Shedding Visualization

The results of the CFD simulations are visualized to understand the vortex shedding behind the oscillating cylinder. The following images capture the vortex shedding at different stages of the oscillation, highlighting the evolution of the wake structure.

Figure 2.3 shows the velocity profile behind the cylinder at an early stage of oscillation. The image shows the oscillation of the cylinder with certain actuation.

Figure 2.4 shows the wake structures become more complex as the amplitude and frequency of the oscillation increase.

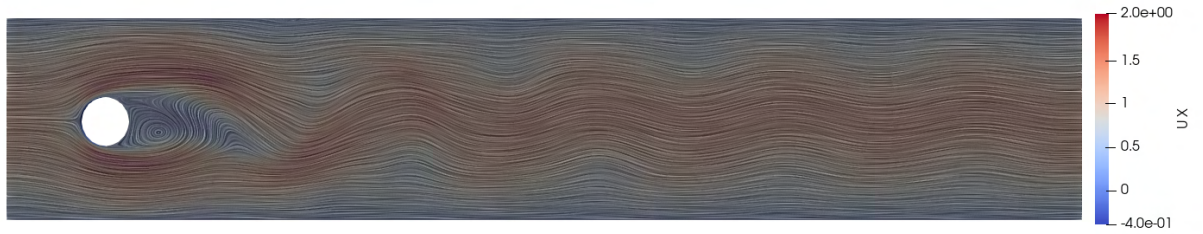

Figure 2.3: Vortex shedding behind the oscillating cylinder (Initial stage).

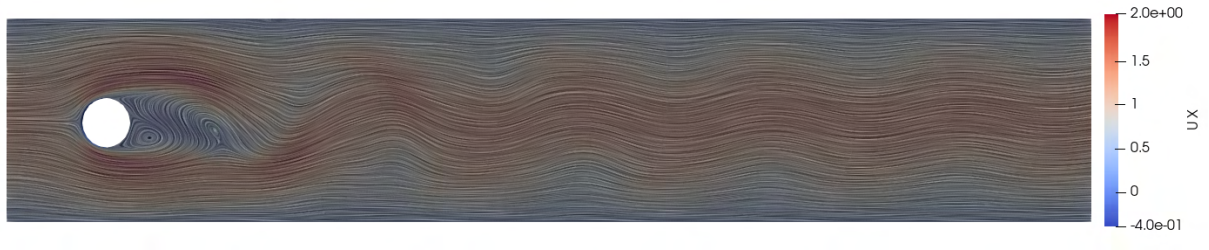

Figure 2.4: Vortex shedding behind the oscillating cylinder (Later stage).

These visualizations provide valuable insight into the fluid dynamics around the oscillating cylinder and help in understanding the effects of oscillation parameters on vortex shedding.

In conclusion, the setup of the CFD simulations with a detailed mesh around the oscillating cylinder is essential for generating accurate time-series data for surrogate model training. The simulations capture key aerodynamic parameters, including drag and lift coefficients, as well as pressure distributions, which are critical for predicting aerodynamic forces. By using machine learning models to replace some CFD simulations, the study aims to provide a computationally efficient method for predicting aerodynamic forces under varying oscillatory conditions. The results from this methodology will contribute to faster and accurate simulations, which can be applied to real-world aerodynamic and engineering problems.

3 Methodology

The proposed methodology aims to develop a computationally efficient surrogate model for predicting aerodynamic forces in fluid dynamics simulations, specifically for scenarios involving an oscillating cylinder. By leveraging deep learning models, particularly FCNN and LSTM networks, this system seeks to replace certain computationally expensive CFD simulations, reducing both time and resource consumption while maintaining high prediction accuracy. The process begins with the setup of a two-dimensional computational domain and mesh, followed by running a series of CFD simulations using OpenFOAM. The collected data is then used to train and optimize neural networks for time-series prediction tasks. This surrogate model will be evaluated for its predictive capabilities and refined for robustness through various optimization techniques. Further extensions of the system are also considered, including adaptations for three-dimensional flows and hybrid model integrations to enhance its application in real-world scenarios.

3.1 Data Preprocessing

Data preprocessing is an essential step in machine learning that helps transform raw data into a suitable format for model training. Proper preprocessing ensures that the neural network can learn efficiently and produce accurate predictions. The key preprocessing steps for aerodynamic force prediction include normalization, splitting the dataset, feature selection, and removing transient phases to focus on the quasi-steady state.

3.1.1 Normalization

Normalization is crucial for scaling the dataset so that all features have comparable ranges. This helps with convergence and stability during training. The most common technique is the Min-Max Normalizer, which transforms the feature values into a fixed range, typically [0, 1]. The Min-Max scaling formula is:

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

Where:

- X is the original value of a feature,
- X_{min} is the minimum value of that feature in the dataset,
- X_{max} is the maximum value of that feature in the dataset,

- X_{scaled} is the normalized value.

Normalization is essential to ensure that all features have a comparable impact on the model, avoiding situations where features with larger scales disproportionately influence the learning process.

However, it's important to consider how the activation function and backpropagation behave when a feature has an extremely large or small value. For example, with the ReLU activation function, if the input is less than 0, the output is 0, and the gradient becomes 0 as well, meaning that no weight updates would occur during backpropagation. This can lead to issues like the "vanishing gradient" problem, where certain features may fail to contribute effectively to the learning process.

3.1.2 Data Splitting

The dataset is divided into training and Validation subsets to evaluate the model's generalization ability. A typical split is 75% for training and 25% for testing. This division helps ensure that the model does not overfit and that its performance is evaluated on unseen data. The training set is used to learn the relationships between inputs (e.g., Force coefficients and pressure distribution) and outputs (e.g., drag and lift coefficients, Pressure distributions), while the test dataset evaluates the model's performance after training.

3.1.3 Feature Selection

Feature selection is a crucial step that identifies the most relevant inputs for predicting the target variables. For aerodynamic force prediction, the selected features include:

- **Drag coefficient and lift coefficient**: These aerodynamic coefficients provide direct insights into the forces acting on the object.
- **Pressure probe distribution**: Measurements from 12 pressure probes capture surface pressure variations, which influence the aerodynamic behavior.
- **Actuation of the cylinder**: The control input applied to the cylinder affects the flow dynamics and, consequently, the aerodynamic forces.

The target variables include the drag coefficient, lift coefficient, and pressure probe distribution, ensuring that the model captures the key aerodynamic characteristics. By focusing on these features, the model improves its predictive accuracy while incorporating relevant aerodynamic influences

3.1.4 Data Augmentation

Data augmentation involves artificially expanding the training dataset by varying parameters such as frequency and amplitude within a defined range. This technique is useful in improving the model's robustness by exposing it to different oscillation conditions. For instance, the amplitude A and frequency f of the oscillating object are varied, helping the model generalize to different fluid dynamics scenarios.

3.1.5 Transient Phase Removal

The trajectory is run over a total duration of 10 seconds with a time step of 0.01 seconds, resulting in 1000 total samples. It is divided into two phases: the transient phase from 0 to 4 seconds, where flow dynamics are unstable and irregular, and the quasi-steady phase from 4 to 10 seconds, where periodic stability is achieved. Since the focus is on modeling quasi-steady-state behavior, only the data from 4 to 10 seconds is considered, while the transient phase is removed. This ensures the model learns and predicts aerodynamic forces based on the stabilized periodic motion, avoiding the influence of initial instabilities.

3.2 Autoregressive Time-Series Modeling with Sequence Sliding Window

An autoregressive time-series model is employed to forecast future aerodynamic responses using historical data. This approach utilizes a **sliding window technique** to capture the temporal dependencies within the dataset. The sliding window ensures that the model focuses on learning from the quasi-steady behavior of the system, effectively excluding transient fluctuations.

In this approach, the model takes an initial set of H values from the true state, forming the first window, and uses this history to predict the subsequent state. The process continues by predicting the next state based on the previous one, sliding the window forward. This sequence-to-sequence mechanism allows the model to leverage past aerodynamic behavior (e.g., drag and lift coefficients) to predict future values, capturing the underlying temporal patterns in the data.

In this framework, **autoregession** ensures that the model learns the temporal patterns from previous aerodynamic behavior and uses that knowledge to predict future states. Both FCNN and LSTM models are trained to predict aerodynamic forces based on the historical data, which is essential for real-time force prediction.

The combination of the sliding window approach and autoregressive modeling allows the surro-

gate model to predict aerodynamic forces with high accuracy, accounting for both the spatial and temporal dynamics of the flow.

3.3 Neural Network Architectures and Training

In this study, an FCNN and LSTM model was developed to predict force coefficients (drag and lift) and pressure distribution in an oscillating cylinder domain. Two neural network architectures, FCNNs and LSTM networks, were employed to compare their effectiveness in capturing the aerodynamic forces. FCNNs are well-suited for extracting spatial features and modeling nonlinear dependencies, while LSTMs excel at capturing temporal variations, making them ideal for handling the dynamic effects of oscillation, such as vortex shedding and phase shifts. The section begins by comparing the characteristics of both architectures to determine their suitability for fluid force prediction. It then details the training process, optimization techniques, and evaluation metrics used to assess each model's accuracy. The primary objective of this study is to evaluate and compare FCNN and LSTM architectures to determine which approach provides the most accurate predictions of aerodynamic forces in an oscillating cylinder system.

3.3.1 FCNNs

- **Spatial Feature Extraction:**

This is accurate given that FCNNs, are among the most straightforward and potent neural network architectures. They have multiple layers and are arranged so that all of the neurons in the layer above are connected to the neurons in the layer below. Because FCNNs describe the functional link between the input features and the output predictions, they are best suited for applications involving spatial feature extraction, which is where spatial data is essential.

In this study we are comparing error rates of force coefficients and pressure probes by optimizing with FCNN. These networks are intended to capture the interaction between the instantaneous aerodynamic quantities and force coefficients are taken at the surface points of the domain and the force coefficients are drag and lift. Since the input data are mapped through one or more hidden layers before being passed through the final output layer, FCNNs are well suited for approximating the nonlinear activation functions between the fluid flow data and the aerodynamic forces and thus chosen for this problem of predicting the aerodynamic forces experienced by an oscillating cylinder.

The key components of FCNNs for spatial feature extraction include:

- **Input Layer**: The input layer includes the preprocessed force coefficients, pressure, and/or

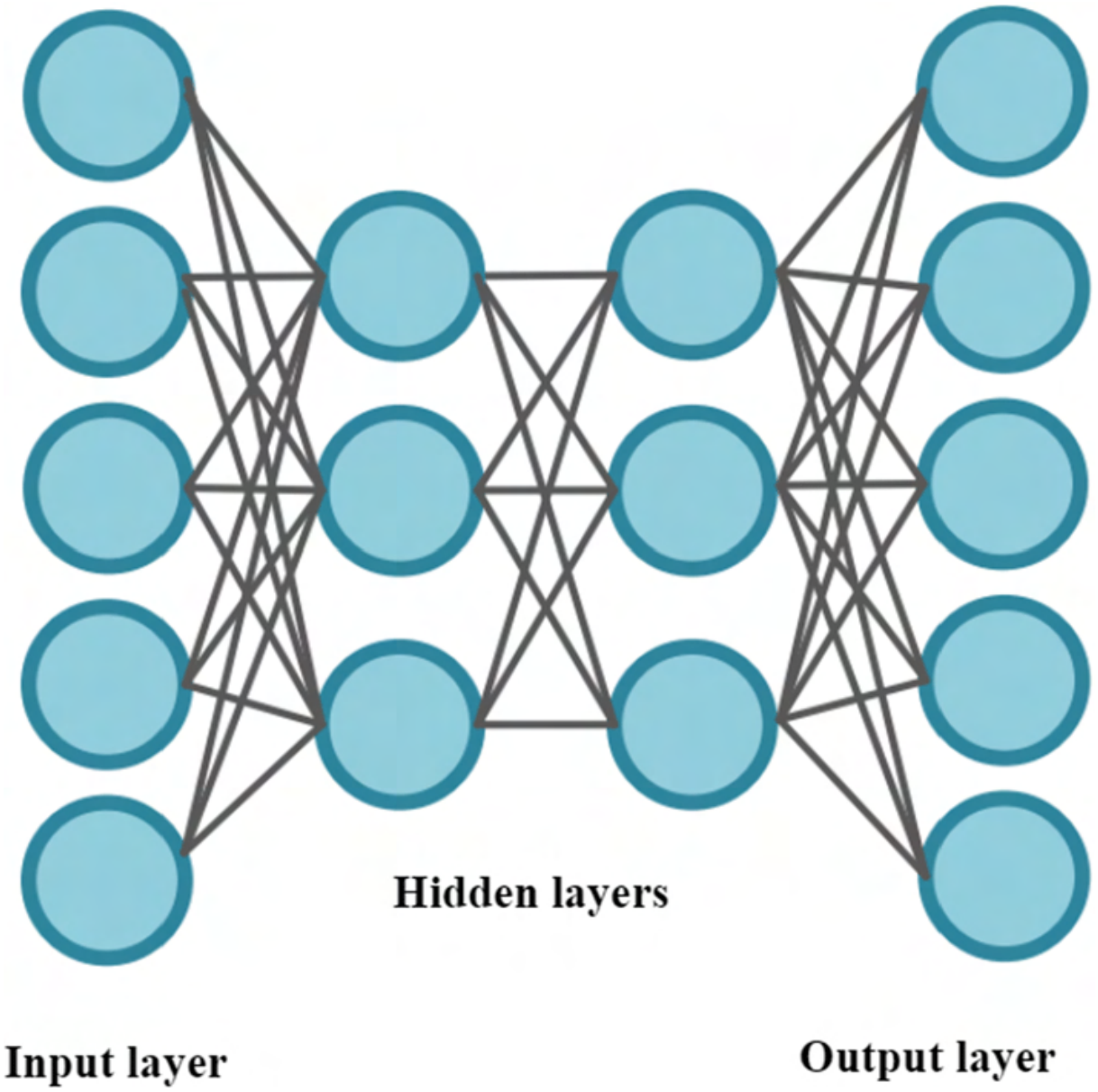

Figure 3.1: Fully connected neural network
https://www.researchgate.net/figure/Fully-Connected-Neural-Network-FCNN_fig5_357871214

other fluid-related values at certain location in the fluid's flow field. These are passed via various layers where the FCNN's architecture determines how interconnected the inputs are.

- **Hidden Layers**: The inputs can be subjected to non-linear transformations and weighted summing by the hidden layers. These layers discover the relationship between the input features and the model's anticipated aerodynamic force output. A non-linear activation function, such as the Rectified Linear Unit and others, frequently permits each hidden layer.

- **Output Layer**: This layer is utilized for the aerodynamic force data, like the drag or lift coefficients, which are the main regions of the simulation's attention. A linear function is the most effective and often used activation function at the output layer, providing a constant value of the feature following processing.

The prediction from the FCNN can be mathematically expressed as:

$$\hat{y} = W^{(out)} a^{(L)} + b^{(out)}$$

where:

- $W^{(out)}$ is the weight matrix for the output layer,
- $a^{(L)}$ is the activation of the last hidden layer,
- $b^{(out)}$ is the bias vector for the output layer,
- $\hat{y}$ is the predicted value (e.g., drag or lift coefficient).
- **Training and Optimization of FCNNs:**

In order to minimize the discrepancy between the intended and actual results, an FCNN is trained by altering the weights and biases among the current neurons. The back propagation approach, which applies the gradient descent technique to the loss function, typically achieves this. MSE, which contrasts the estimated values of the aerodynamic forces with real data, is one of the performance-determination criteria that apply in this situation.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

where:

- y_i is the actual value (drag or lift coefficient) for the i-th sample,
- $\hat{y}_i$ is the predicted value for the i-th sample,
- n is the number of samples in the training set.

Backpropagation is used in the optimization process to find the gradients of the loss function with regard to the bias terms and weights. The weights and bias are then subjected to these gradients in an effort to minimize the MSE.

The hyperparameter must be configured, which includes determining the learning rate, the number of hidden layers, and the number of neurons in each layer to achieve the highest accuracy.

Overall, problems where the spatial arrangement of inputs significantly affects the output can be resolved by FCNNs. In order to accurately anticipate the aerodynamic forces of oscillating flows,

the current work uses FCNNs to match the spatial dependencies between the aerodynamic characteristics and force coefficients. More generalization and assurance of the model's dependability under unknown flow conditions are made possible by back propagation and further hyperparameter optimization.

- **Dense Layers and Activation Functions:**

Dense Layers

One of the most fundamental layers in FCNNs is the dense layer, which links each neuron in the current layer to every other neuron in the layer above. This layer was referred to as "fully connected." Obtaining the weights that serve as metrics that quantify the dependent relationship between the seven input features and the output prediction,in this case, the aerodynamic force coefficients is the primary driving force behind the dense layer.

Prior to being joined together and going through the activation function, the inputs from the preceding layer are weighted and then sent to the neurons in each dense layer. This can be shown as a weighted sum of the formula that follows:

$$z = W \cdot x + b$$

where:

- z is the output of the weighted sum for each neuron in the current layer,
- W is the weight matrix that contains the learnable parameters,
- x is the vector of input values from the previous layer,
- b is the bias term, which allows the model to shift the activation function to better fit the data.

Nonetheless, the outcome of the dense layer is sent to the output layer of a neural network after passing through an activation function.

The ideal method for capturing the interdependencies between variables in the provided data set is to use dense layers. To determine the proper conversion of input data into output, the learned weights and biases in each layer are tuned to lower the loss function.

Activation Functions: ReLU

ReLU is one of the most popular activation functions used with deep learning models like FCNNs. In order to learn or map non-linear relationships between the input and the output, the activation

function is crucial for adding non-linearity to the developed model. Without an activation function, a neural network structure can only model linear relationships, which severely restricts the network's ability to learn from data.

ReLU is a simple but powerful activation function defined as:

$$f(x) = \max(0, x)$$

where x is the weighted sum of the inputs, which is the neuron's input. If the input is positive, ReLU outputs it straight; if not, it produces zero. In essence, it "rectifies" negative values to zero so that only positive values can go through.

Working of ReLU in Training:

The gradients of the loss function with respect to the weights and corresponding biases are calculated during the training phase via back propagation. Weight changes are made from the computed gradients to try to minimize the total loss.

Calculating the derivative of ReLU is easy:

$$f'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

This means that during backpropagation, if the neuron is active (i.e., $x > 0$), the gradient is 1, and if it is inactive (i.e., $x \leq 0$), the gradient is 0. This property helps ReLU activate the appropriate neurons during training, enabling the model to learn the relationships in the data.

ReLU in Action for Predicting Aerodynamic Forces:

ReLU is helpful in describing the relationship between the flow field characteristics (such as pressure and velocity) and the force applied to a cylinder when it comes to force prediction, particularly for aerodynamic forces like drag and lift. In order to capture the aerodynamic forces, the input, such as the pressure and velocity fields, is passed through a number of fully connected layers that incorporate nonlinearity in the form of ReLU activation.

Similarly, by using the ReLU activation function to apply the weighted sums from the preceding layers, the network is able to determine pertinent characteristics and how they interact with one another in the flow to produce accurate drag and lift coefficient estimates. Because ReLU allows

gradients to flow more quickly than any other activation function, it also helps to shorten the time it takes for the training process to converge.

In summary, ReLU plays a critical role in the presence of a link between force coefficients and aerodynamic characteristics. It is found that, in the case of the thick layers, the ReLU parameter helps the model learn and forecast the forces operating on an oscillating cylinder by using the input data.

3.3.2 LSTM Networks

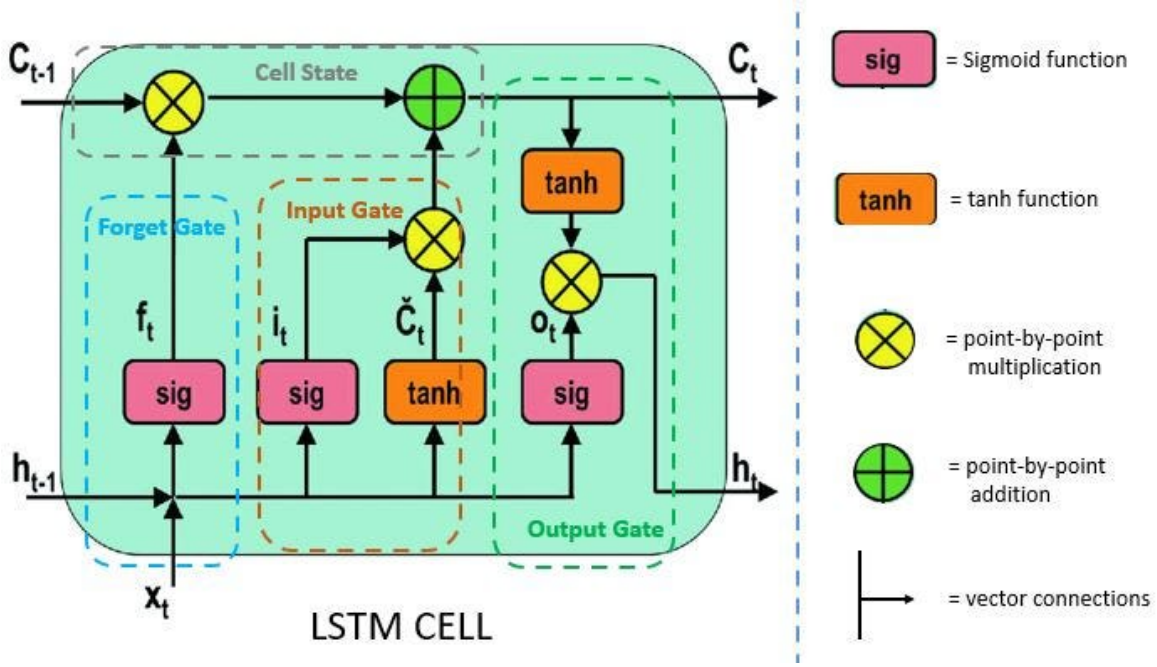

Figure 3.2: LSTM Architecture
https://www.pluralsight.com/resources/blog/guides/introduction-to-lstm-units-in-rnn

- **Temporal Dependencies in Fluid Flow:**

In aerodynamics, especially in scenarios involving unstable or oscillating flows, the state of the flow at any given time is often influenced by its state at previous moments. This is particularly relevant in phenomena such as vortex shedding, where the current fluid behavior is tied to past values, making it a time series problem. To predict future states, such as forces and pressures acting on a body (e.g., drag and lift coefficients) and pressure measurements from probes, accurately capturing these temporal relationships is essential.

For example, in the case of an oscillating cylinder, the conditions of the flow at earlier time steps influence the wake and vortex formation at any given moment. Therefore, it is crucial to account for these time-dependent dynamics to model the flow accurately.

LSTM networks are particularly suited for such tasks due to their ability to capture long-term dependencies and learn from previous time steps. LSTMs excel in following the flow's history and predicting the associated aerodynamic forces, such as lift and drag, as well as the fluctuating pressure from fluid flow oscillations.

In the case of vortex shedding behind a cylinder, the flow pattern is dynamic and changes over time due to variations in the fluid's properties and the cylinder's velocity. The aerodynamic forces at a given time are influenced by these prior interactions. LSTMs can capture these temporal dependencies effectively, enabling more accurate future predictions, particularly in complex and unstable flow situations. Their specialized memory units make LSTMs superior to other models, as they maintain the necessary historical data to make informed predictions.

- **Memory Gates for Retaining Aerodynamic Information:**

The three coupling loops that make up LSTM networks are the entry gate, the exit gate, and the octal gate, which controls the forgetting rate. By regulating how much data should be updated at any given time and how much should be saved in the past, these gates help regulate and manage the LSTM's memory. These gates are crucial for updating the network with past flow condition values and for predicting future aerodynamic forces, which may not be directly related to flow history.

1. Forget Gate: This is the mechanism responsible for the part of the memory to be cleared after some information is processed and or computed. This is especially the case in fluid flow prediction where not all information concerning such flow is always of relevance in the future. For instance, one or several of the flow features may disappear after some period of time due to the changes in the flow regime or vortex shedding frequency. The forget gate outputs a value between 0 and 1, which is then multiplied by the previous memory cell state C_{t-1} to decide which parts of the past memory to retain.

The forget gate's operation is mathematically represented as:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

where:

- f_t is the output of the forget gate,
- σ is the sigmoid activation function,
- W_f is the weight matrix for the forget gate,
- h_{t-1} is the hidden state from the previous time step,
- x_t is the current input,
- b_f is the bias term.

The forget gate's output f_t determines how much of the previous memory cell state C_{t-1} should be passed forward. Specifically, the information in the previous memory state C_{t-1} is multiplied by f_t, where values closer to 0 will cause the network to forget those aspects of the past, and values closer to 1 will allow those aspects to be retained.

2. Input Gate: Additionally, the gate controls how much of the input signal can enter the memory cell and be stored there. However, it also needs the prior hidden layer state and the current input as inputs. Two values are provided by the employees' suggestion app: one for updating the information in memory cell two and another to gauge the extent to which the new information has been adopted.

The input gate is represented as:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

and the candidate memory cell state is given by:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

where:

- i_t is the output of the input gate,
- $\tilde{C}_t$ is the candidate memory cell state,
- W_i, W_C are the weight matrices for the input gate and candidate memory,
- b_i, b_C are the bias terms.

3. Output Gate: The output gate determines the model's outcome or prediction by controlling what is sent out of the network at each time step. Additionally, it predetermines whether portion of the current memory state will be used in the prediction or permitted to move on to the next time step. The output gate aids in illuminating which memory locations will be utilized for both output generation and hidden state calculation.

The output gate is calculated as:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

and the final hidden state is given by:

$$h_t = o_t \cdot \tanh(C_t)$$

where:

- o_t is the output gate's output,

- C_t is the memory cell state at time t,
- h_t is the hidden state that gets passed to the next time step or is used for predictions.

These upper and lower memory gates are in a way that they collect informations on aerodynamics and help to save this information for future use of the predictions. Through the forget gate, LSTMs are able to identify which parts of past flow data are relevant and which new data should be incorporated through the input gate , and also specify how the memory should be used in the prediction process through the output gate, thus making the architecture suitable for flow data with time-varying characteristics. This ability is important for predicting lift and drag coefficients and pressure measurement because the flow evolution is an essential factor for these calculations.

To be more precise, LSTM memory gates enable the storage of valuable aerodynamic data from previous time steps while feeding useless data to the memory dump and then utilizes the stored aerodynamic data to make correct forecasts regarding future aerodynamic behavior. This makes LSTMs well suited in the fluid-structure interaction problems because the history of flow state impacts the future aerodynamic force and pressure.

3.3.3 Model Training and Optimization

- **Backpropagation and Loss Function:**

Now, backpropagation is the most common and well-known algorithm for training the neural networks. The backpropagation step carries out this process by changing the weights used in the model in an attempt to come up with the final weights that give a minimum loss value as determined by the loss function that measures the difference between the true values and the predicted values given by the model. In backpropagation, the gradient of the loss function is derived with respect to its weights, and these gradients are employed to update the weights by the use of the gradient descent.

In regression problems as in the case of force or pressure prediction the errors can be determined via the MSE. It computes the sum of the square of the difference of the predicted and actual values and divides it by the total number of records.

The MSE is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where:

- y_i is the actual value of the i-th sample,
- $\hat{y}_i$ is the predicted value of the i-th sample,

- n is the number of samples in the dataset.

During backpropagation the derivative of the MSE is taken with respect to the weights of the network and these are utilised to change the weights in such a manner that it minimises the difference between the predictions of the model and the actual values.

- **Optimizer: Adam:**

The **Adam** optimizer is a popular choice for training neural networks due to its adaptive learning rate mechanism. It combines the benefits of both **Momentum** and **RMSprop**, adjusting the learning rate for each parameter individually. That is why this concept aids with the overall effort of convergence since the learning rate is adjusted based on gradient details.

- **Hyperparameter Tuning:**

The performance of a neural network can significantly depend on its hyperparameters, including the number of layers, number of neurons per layer, learning rate, and batch size. **Hyperparameter tuning** is the process of selecting the best values for these parameters to improve the model's accuracy and generalization.

- **Number of Layers:** A parameter used in the network architecture is the depth or the number of hidden layers in the network, which determines the capabilities of the network to learn representation in the data. This allows for capture of complex patterns because we end up with deeper layers but comes with the tendency to overfit.
- **Number of Neurons:** It shows that the total amount of neurons present in each layer of the model influences its learning ability to distinguish between intricate attributes. Some limitations are the lack of neurons may result in underfitting while many neurons may cause overfitting.
- **Learning Rate:** The learning rate controls how much the weights are adjusted with each update. A small learning rate may result in slow convergence, while a large learning rate could cause overshooting of the optimal solution.
- **Batch Size:** The batch size determines how many training samples are processed before updating the model's weights. Smaller batch sizes allow for more frequent updates, which can lead to better generalization, but may introduce noise in the gradients.

To find the optimal hyperparameters, tools like **Optuna** can be used for automated hyperparameter optimization. Optuna uses techniques such as Bayesian optimization to explore the hyperparameter space efficiently and find the best combination of values.

Hyperparameters for FCNN and LSTM Models

Hyperparameter	FCNN	LSTM
Number of Layers	3	3
Number of Neurons per Layer	254	242
Learning Rate	0.000119	0.000104
Batch Size	43	46

Table 3.1: Best hyperparameter values for FCNN and LSTM models after tuning.

- **Evaluation Metrics:**

After training the model, it is crucial to evaluate its performance using appropriate metrics. In regression tasks, the two most common evaluation metrics are the **MSE** and **MAE**, which both provide insights into the model's accuracy in predicting continuous values.

Mean Squared Error (MSE)

- **Mean Squared Error:**

The **Mean Squared Error (MSE)** evaluates how well the model's predictions match the true values. It penalizes large errors more than smaller ones, as it squares the difference between predicted and actual values. This makes MSE sensitive to large deviations and is especially useful when you want to penalize larger errors more heavily.

A lower MSE indicates better predictive accuracy, as it reflects smaller deviations from the actual values.

- **Mean Absolute Error:**

The **Mean Absolute Error (MAE)** calculates the average of the absolute differences between the true values and the predicted values. Unlike MSE, MAE does not penalize large errors as severely, providing a more direct interpretation of the model's error in the same units as the input data.

The MAE is given by:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

where:

- y_i is the actual value for the i-th sample,
- $\hat{y}_i$ is the predicted value for the i-th sample,

- n is the number of samples in the dataset.

MAE is useful when you want a balanced assessment of the average prediction error, without disproportionately punishing large mistakes.

- **Temporal Consistency with CFD Data:**

When performing aerodynamic simulations and in particular when it comes to fluid-structure interaction problems, it is equally desirable that the proposed model is time accurate to some Computational Fluid Dynamics. Temporal consistency is measured in how well the predicted forces (like drag and lift) as well as the pressure readings mimic the time-evolving pattern as observed in CFD.

In order to compare the model's output with CFD over time and gauge temporal consistency, the results generated are compared with CFD data at different time instances. This guarantees that the concerned forces and pressure correspond to the correct pattern and reflect the changing and dynamic nature of the vortices and other phenomena in the flow. Temporal consistency can be measured using the correlation coefficient R between the predicted and CFD-derived values over time, given by:

$$R = \frac{\sum_t (X_t - \bar{X})(Y_t - \bar{Y})}{\sqrt{\sum_t (X_t - \bar{X})^2}\sqrt{\sum_t (Y_t - \bar{Y})^2}} \tag{3.1}$$

where X_t and Y_t are the model-predicted and CFD values at time t, and $\bar{X}$ and $\bar{Y}$ are their respective mean values. A high correlation coefficient close to 1 indicates strong temporal consistency between the model's predictions and CFD results.

Sensitivity to time history is important in order to make accurate predictions in conditions where the forces are varying over time due to vortices and such like. Comparing with CFD data, it was seen that the temporal characteristics of the model are valid and the results obtained from the model are satisfactory regarding aerodynamics design and analysis.

4 Results and conclusions

4.1 Results

This section is the next part of the research paper where the findings from the simulations and surrogate model will be shown. In this paper, it will study the prediction of the aerodynamic force by the model and evaluate it against the available CFD simulation data in order to test the feasibility of the model in real environment.

4.1.1 FCNN Results

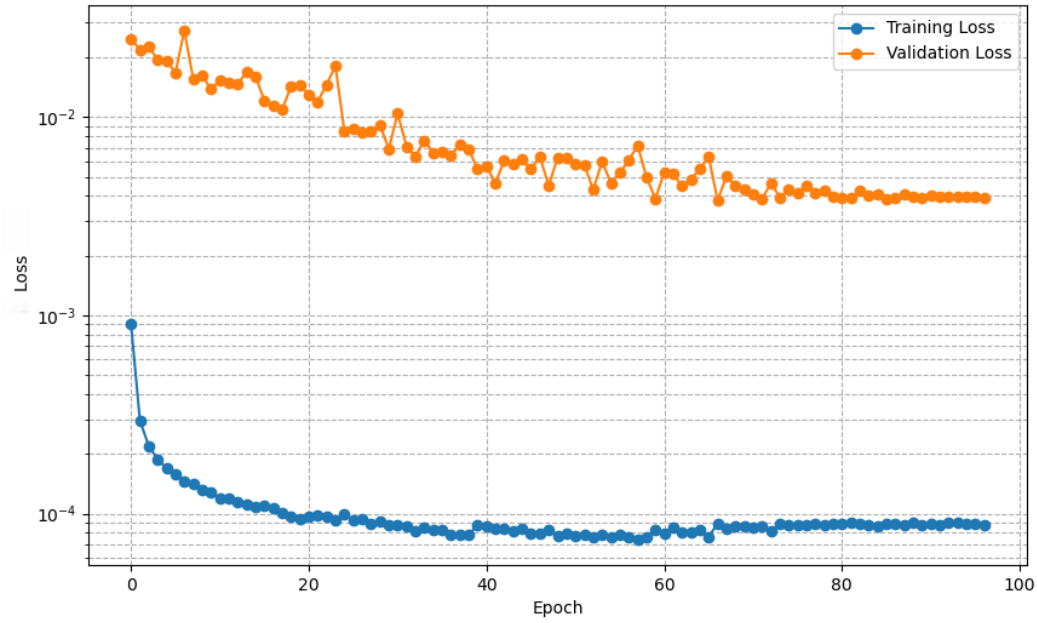

Figure 4.1: Training and Validation loss plot

In the result section, the curves of the training and validation loss for the neural network model over 100 epochs are provided. This is depicted in the plot presented in Figure 4.1, which shows the loss during both the training and validation phases. After the initial few epochs, the training loss begins to decrease significantly, indicating that the model is gradually learning from the training data. However, the validation loss exhibits some instability in certain epochs, suggesting that while the model generalizes well in the early epochs, it starts to overfit to the training data in the later epochs.

Figure 4.2 illustrates the scatter plot of the scaled L_2 loss, which measures the difference between predicted and true pressure values from various probes. The plot shows the L_2 loss for different probes: the aerodynamic coefficients C_d and C_l along with several pressure probes (denoted as p_0 through p_{11}).

The L_2 loss is calculated using the formula:

$$L_2 = \frac{\sqrt{\sum_{i=1}^{n}(p_{\text{true}} - p_{\text{pred}})^2}}{\sqrt{\sum_{i=1}^{n}(p_{\text{true}})^2}} \tag{4.1}$$

where p_{true} represents the true pressure values and p_{pred} represents the predicted pressure values for each probe. The plot clearly shows that the L_2 loss for aerodynamic coefficients C_d and C_l is near zero, indicating that the model's predictions are accurate for these parameters. However, as the probes move from p_0 to p_{11}, the scaled L_2 loss increases, indicating a larger discrepancy between predicted and true values for pressure readings in those probes.

From this it can be inferred that the model holds excellent for the aerodynamic coefficients however the error in pressure prediction tends to increase the farther from the cylinder one goes. As the probed number increases, the loss gradually increases and the figure also highlights the points that need enhancement of the model's potential in prediction.

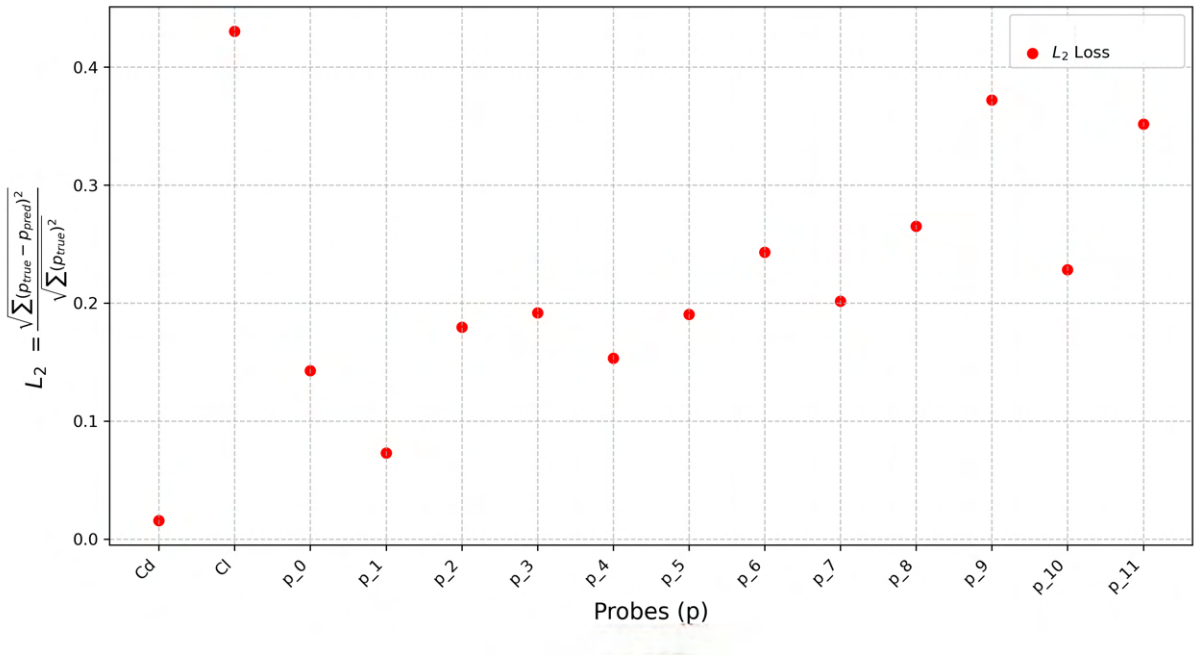

Figure 4.2: L2 loss scatter plot

Figure 4.3 displays the prediction performance with error bars for various probes, including the aerodynamic coefficients C_d and C_l as well as pressure probes from p_0 to p_{11}. Each point on the plot represents the average prediction accuracy for a particular probe, with the error bars reflecting the variation or uncertainty in the results. The average accuracy is computed across the multiple predictions or samples for each probe, providing an overview of the model's consistency and reliability in predicting aerodynamic parameters.

The prediction for C_d and C_l is notably high, around 0.8, suggesting that the model performs well for these aerodynamic coefficients. However, for the pressure probes p_0 to p_{11}, the performance shows a range of values from 0.6 to 1.0, with higher accuracies but also increased error bars. This variability indicates that the model's performance is generally consistent, although with some uncertainty for the probes. The error bars represent the variability in the model's predictions, with thinner bars indicating more stable predictions and higher confidence in the results.

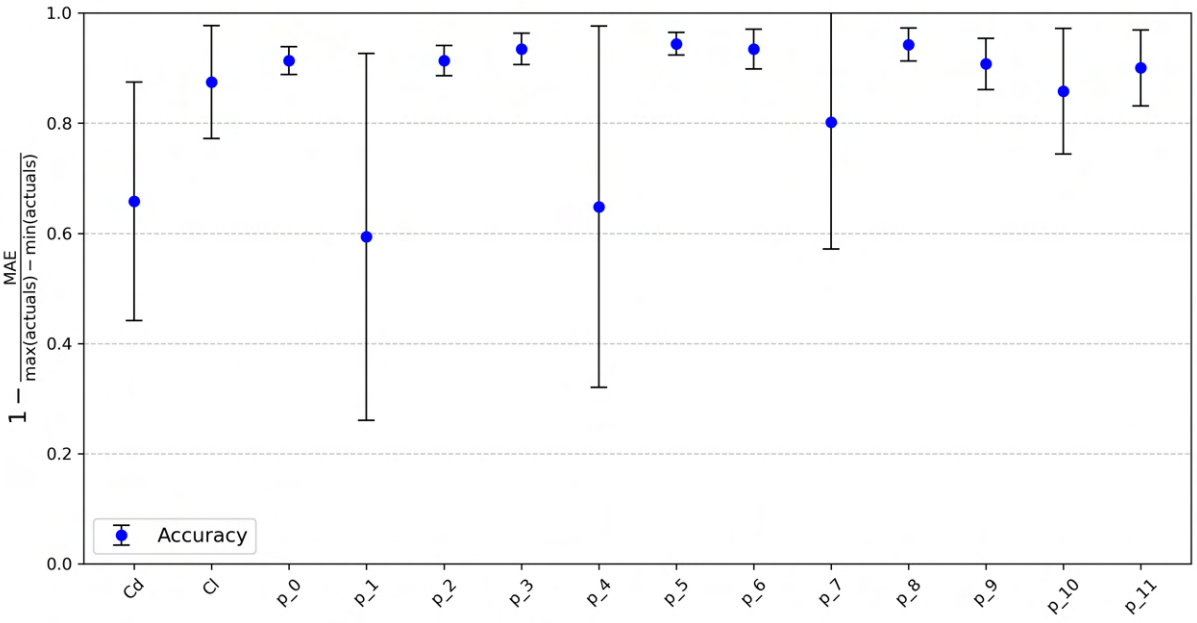

Figure 4.3: prediction accuracy over multiple trajectories

Figure 4.4 shows the comparison between the true values (labels) and predicted values for each pressure probe (from p_0 to p_{11}) across different samples. The blue lines represent the true values (labels), and the orange dashed lines represent the predicted values from the model. Each subplot corresponds to a specific pressure probe, and the x-axis represents the number of samples, while the y-axis represents the measured values.

In most of the subplots, we observe that the model's predictions closely follow the true values, which is an indication of a good fit between the predicted and actual pressure readings. However,

in some cases, small deviations can be seen, especially in the regions with higher fluctuations. These discrepancies highlight areas where the model may still have some room for improvement in terms of capturing the complex dynamics of the pressure variations.

The plots for probes like p_0, p_2, p_5, p_7 show a relatively tight agreement between labels and predictions, whereas probes like p_1, p_6, p_10 exhibit slightly larger gaps, suggesting that the model may not perfectly capture the fluctuations in certain parts of the data. Despite these small discrepancies, the overall pattern indicates that the model is generally effective at predicting the time-series data for all probes.

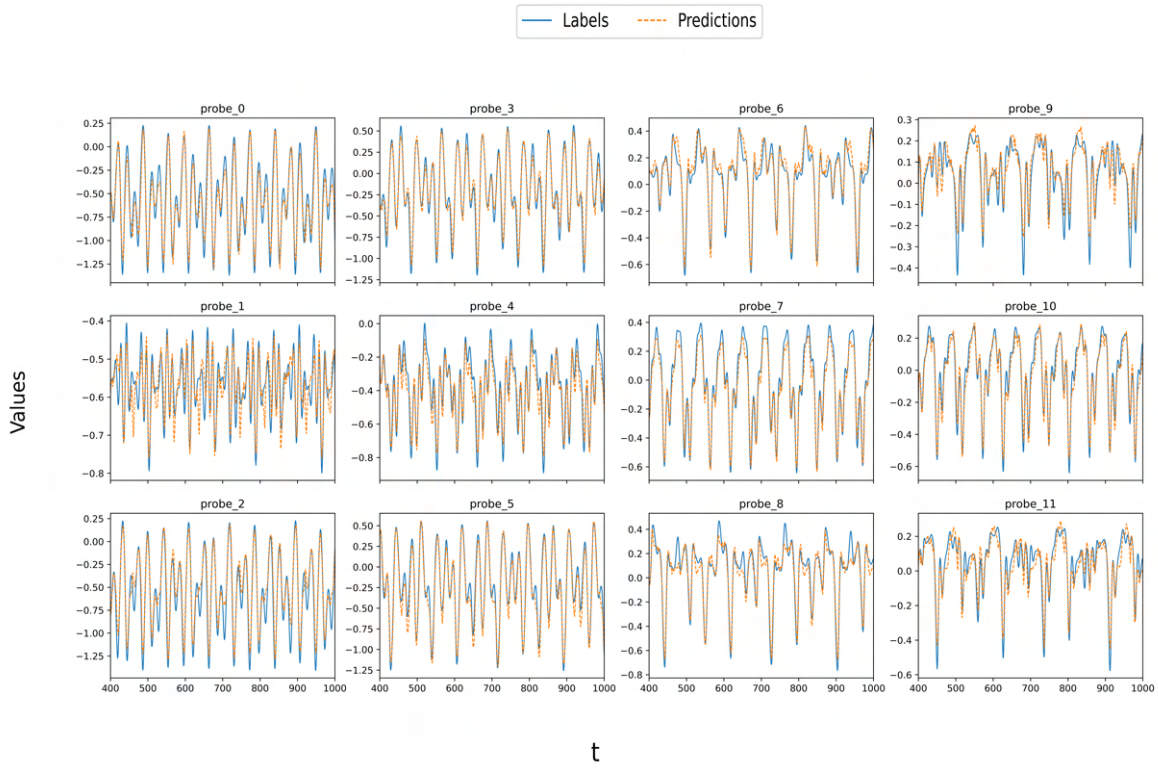

Figure 4.4: Comparision of labels and predictions plot for probes

Figure 4.5 shows the comparison between the true values (labels) and predicted values for the C_d and C_l over a range of samples. The blue lines represent the true values (labels), and the orange dashed lines represent the predicted values.

On the left side of the plot, the C_d values exhibit a relatively close agreement between the true and predicted values. The model captures the major fluctuations and trends in the drag coefficient, though there are minor deviations in certain regions, indicating the model's ability to predict the dynamic behavior of the drag force.

On the right side, the C_l values show similar patterns of agreement between the true and predicted

values. The model's predictions for the lift coefficient follow the actual values closely, with only a few areas where the predictions slightly deviate from the true values, demonstrating that the model is able to handle the fluctuations in the lift force effectively.

Overall, the results indicate that the model performs well in predicting both C_d and C_l, capturing their time-varying behavior with relatively small discrepancies.

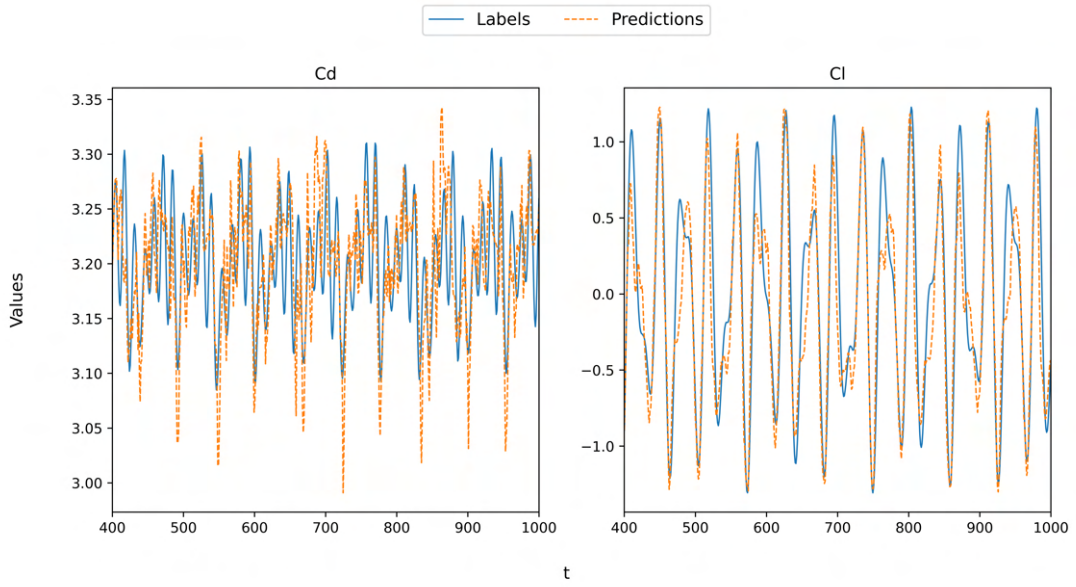

Figure 4.5: Comparision of Labels and predictions for Cd and Cl

Figure 4.6 presents the contour plots of C_d and C_l against f and A. The plots visualize the relationship between the amplitude, frequency, and the prediction performance achieved by the model. The colors in the contour plots represent prediction, with warmer colors indicating higher prediction and cooler colors indicating lower prediction.

On the left, the C_d contour plot shows that higher prediction is achieved in regions where A is around 1.5 and f is in the range of 4.0 to 4.5, as indicated by the darker, warmer colors. The prediction decreases towards the top-right corner, where higher f and A values lead to a reduction in performance.

On the right, the C_l contour plot exhibits a similar trend, where higher prediction is achieved around a frequency of 4.0 and an amplitude of 1.5. Both plots highlight regions with the highest prediction, as well as the influence of f and A on the model's performance.

Data points are overlaid as red crosses, indicating the specific locations sampled for training the model.

Figure 4.7 presents the contour plots of prediction for each pressure probe (p_{probe_0} to $p_{\text{probe}_{11}}$) against f and A. Each plot visualizes how the prediction varies based on these two parameters,

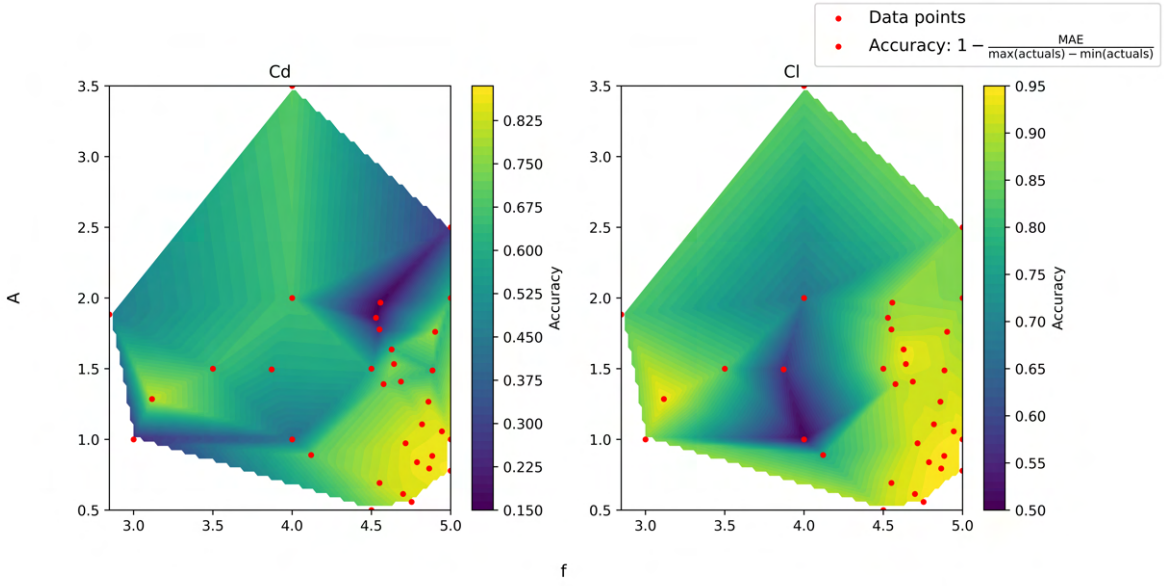

Figure 4.6: Contour plot of Cd and Cl against A and f prediction performance

where warmer colors indicate higher prediction and cooler colors show lower prediction.

Across all plots, we observe that the highest prediction is found in regions with moderate to high f values (around 3.5 to 4.5) and A values around 1.5 to 2.0. This suggests that the model performs best within this f-A range for most probes. For some probes, such as p_{probe_3} and p_{probe_6}, the prediction is higher in specific regions of the f-A space, with data points highlighted by red crosses, representing the sampled locations used during model training.

The varying prediction levels across different probes and the consistent pattern of better performance at mid-frequency and amplitude levels provide valuable insights into how the model's predictions are influenced by different parameters for each probe.

4.1.2 LSTM Results

Figure 4.8 presents a plot comparing the training and validation losses over epochs, displayed on a logarithmic scale. The training loss (in blue) rapidly decreases during the initial epochs, indicating that the model is quickly learning and fitting the training data. After the initial drop, the training loss stabilizes and continues to decrease at a much slower rate, reflecting the model's approach to convergence.

In contrast, the validation loss (in orange) starts at a higher value and then decreases more slowly, eventually stabilizing as well. The gap between the training and validation losses narrows after the initial phase, suggesting that the model is generalizing well, although there is some indication of

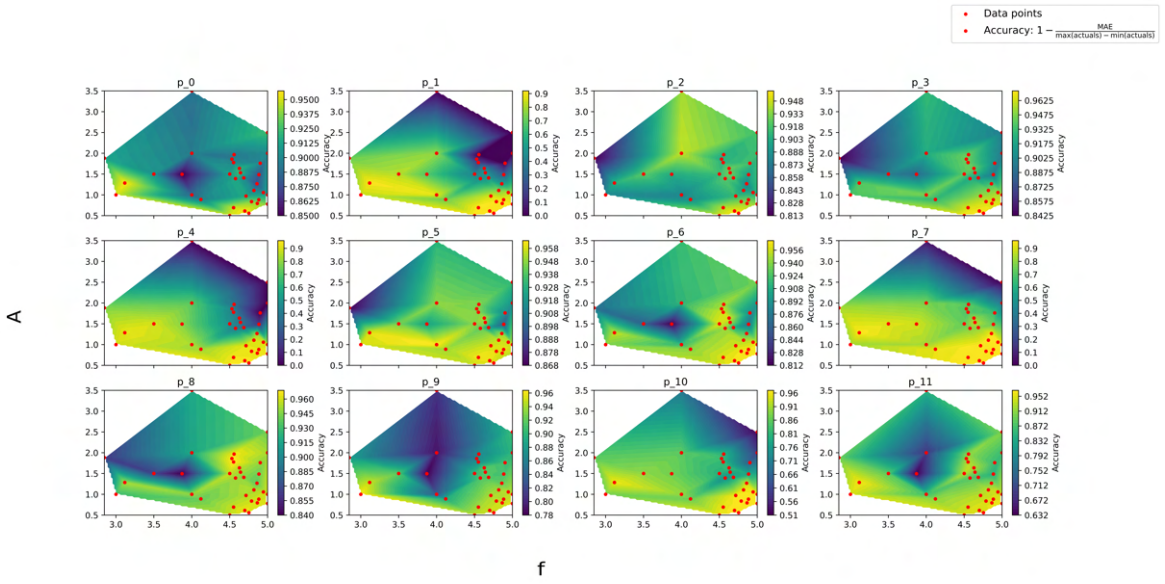

Figure 4.7: Contour plot of prediction performance of probes against A and f

overfitting as the validation loss plateaus while the training loss continues to decrease slightly. This behavior is typical in training deep learning models, where the model learns rapidly at first and then improves at a diminishing rate.

This plot provides insights into the model's training process, with both training and validation losses showing significant improvement, but also highlighting the challenge of balancing model fit and generalization.

Figure 4.9 presents a scatter plot of the L_2 loss, which is a measure of the prediction error for each probe. The x-axis represents the probes labeled from C_d to p_{11}, while the y-axis shows the L_2 loss. The L_2 loss is calculated as the square root of the sum of the squared differences between the predicted and actual values, normalized by the sum of the squared actual values.

From the plot, we observe that the scaled L_2 loss is lowest for C_d and C_l, indicating that the model performs well in predicting drag and lift coefficients. As we move towards the pressure probes, the loss increases, with probes such as p_0 and p_1 exhibiting slightly higher scaled losses. This suggests that the model's accuracy decreases when predicting the pressure values compared to the aerodynamic coefficients.

This plot provides a clear comparison of the model's predictive performance across different probes, helping to identify where the model's predictions are more accurate and where improvements may be needed.

Figure 4.9 depicts the column-wise prediction accuracy of the LSTM model for various probes. The x-axis represents the probes from C_d to p_{11}, while the y-axis indicates the accuracy of predictions.

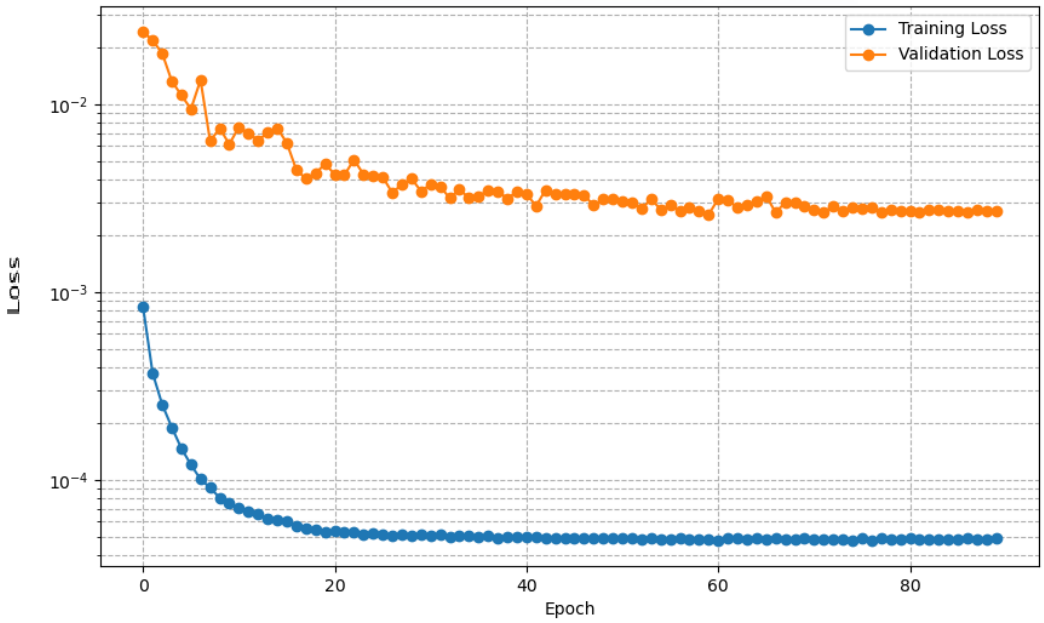

Figure 4.8: Training and Validation loss plot

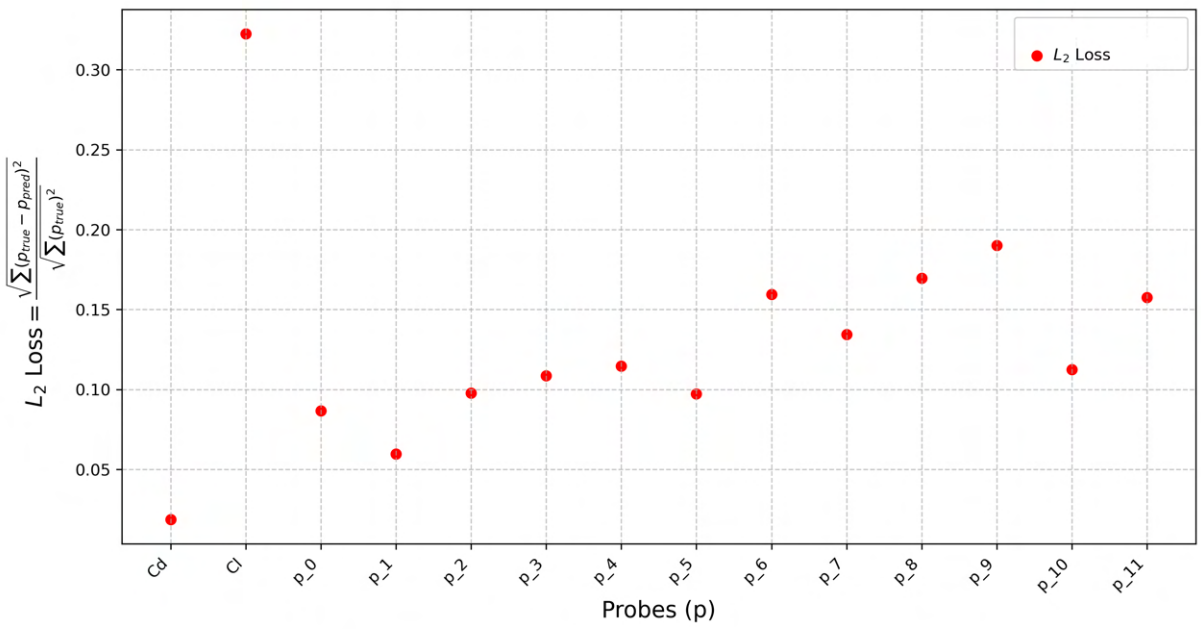

Figure 4.9: L2 loss scatter plot

The blue points denote the accuracy of the predictions for each probe, and the error bars represent the variability or uncertainty in the accuracy of the model's predictions.

From the plot, we observe that the accuracy for C_d and C_l is notably higher, with accuracy values close to 1, indicating strong predictive performance for these coefficients. However, as we move to the pressure probes (e.g., p_0 through p_{11}), the accuracy tends to decrease, especially for probes like p_1, with larger error bars. This suggests that the model struggles more with predicting pressure readings compared to the aerodynamic coefficients.

The error bars highlight the variability in accuracy, where some probes exhibit larger fluctuations, indicating less stable predictions. This analysis provides insights into the model's performance across different data types, showing that while it performs well for aerodynamic coefficients, there is room for improvement when predicting pressure readings.

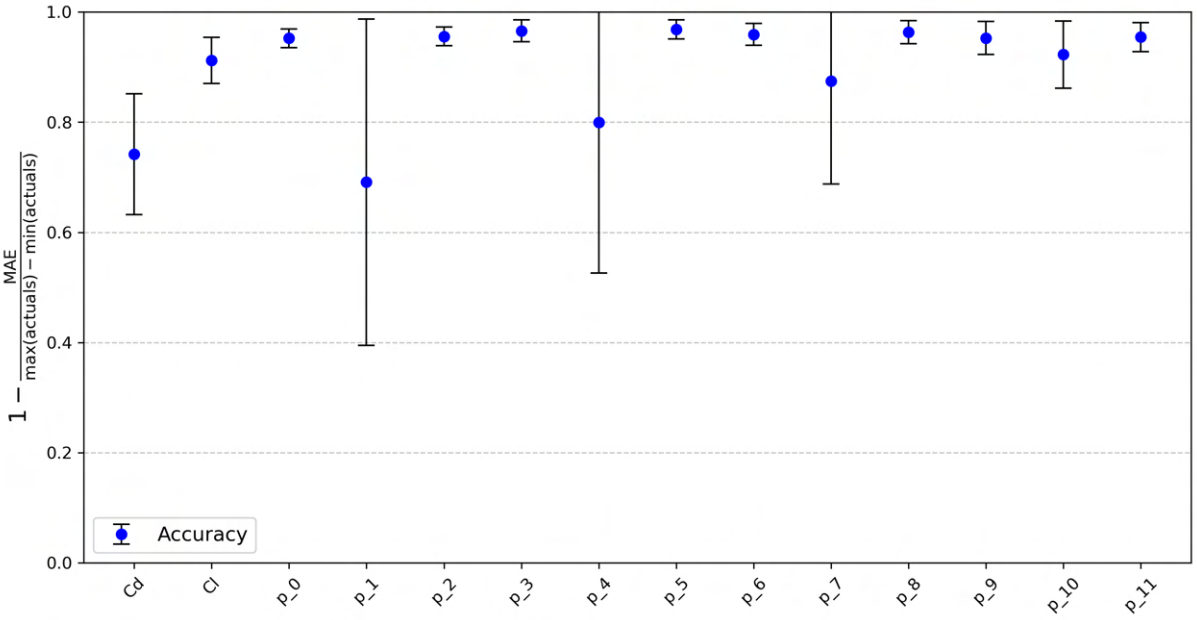

Figure 4.10: Prediction accuracy over multiple trajectories

Figure 4.10 shows the time-series predictions for each probe, including both the ground truth (labels) and the model's predictions. The x-axis represents the number of samples, while the y-axis shows the values for each probe. The blue lines represent the true values (labels), and the orange dashed lines show the predicted values for each corresponding probe.

From the plot, we observe that the model performs well in capturing the oscillatory behavior of the probes. For most probes, including $p_0, p_1, p_3, p_4, \ldots$, the predicted values closely follow the true values, indicating that the model accurately tracks the dynamic changes in the data. However, slight deviations can be seen, especially for probes like p_5, p_8, p_10, where the predictions lag slightly behind or slightly overshoot the true values at certain points.

These results suggest that the model has learned the general oscillatory pattern of the data but may require further tuning or model refinement to minimize the discrepancies in some of the probes.

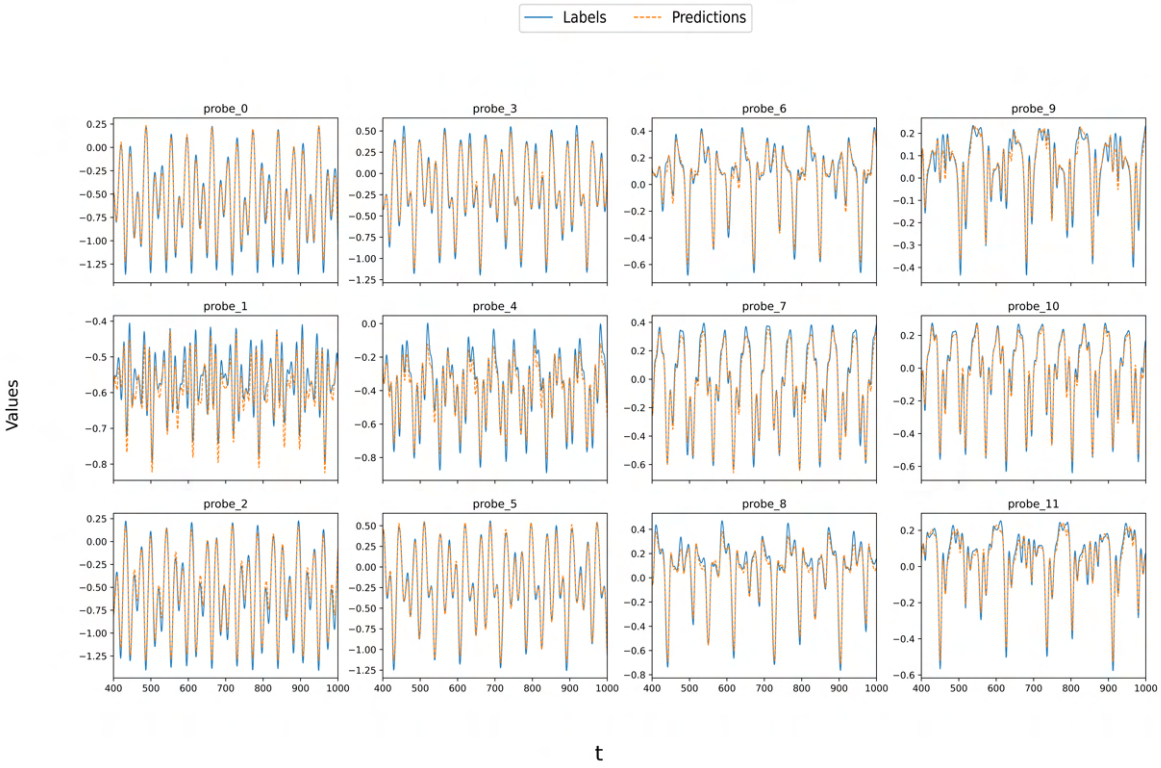

Figure 4.11: Comparision of Labels and Predictions plot for probes

Figure 4.11 presents the time-series predictions for C_d (drag coefficient) and C_l (lift coefficient), where the blue lines represent the true values (labels) and the orange dashed lines represent the model's predicted values. The x-axis shows the number of samples, while the y-axis displays the values for both C_d and C_l.

For the C_d plot (on the left), we observe that the predictions follow the true values closely, with minor discrepancies at specific peaks and troughs. The model is capable of capturing the general oscillatory behavior of the drag coefficient, though there are occasional moments where the predicted values slightly deviate from the actual data.

Similarly, in the C_l plot (on the right), the model shows strong alignment with the true lift coefficient values. However, there are regions where the predictions show some lag or overfitting, particularly in the more extreme values of C_l.

In conclusion, the model demonstrates an overall good fit to the time-series data for both drag and lift coefficients, with room for refinement in capturing more extreme fluctuations.

Cd Contour The plot on the left (Fig. 4.13) indicates that higher accuracy is observed at lower fre-

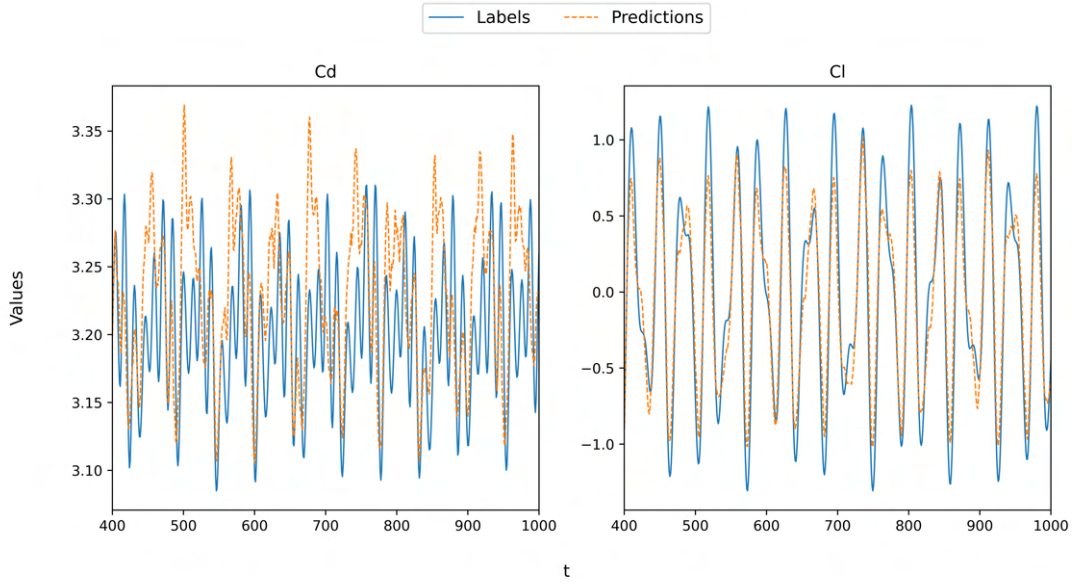

Figure 4.12: Comparision of labels and predictions for Cd and Cl

quencies (around 3.7-4.0) and higher amplitudes (near 2.0). The red points represent specific data points that align well with the areas of higher accuracy. The accuracy ranges from approximately 0.63 to 0.88, with the highest accuracies occurring at the lower frequencies and higher amplitudes. This suggests that the model performs particularly well for these frequency-amplitude combinations, implying that these features have a stronger influence on the accuracy of the Cd prediction.

Cl Contour The plot on the right (Fig. 4.13) shows the relationship for the Lift coefficient (Cl). Similar to the Cd plot, accuracy increases with lower frequencies around 4.0 and higher amplitudes close to 2.0. The accuracy ranges from 0.82 to about 0.96, again with higher values for certain frequency-amplitude combinations. The red points indicate specific instances of data that align with higher accuracy values, showing the optimal regions for predicting Cl.

These contour plots provide valuable insights into how the model's performance varies based on the frequency and amplitude of the input data and help identify the optimal parameter ranges for accurate predictions. They also suggest that there is a clear frequency-amplitude dependency in the prediction accuracy, which could be leveraged to enhance the model's performance by focusing on these regions.

The contour plots illustrate the model's prediction performance across various probes. Each plot represents a specific probe, such as p_{probe_0}, p_{probe_1}, and continuing up to $p_{\text{probe}_{11}}$. These plots are useful for identifying the frequency and amplitude combinations that result in the best prediction for each probe.

For the majority of probes, the highest prediction performance is concentrated in the regions of

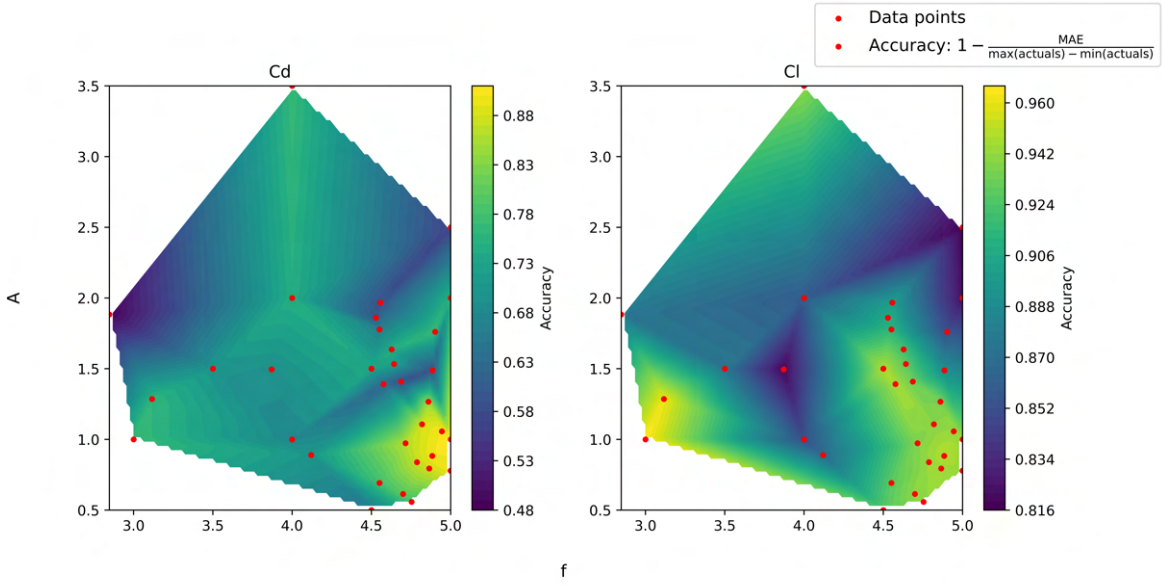

Figure 4.13: Contour plot of Cd and Cl agianst A and f prediction performance

lower f (around 3.5 to 4.5) and higher A (around 1.5 to 2.0). These frequency-amplitude ranges consistently show good model performance, as reflected in the color gradients, which represent high prediction values (ranging from approximately 0.85 to 0.98). Red dots on the plots mark specific data points used in the analysis, which are typically located in areas of high prediction performance, supporting the model's accuracy.

Interpretation of Contour for Each Probe

- **Probes p_{probe_0} to p_{probe_3}:** These plots show relatively high accuracy values, mostly above 0.9, in regions with frequencies around 4.0 and amplitudes closer to 1.8. The accuracy decreases slightly as we move towards higher frequencies and lower amplitudes.
- **Probes p_{probe_4} to p_{probe_7}:** The accuracy values remain high, peaking in the same frequency-amplitude regions, with accuracy values near 0.95. However, probes p_{probe_6} and p_{probe_7} show a decrease in accuracy at higher frequencies (near 5.0), suggesting that the model struggles with these data points.
- **Probes p_{probe_8} to $p_{\text{probe}_{11}}$:** These plots show accuracy values reaching as high as 0.97, with similar patterns where higher accuracy is achieved at lower frequencies and higher amplitudes. Probes p_{probe_9} and $p_{\text{probe}_{11}}$ maintain relatively stable accuracy, reinforcing the model's performance in these regions.

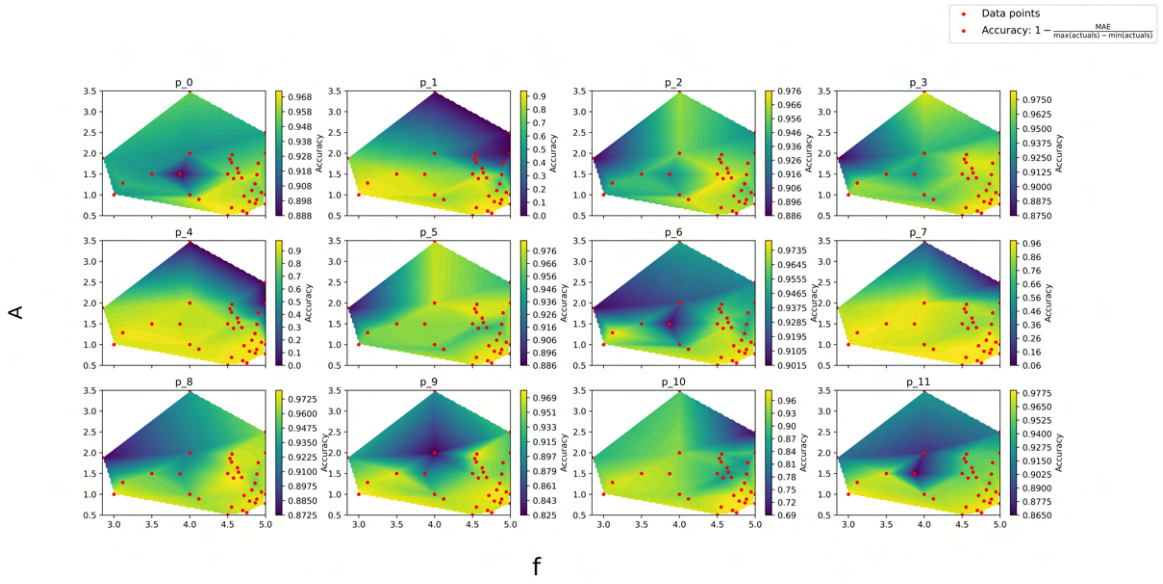

Figure 4.14: Contour plot of prediction performance of probes against A and f

4.2 Conclusion

In this study, both Fully Connected Neural Networks (FCNNs) and Long Short-Term Memory (LSTM) networks were employed to predict aerodynamic forces, specifically drag and lift coefficients. The FCNN model achieved a prediction accuracy of 95 percent, while the LSTM model reached 98 percent, highlighting its superior performance in capturing temporal dependencies inherent in aerodynamic data. These results were obtained after extensive hyperparameter tuning, ensuring optimal network configurations and improved predictive accuracy. The tuning process involved adjusting the number of layers, neurons per layer, learning rate, and batch size to minimize loss and enhance generalization. The LSTM's ability to retain memory of past states allows it to capture complex time-dependent aerodynamic interactions more effectively than the FCNN, making it particularly suitable for dynamic flow conditions. The results suggest that LSTM models are well-suited for real-time aerodynamic force prediction, where capturing transient effects and periodic flow variations is crucial.

These findings align with existing literature comparing FCNN and LSTM architectures in Computational Fluid Dynamics (CFD) applications. For instance, a study on unsteady CFD simulations evaluated various neural network architectures, including autoencoders, UNet, and ConvLSTM-UNet. The ConvLSTM-UNet, which integrates convolutional operations with LSTM units, consistently outperformed other models in predictive accuracy and robustness, particularly in autoregressive time-series predictions. This suggests that architectures incorporating temporal modeling capabilities, like LSTMs, are more effective in capturing the dynamics of unsteady fluid flows. In contrast, other

research indicates that FCNNs can outperform LSTMs in specific applications. For example, a study on turbulent flow simulations validated a CNN-LSTM model, demonstrating its ability to provide correct turbulence statistics over extended timeframes, proving effective in capturing spatial features and nonlinear dependencies. Overall, both FCNN and LSTM models exhibit strong performance in predicting aerodynamic forces. However, the LSTM model's superior accuracy in this study, particularly after hyperparameter tuning, suggests that its ability to model temporal sequences makes it particularly well-suited for tasks involving time-dependent aerodynamic phenomena. These results underscore the potential of deep learning techniques, especially those incorporating temporal dynamics, to enhance predictions in fluid dynamics applications.

5 Summary and outlook

5.1 Summary

The study explores the use of neural networks, particularly FCNNs and LSTM networks, to predict aerodynamic forces such as drag, lift, and pressure distribution around an oscillating cylinder. Traditional CFD simulations, while accurate, are highly computationally expensive, making realtime applications difficult. To overcome this, a surrogate model is developed using high-fidelity CFD-generated time-series data from OpenFOAM. The numerical setup involves simulating a two-dimensional oscillating circular cylinder at a Reynolds number of 100, capturing vortex shedding and flow dynamics under varying frequencies and amplitudes. Governing equations based on the Navier-Stokes formulation are solved to establish flow characteristics, and a structured computational mesh is created to ensure accurate spatial and temporal resolution. The computational domain is defined with appropriate boundary conditions, and the simulation setup includes 34 different CFD cases with varying oscillation parameters. The collected data, consisting of force coefficients, pressure distributions, and velocity fields, is used to train deep learning models, aiming to replace direct CFD computations with a faster, data-driven approach.

The methodology involves extensive data preprocessing, including normalization, feature selection, data augmentation, and transient phase removal, to ensure meaningful training. An autoregressive time-series modeling technique with a sliding window approach is used to capture temporal dependencies. FCNNs are applied for spatial feature extraction, while LSTMs, which are specialized for sequential data, are employed to capture time-dependent flow patterns. Both models undergo hyperparameter tuning to improve prediction accuracy, and their performance is evaluated using Mean Squared Error (MSE), Mean Absolute Error (MAE), and correlation with CFD reference data. The results show that LSTMs significantly outperform FCNNs in predicting high-frequency oscillations and transient aerodynamic interactions, making them more effective for dynamic flow conditions. FCNNs, on the other hand, perform well for steady-state or low-frequency cases but struggle with temporal dependencies. The surrogate model reduces computational costs while maintaining high predictive accuracy, demonstrating its potential for real-time aerodynamic predictions. Visualizations of vortex shedding and force variations validate the models' predictions. The study concludes that machine learning-based surrogate models offer an efficient alternative to traditional CFD simulations, with future work focusing on extending the approach to three-dimensional flows and hybrid neural network architectures to enhance generalization and robustness.

5.2 Outlook

Future advancements in this field could focus on extending the neural network models to three-dimensional aerodynamic simulations. This would involve incorporating more complex fluid-structure interactions and handling the increased computational complexity of three-dimensional flow fields. By leveraging advanced architectures like hybrid models, which combine the strengths of both FC-NNs and LSTMs, it may be possible to further enhance the accuracy and efficiency of the predictions. Additionally, integrating real-time flow control strategies using reinforcement learning could lead to the development of more dynamic, adaptive control systems capable of optimizing aerodynamic forces in real-time.

Another potential direction for future research is the application of transfer learning to enhance the generalization capability of the models across different flow regimes. By training the neural networks on a broader set of simulations with varying flow conditions, the models could become more robust, ensuring accurate predictions even in previously untested scenarios. Incorporating unsupervised learning techniques and more advanced feature extraction methods could also improve the models' ability to learn from limited or noisy data, making them more applicable to practical, real-world applications where data may be sparse or incomplete.

6 Bibliography

1. Schafer, Michael and Turek, "Benchmark computations of laminar flow around a cylinder," 1996
2. A. Magrini, D. Buosi, F. Poltronieri, E. D. Leo, and E. Benini, "CFD-Based Analysis of Installed Fuel Consumption and Aerodynamics of Transonic Transport Aircraft During Cruise Flight," Energies, vol. 16, no. 8, p. 3323, 2023.
3. A. Weiner and J. Geise, "Model-based deep reinforcement learning for accelerated learning from flow simulations," Journal of Fluid Mechanics, vol. xxxx, 2021, doi: 10.xxxx/jfm.2021.123456.
4. C. Chen, B. Zhang, H. Huang, Z. Xie, C. Yang, D. Meng, and H. Yue, "A Multi-Task Learning Framework for Aerodynamic Computation of Two-Dimensional Airfoils," Physics of Fluids, vol. 36, no. 11, p. 117141, 2024. Link.
5. D. M. Lemos, F. D. Marques, and A. J. M. Ferreira, "A Review on Bistable Composite Laminates for Aerospace Applications," 2024.
6. D. Thummar, "Active flow control in simulations of fluid flows based on deep reinforcement learning," Technische Universität Braunschweig, Studienarbeit Nr. 794, 2021, supervised by Dr.-Ing. Andre Weiner.
7. E. A. R. Camacho, A. R. R. Silva, and F. D. Marques, "Optimal Leading-Edge Deflection for Flapping Airfoil Propulsion," 2023.
8. E. A. R. Camacho, M. M. Silva, A. R. R. Silva, and F. D. Marques, "Real-Time Optimization of Wing Drag and Lift Performance Using a Movable Leading Edge," 2024.
9. G. L. S. Torres and F. D. Marques, "Nonlinear Geometric Decomposition of Airfoils into the Thickness and Camber Contributions," 2024.
10. J. A. I. da Silva, L. Sanches, G. Michon, and F. D. Marques, "An Enhanced Nonlinear Energy Sink for Hybrid Bifurcation Passive Mitigation and Energy Harvesting from Aeroelastic Galloping Phenomena," 2024.
11. J. Camacho, F. D. Marques, and A. R. R. Silva, "Fast Flapping Aerodynamics Prediction Using a Recurrent Neural Network," 2023.
12. J. Geise, "Robust model-based deep reinforcement learning for flow control," Braunschweig University of Technology, Student Thesis No. 837, 2023, supervised by Dr.-Ing Andre Weiner.
13. W. Jia and Q. Chen, "A Convolutional Neural Network-Based Stress Prediction Method for Airfoil Structures," 2024.

14. X. Liu, Y. Chen, and Z. Wang, "Deep Learning-Based Fast Prediction of Aerodynamic Parameters for Ducted Propellers," 2024.
15. Y. Shi, Q. Lan, X. Lan, J. Wu, T. Yang, and B. Wang, "Robust Optimization Design of a Flying Wing Using Adjoint and Uncertainty-Based Aerodynamic Optimization Approach," Structural and Multidisciplinary Optimization, vol. 66, p. 110, 2023.
16. Y. Zhang, L. Wang, and H. Li, "Data-Driven Prediction of Aircraft Aerodynamic Coefficients Using GA-LM-BP Neural Network," 2024.
17. S. K. Singh and R. K. Sinha, "Recurrent Neural Network for Estimation of Aerodynamic Parameters," 2021.
18. Sangam Khanal and shilaj Baral and Joongoo Jeon,"Comparision of CNN-based deep learning architectures for unsteady CFD acceleration on small datasets," 2025

Statement of Authorship

I hereby declare that this research project about

Analysis of neural network architectures for predicting time-series data from simulations

has been composed by myself and describes and constitutes my own work unless otherwise acknowledged in the text. All references and verbatim extracts have been distinguished and all sources of information have been specifically acknowledged. The research project was handed in to the Department of Mechanical Engineering at Technical University of Dresden.

Dresden,

Sumanth Reddy Yeddula