COMPUTATIONAL FLUID DYNAMICS ANALYSIS SURROGATES BASED ON POLYNOMIAL REGRESSION AND CONVOLUTIONAL NEURAL NETWORK MACHINE LEARNING

By

Katherine Lee Owens

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Computer Science

May 12, 2023

Nashville, Tennessee

Approved:

Theodore Bapty, Ph.D.

Jason Scott, Ph.D.

ACKNOWLEDGMENTS

I would like to express my gratitude to Dr. Theodore Bapty for his guidance and mentoring throughout my research efforts. His guidance in modeling solutions and machine learning was extremely instrumental, but most of all his creative approaches to innovative solutions to engineering problems were invaluable. Additionally, I would like to thank Mr. Fred Eisele for his numerous contributions to my research efforts. His deep understanding and extensive experience in both engineering and machine learning contributed immeasurably to the overall success of my research project. Finally, I would like to express my gratitude to Dr. Jason Scott for his overall review and extremely informed input into this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Computational fluid dynamics (CFD) is the standard method for simulating fluid flow through and around various flow fields. The main drawbacks to CFD are the labor-intensive setup required to produce accurate results and the high computational costs to run a single simulation. Accurate CFD results require high resolution meshes, containing millions of mesh elements. The governing fluid dynamic equations are the continuity, moment, and energy equations which are used to derive the Navier-Stokes equation (Anderson, 1992). The governing equations are iteratively solved for every mesh element. Reaching suitable convergence can take thousands of iterations. This lends itself to computationally expensive simulations that can take days to complete. Current methods for speeding up CFD analysis without compromising the accuracy of the solution include domain decomposition, parallel processing, or utilizing supercomputers (Moureau et al., 2011).

One use of CFD is to find an optimal geometric design through design space exploration. For example, when designing an aircraft, a geometry with low drag is desired. To find a design to accomplish this requires testing different fuselage lengths, nose cone shapes, wing sizes, and location of the wings along the fuselage. Currently, doing this entails running separate CFD solutions for each geometric variation. To do so with high-resolution CFD simulations is extremely time consuming. Machine learning can be used in conjunction with CFD to reduce computational times. There are two main approaches for using machine learning to create surrogates for CFD simulations. The first is using physics-based machine learning algorithms. In this approach, fluid dynamic equations are integrated into the neural networks to aid the network's predictive abilities (Brunton et al., 2020). Another approach is to train the neural networks with physics-based models or images using a network without integrated physics knowledge (Thuerey et al., 2020). This paper proposes two versions of the latter approach to create surrogates for CFD analysis. The objective of this work is to develop fast methods for computing an estimate of the system model, given pre-computed component solutions and system specifications. The surrogates considered are polynomial regression and convolutional neural network machine learning algorithms. The polynomial regression surrogate uses the dimensions of the geometry under consideration to predict the drag force, lift force, and moment. The polynomial regression model performed well but did not generalize to unseen geometries. The CNN surrogate uses pre-computed pressure contours from each component in the geometry under analysis to predict the surface pressure on the aggregation of components. This approach allows surface pressure predictions for parametric exploration in a matter of minutes. The CNN surrogate predicted surface pressure values on geometries seen in the training

set with 93.96% of the predictions having less than 25% error and unseen geometries with 76.91% of the predictions having less than 25% error. Additionally, the CNN surrogate demonstrated a 3.5 times decrease in execution time compared to traditional CFD methods.

1.1 Problem Context

The problem context was based on the Symbiotic Design for Cyber Physical Systems (SymCPS) project (Wilding, 2022), which proposed using artificial intelligence (AI) to synthesize designs. Figure 1.1 shows the overall process. As the AI explores the design space, it will create many thousands of candidate designs, which must be assessed for performance. This assessment is the responsibility of the design oracle (Bapty et al., 2022), which uses physics-based analysis on the candidate design specification. Given the large number of candidate designs, there is a significant pressure to minimize the computational expense. It is not the intent of this thesis to describe the SymCPS capabilities, but instead describe the portion of the SymCPS that was applicable to CFD. The overall approach, pertinent to CFD, was to explore the design space based on the Corpus, programmatically generate CAD designs, and to programmatically perform CFD analyses on those designs. Subsequent sections describe this process in detail. As previously mentioned, the key limitation of CFD analysis is the considerable time to run a single analysis. If the design space consists of thousands of design points, then the analysis time becomes a significant barrier. To overcome this barrier, machine learning was used to create a surrogate for the CFD analysis for the domain represented by the seed designs.

1.2 Machine learning Methodologies

The subsequent sections explain the two approaches to CFD surrogate modeling in further detail. Polynomial regression algorithms are used to model non-linear relationships between the independent and dependent variables to predict the drag, lift and moment values. Convolutional Neural Networks (CNNs) are supervised deep learning algorithms used to model complex relationships between the independent and dependent variables to predict the surface pressure values.

1.2.1 Multivariable Polynomial Regression Surrogate

General regression analysis studies the relationship between two or more variables. Polynomial regression machine learning algorithm is a supervised learning model that is useful for predicting the dependent variable when the independent variable is curvilinearly related. Equation 1.1 shows the equation for polynomial regression where β is the parameter vector and k is the degree of the polynomial. The degree of the polynomial is preselected based on the complexity of the input data. Multivariable regression, shown in Equation 1.2, refers to regression models with more than one independent variable (Ostertagová, 2012). Combining

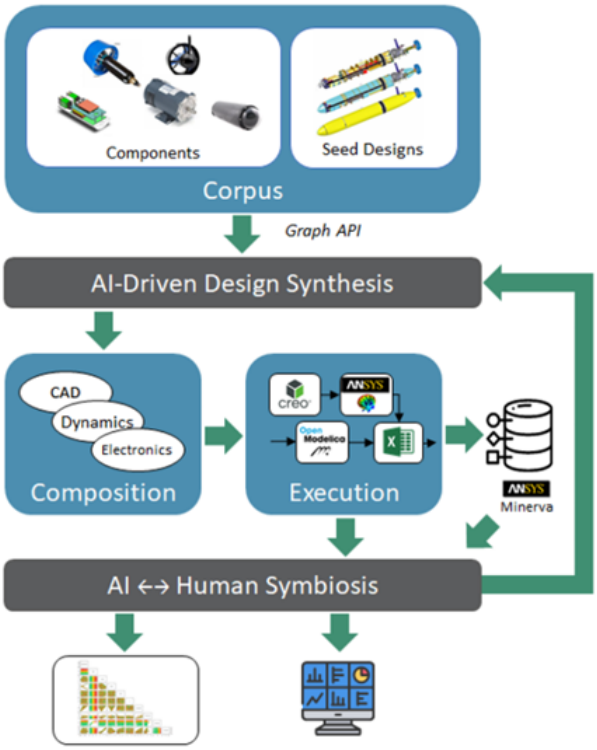

Figure 1.1: SymCPS Design Synthesis Process (Bapty et al., 2022)

polynomial regression and multivariable regression allows the regression model to learn complex curvilinear relationships between multiple independent variables and dependent variables.

$$y_i = \beta_0 + \beta_2 x_i + \beta_3 x_i^2 + \beta_4 x_i^3 + ... + \beta_k x_i^k \tag{1.1}$$

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + ... + \beta_k x_{ik} \tag{1.2}$$

The first surrogate approach utilizes multivariable polynomial regression machine learning models to make fast predictions of the drag force, lift force, and moment given the system specifications and the dimensions of the geometric model under analysis. The input data for this model is collected by performing CFD analysis on hundreds of parametric variations of the geometry. The independent variables contain information about the dimensions and rotations of each component in the system. The dependent variables are the drag, lift, and moment values from the CFD analysis with the corresponding independent variable parameters. Once the model is trained, the drag, lift, and moment prediction can be made for combinations of dimensions and rotations unseen in the training data without the use of CFD analysis.

1.2.2 Convolutional Neural Networks Surrogate

CNNs are commonly used in computer vision and image processing applications for classification and regression models (Li et al., 2021). CNNs are a type of feedforward neural network for supervised learning. CNN image processing consists of a series of convolutional layers and pooling layers. The convolutional layers produce a set of filters that the neural network can use to learn patterns and key features in the images (Ameri et al., 2019). Figure 1.2 (b) shows the convolution operation on a 2D image. The typical filter size is 3x3. Each filter is designed to detect edges, corners, or other features in the image. In this example, the 3x3 filter is applied to the top left and bottom right corners of the image. The 3x3 window usually starts at the top left corner and shifts to the right one column at a time until it has reached the last column in the image. Then the window will shift down a row and continues this process until the filter has covered the whole image. Each 3x3 window in the input image is multiplied element-wise by the filter and summed to produce the final value in the feature map. Convolutional layers are usually followed by pooling layers. Pooling layers are used to progressively decrease the image dimensions. This also reduces the number of parameters in the model (O'Shea and Nash, 2015). There are several types of pooling functions including: max pooling, average pooling, and general pooling. Figure 1.2 (a) shows an example of the 2D max pooling process. In this case, the pooling kernel size is 2x2. The pooling kernel is applied to 2x2 sections of the input image with a stride of 2. The max pixel values in each section are gathered to populate the representative feature map. After the series of convolutions and pooling layers, the image is flattened in preparation for input into the fully connected neural network layers.

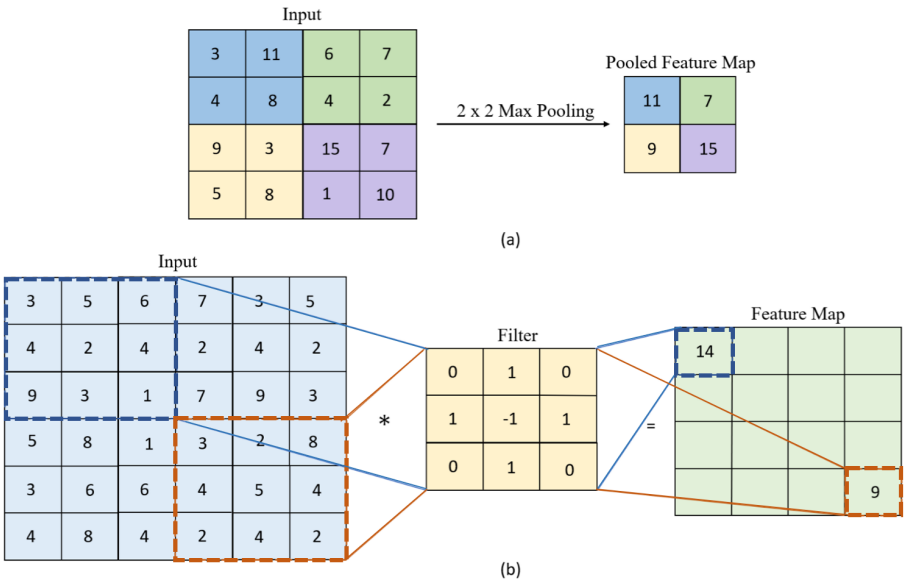

Figure 1.2: Convolution and Pooling Operations

The second approach to surrogate modeling uses convolutional neural network (CNN) machine learning models to predict properties of the system model. CNNs learn sequences of patterns in the input images making them a good candidate for learning the fluid flow patterns found in CFD analysis. The goal is to train a CNN model to predict the surface pressures on an assembly of parts. The CNN will do so by aggregating the CFD pressure domains of each individual component in the assembly. The components are known a priori, and can be analyzed beforehand; however, their combination cannot be known until the system model is known, which typically involves a full CFD solution. *The full CFD solution run would be replaced with the CNN surrogate that learns how to combine the pressure fields of individual components into a single pressure field for the assembly, wherein the combined pressure fields would yield similar results as a complete CFD solution.* A system design consists of an interconnection of a fixed set of components, along with values of parameters for those components that can control their geometric properties (size, rotation, shape, etc.). The interconnection of the components specifies how their shapes are placed/constrained relative to each other. The CNN surrogate executes very quickly, and thus allows the orientation and dimensions of the components to easily be modified and revaluated. This approach to design space exploration addresses the issue previously mentioned for performing CFD analysis on models with slight variations in order to find an optimal configuration of parts in the assembly. Since the parts are known a priori, and are typically used at known orientations, the expensive CFD analysis can be done once, and reused a large number of times, amortizing the computational cost.

CHAPTER 2

Related Work

Thuerey et al. (2020) used a modernized U-net CNN architecture to analyze the accuracy of pressure and velocity distribution predictions for Reynolds-averaged Navier–Stokes (RANS) solutions. The paper represented the CFD flow fields of 2D airfoils using Cartesian grids. They intentionally refrain from integrating physics based priors into the CNN architecture. This is done to gauge purely the CNN's predictions performance of the non-linear behavior of the Navier-Stokes equations without outside information. The CNN takes as input 128x128x3 images. Channel one of the input image contains the freestream velocity in the x-direction, channel two holds the freestream velocity in the y-direction, and channel three contains a masking image. The masking image, a binary image, indicates the regions of the flow field containing the airfoil and the regions containing the fluid. This paper investigated multiple models and found that its top performing model yielded predictions with less than a 3% relative error. Kashefi et al. (2021) identified the drawbacks of using a Cartesian grid system to represent 2D CFD flow fields. Those drawbacks included decreased order of accuracy from interpolation or extrapolation, uniform mesh density throughout the whole domain, and the coarsening of smooth surfaces on the model. To avoid these sources of error, Kashefi et al. (2021) proposes a regression adaptation of the novel point-cloud architecture developed by Qi et al. (2017) for classification and segmentation. The point-cloud grids allow for an unstructured representation of the CFD grid. This mitigates the drawbacks seen in the Cartesian grid approach by distributing more node points close to the geometry and wake region and less in the far-field. The point-cloud grid approach follows the mesh distribution generated from the CFD mesh. The CNN was trained on the pressure and velocity flow field of seven different shapes at various Reynolds numbers. The network's prediction successfully captured flow separation and generalized well to geometric shapes unknown to the CNN. Zhao et al. (2020) proposed a CFD-driven machine learning approach for predicting high-pressure wake mixing seen in turbomachines. These types of flow conditions are usually simulated with high-fidelity turbulence models such as large-eddy simulations (LES) or direct numerical simulations (DNS) (Zhao et al., 2020). These types of simulations are more computationally expensive than RANS, but also provide more accurate results. The paper uses gene-expression programming machine learning algorithms to enhance the flow predictions and increase the accuracy of RANS models for use in high-fidelity applications. The CFD-driven method was found to improve the accuracy in wake mixing predictions.

CHAPTER 3

Multivariable Polynomial Regression Approach

3.1 Model: Remus UUV

The Remus unmanned underwater vehicle (UUV) was modeled with computer-aided design (CAD) software [1] using the dimensions described in Prestero (2001). The Remus model is shown in Figure 3.1. The model is composed of seven components: the nose section, midbody section, tail section and four fins. The dimensions of the Remus model are shown in Table 3.1. The objective is to create a model-based design space exploration of UUV's, composing of systems using a corpus of components. This is done with the help of OpenMeta to systematically create hundreds of permutations of the Remus geometry, run CFD analysis using Ansys Fluent [2], and compute the drag force, lift force, and moment on the geometry.

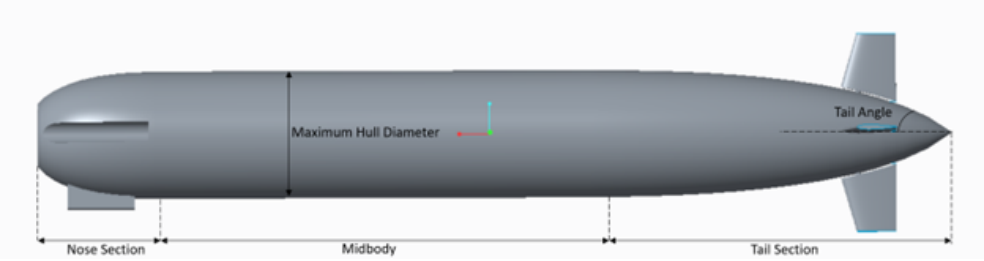

Figure 3.1: Remus UUV Model

3.2 OpenMeta Test Bench

The OpenMeta CFD test bench resides in an OpenMeta model. The test bench invokes three programs: CyPhy2CAD, CreateAssembly, and Run_AnsysFluent.py. The CyPhy2CAD interpreter creates files that describe the CAD assembly and test bench parameters. The CreateAssembly program generates a Creo assembly and exports it as a STEP file based on the geometry format specifications detailed in the test bench. The Run_AnsysFluent.py program opens an Ansys workbench and executes the geometry, meshing, and analysis steps, and then produces drag, lift, and moment calculations. The OpenMeta CFD test bench is composed of the geometric system under test, parameters, metrics, a code block (i.e. Python script), and Cyphy2CAD interrupter. Figure 3.2 shows the layout of the test bench. The parameters are grouped according to their purpose. The enclosure parameters, shown in 3.3, control the dimension of the fluid enclosure and the refinement region enclosure. Each enclosure offset value is the distance from the edge of the geometry to the wall

[1] PTC Creo Parametric 3.0 was used for all CAD modeling
[2] Ansys® Academic Research Fluent, Release 21.1

Table 3.1: Remus Dimensions

PARAMETER	VALUE	UNITS
NOSE SECTION	0.19	METERS
MIDBODY SECTION	0.654	METERS
TAIL SECTION	0.541	METERS
MAXIMUM HULL DIAMETER	0.191	METERS
TAIL ANGLE	0.436	RADIANS

of the enclosure. For example, the enclosure offset in the negative x-direction is measured from the furthest point of the Remus in the negative x-direction to the wall of the respective enclosure. The meshing parameters, shown in 3.4, govern the mesh cell sizes in various regions of the mesh. The mesh element size and max element size parameters control the global cell sizes, the body of influence element size parameter controls the refinement region cell sizes, the mesh curvature normal angle and curvature min size parameters control the cell size around the geometry. The Fluent parameters dictate the CFD inputs for Ansys Fluent solver. The inputs include inlet velocity, the type of fluid, the maximum number of iterations the solver should execute, and the drag, lift, and gravity direction relative to the geometry. The metrics are the post-processing results collected in a CSV file after the CFD analysis is completed. Figure 3.5 shows the internal components of the system under test. The system under test contains the seven CAD parts and all the necessary information to assemble the parts relative to each other. Each of the CAD components has additional parameters that control the part dimensions. The geometric parameters can be automatically varied in a Parametric Exploration Test (PET) run to generate and test hundreds of variations.

3.3 Machine Learning On Remus Model

A multivariable polynomial regression algorithm was used to predict the drag, lift, and moment values. Data was collected by running the OpenMeta PET for 175 iterations. 70% of the data was used to train the model and 30% of the data was used to test the model. The independent variables in the data include the nose length, midbody length, tail length, rotations about the x, y, and z axes, and inlet velocities. The dependent variables are the drag force, lift force, and moment. The independent variables are the parametric dimensions under exploration. Any outlier values in the data were removed to prevent additional error. Figure 3.6 shows the actual values compared with the predicted machine learning values of the drag force, lift force and moment. Table 3.2 shows the R-squared score of the plotted actual vs. predicted results. The drag predictions showed the best results with an R-squared value of 0.992. The lift force showed slightly lower prediction performance with an R-squared value of 0.912. The moment prediction had the lowest accuracy with an R-squared value

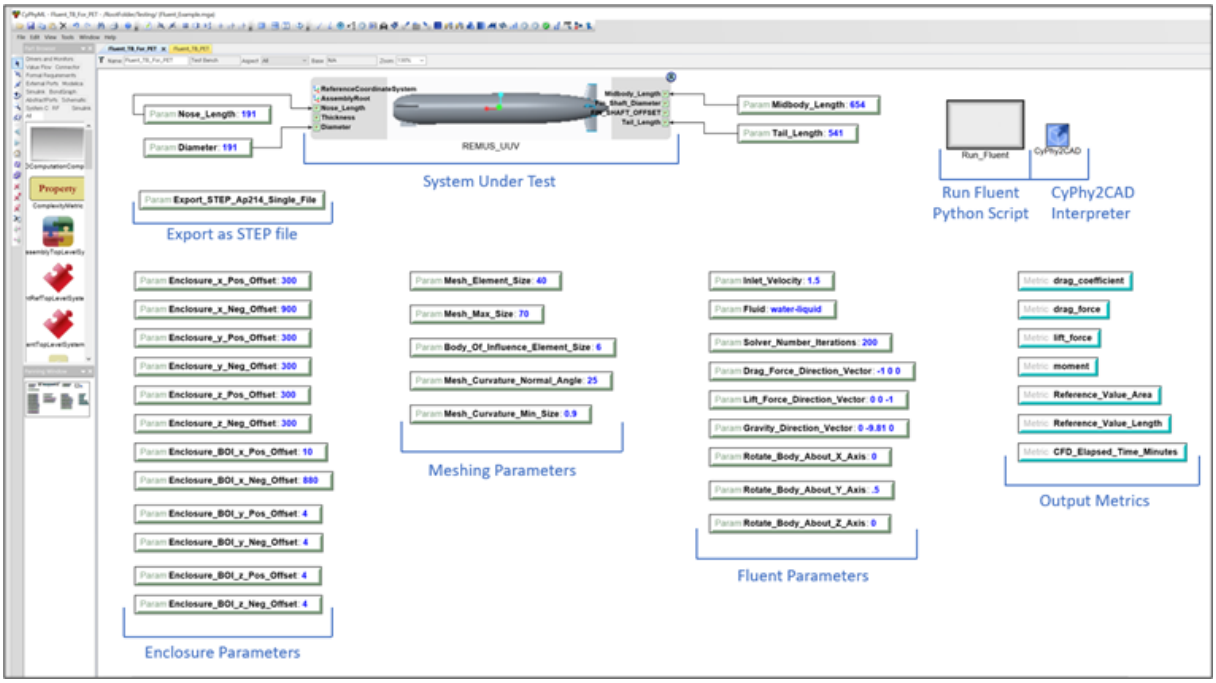

Figure 3.2: OpenMeta CFD Test Bench

Table 3.2: Polynomial Regression Results

Prediction	R squared Score
Drag Force	0.992
Lift Force	0.915
Moment	0.562

of only 0.562. The data set used to train and test the Polynomial Regression model was relatively small. Increasing the number of training points may increase the accuracy of the lift and moment prediction.

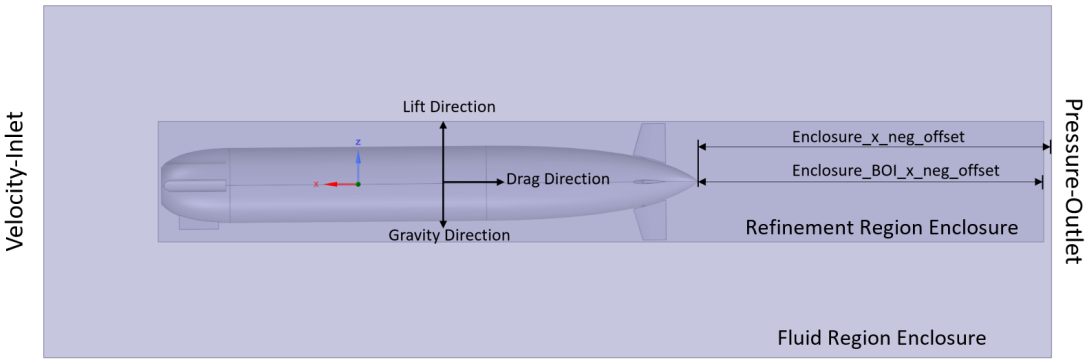

Figure 3.3: Remus Enclosure

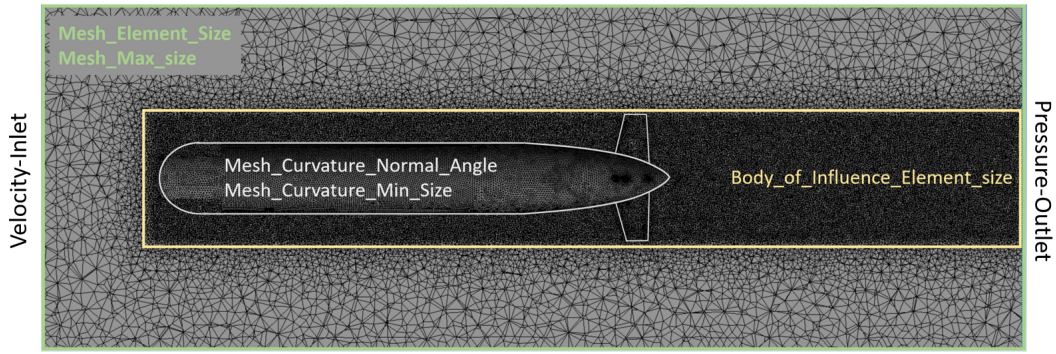

Figure 3.4: Remus Meshing Metrics

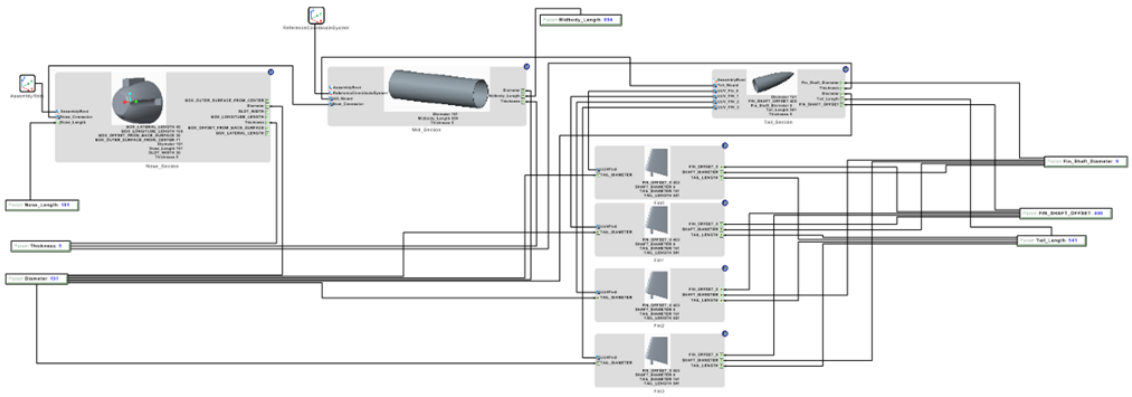

Figure 3.5: System Under Test

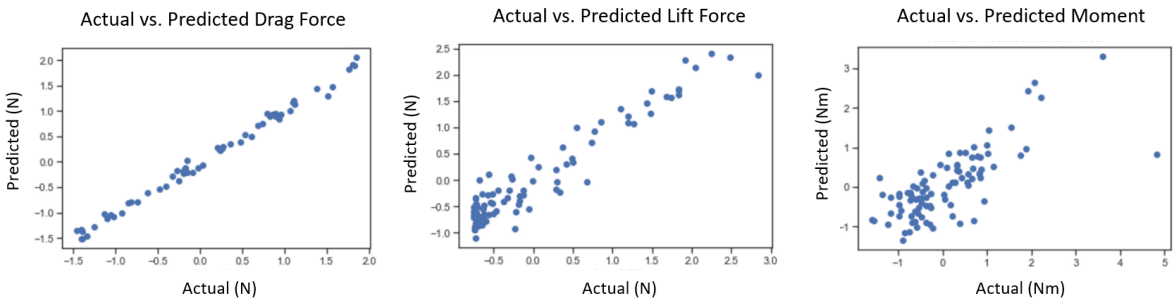

Figure 3.6: Polynomial Regression Results

CHAPTER 4

Convolutional Neural Network Surrogate

4.1 Model: Cube-Cylinder-Sphere

The model selected to train and test the CNN consists of simple parts with known drag and lift values. This particular geometry was selected because it captures multiple fluid flow phenomenon such as the formation of turbulent boundary layers, the detachment and reattachment of the boundary layers, transitions between laminar and turbulent flow, recirculation regions, and the turbulent interactions between the fluid flow around each component. For this reason the geometry is a good test for evaluating the abilities of the CNN to predict complex fluid flows. The sphere, cylinder, and cube are 3-dimensional CAD parts. The geometry is shown in Figure 4.1 with the dimensions listed in Table 4.1. Three variations of this geometry are used. The cube and sphere dimensions are constant throughout the three models. The length of the cylinder is the parametric component under evaluation. This allows the machine learning model to be exposed to different turbulent interactions as a result of the different cylinder lengths. The x-axis aligns with the cylinder center line and each model is symmetric about the y-axis and z-axis.

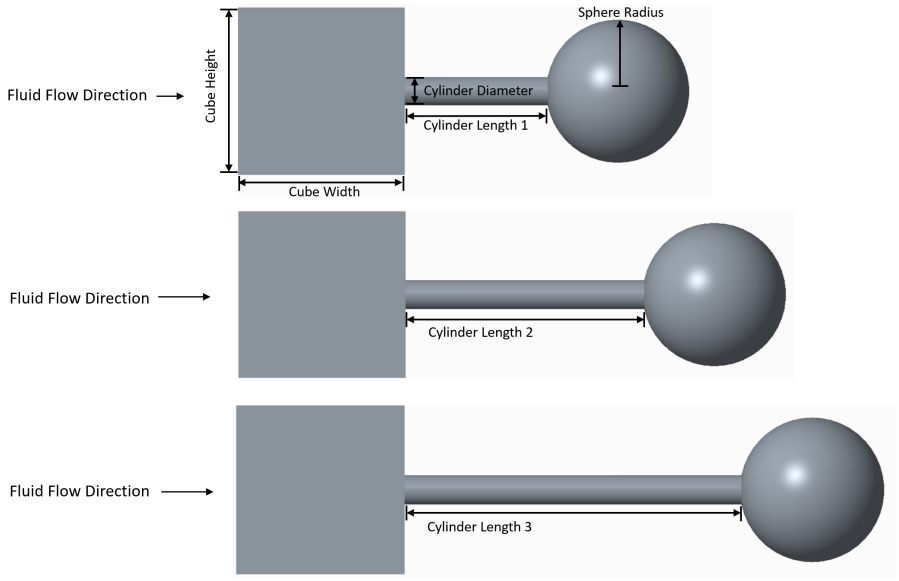

Figure 4.1: Assembly Geometry

Table 4.1: Geometry Dimensions

DIMENSION	VALUE	UNITS
CUBE HEIGHT	175	MM
CUBE WIDTH	175	MM
CUBE DEPTH	175	MM
CYLINDER LENGTH 1	150	MM
CYLINDER LENGTH 2	250	MM
CYLINDER LENGTH 3	350	MM
CYLINDER DIAMETER	30	MM
SPHERE RADIUS	150	MM

4.2 CFD Analysis

The CFD analysis process consists of four steps: geometry set up, mesh generation, fluent solver, and post-processing.

4.2.1 OpenMeta

Similar to the Remus model, an OpenMeta CFD test bench was created to perform the CFD analysis for assemblies 1, 2, and 3. The configuration of the assembly model is specified in the system under test. The test bench parameters are grouped into four categories: enclosure parameters, meshing parameters, Fluent parameters, and export parameters. The enclosure parameters dictate the dimensions of the fluid enclosure and the mesh refinement region. The meshing parameters control the sizes of the mesh elements in each region of the fluid domain. The Fluent parameters provide the necessary information to the Fluent CFD solver, such as the inlet velocity and fluid properties. Finally, the export parameters tell the CAD software the desired export format (e.g. STEP). The test bench also includes output metrics collected from the CFD analysis.

4.2.2 CFD Setup

To model the fluid flow around the selected geometry, the Reynolds-Averaged Navier–Stokes (RANS) turbulence model and k-ε viscous model for incompressible flow were used with a constant inlet velocity of 1.5 m/s. These are the typical CFD setting used for this category of fluid flow problems. The boundary conditions are shown in Figure 4.3. Two enclosures were created around the geometry. The smaller enclosure labeled “Mesh Refinement Region” is only for meshing purposes and does not affect the boundary condition. The inlet is located to the left of the enclosure and is given the velocity inlet boundary condition. The outlet is on the opposing end of the enclosure and is given the pressure outlet boundary condition. The exterior enclosure walls are assigned a slip boundary condition to simulate an unbounded fluid. The exterior of the geometry is given a no-slip boundary condition. Water was selected for the incompressible fluid. The lengths

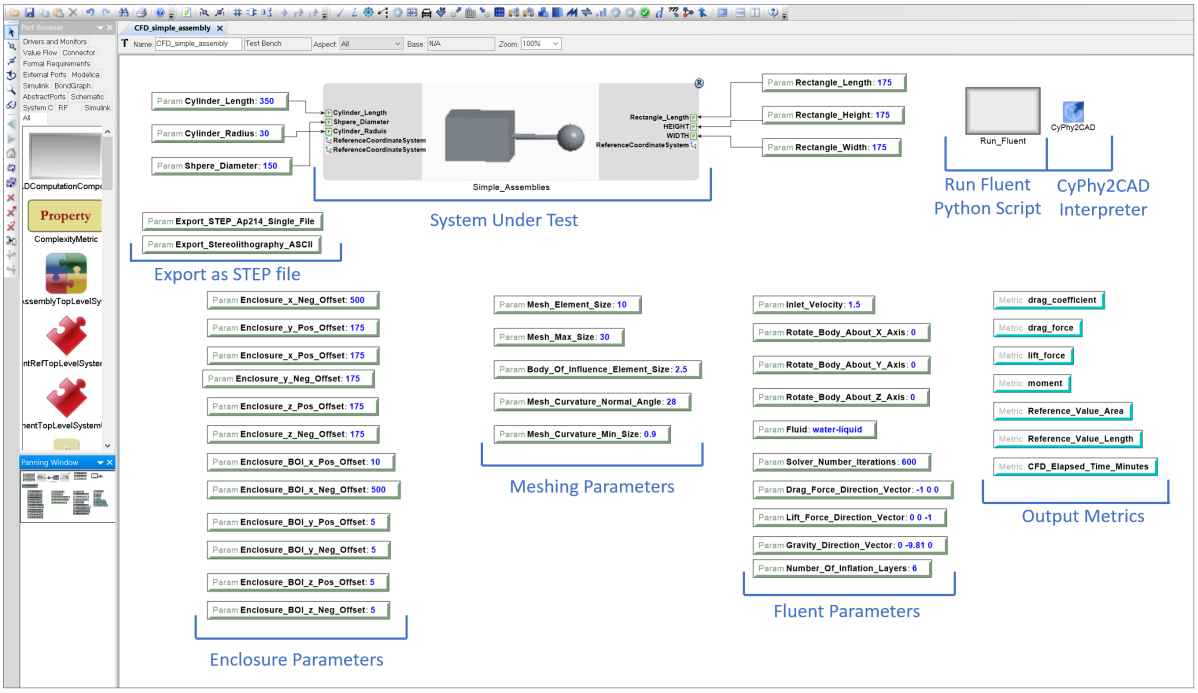

Figure 4.2: CFD OpenMeta Test Bench

of the assemblies are all slightly greater than half a meter. CFD analysis was performed on each of the three assemblies and each individual part in the assemblies (i.e. cube, cylinder in each of its three characteristic dimensions, and sphere) using the same boundary conditions, viscous model, and inlet velocity. The mesh and meshing metrics, shown in Figure 4.4 and Table 4.2, were used for CFD analysis on each of the assemblies and each of the individual parts. Tetrahedral meshing elements were used throughout the fluid domain. Element size indicates the general mesh size. The max element size indicates the largest size of any mesh element in the domain. The body of influence element size is the general size of the mesh element in the body of influence refinement region. The mesh curvature normal angle controls the mesh refinement level around curved features in the geometry. Finally, the mesh curvature min size limits the size of mesh elements refined based on the curvature normal angle. The most refined mesh is contained in the regions closest to the model allowing the CFD analysis to produce accurate results while still being relatively efficient computationally. Each CFD simulation was run for 600 iterations allowing the residual error to be reduced.

4.2.3 OpenMeta CFD Test Bench Execution Times

The time to run the OpenMeta CFD test bench for each of the three assemblies and each component in the assemblies is shown in Table 4.3. The execution time includes the CAD model generation, CFD enclosure set up, CFD meshing, and the time for the CFD solver to complete 600 iterations.

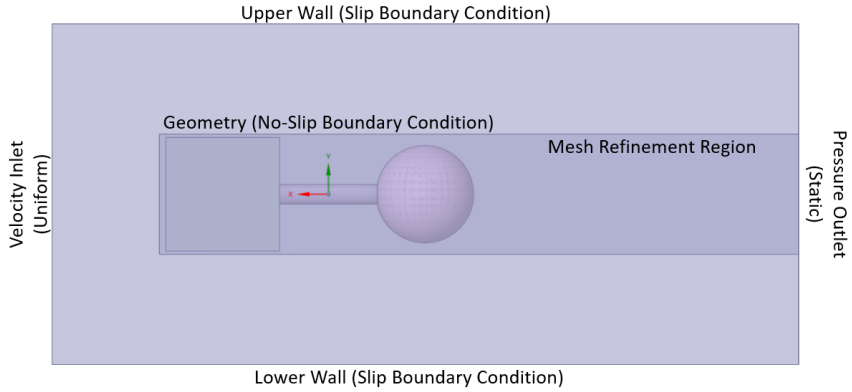

Figure 4.3: CFD Setup

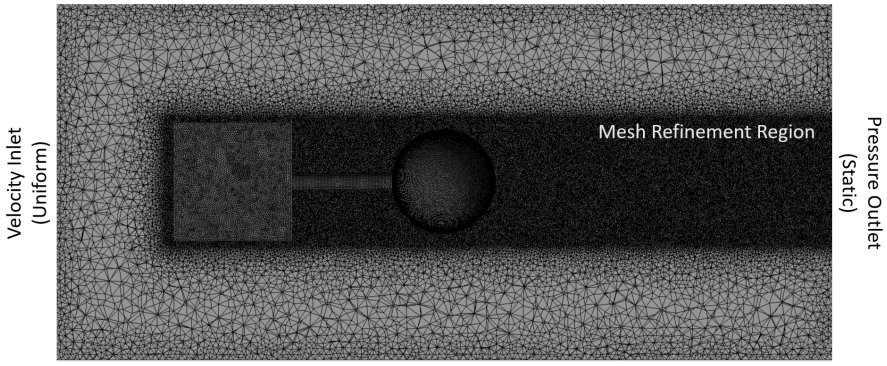

Figure 4.4: CFD Mesh

4.2.4 Model Flow Fields

Selecting a non-streamline model adds additional complexity in the CFD analysis. Having the cube at the forefront of the model creates a high pressure region on the front face of the cube normal to the fluid flow. This also creates a large pressure gradient between the front face of the cube normal to the flow and the four neighboring faces. Figures 4.5 and 4.6 show the velocity and pressure contours respectively for the three assembly models. In each figure, the top contour is of assembly 1, the middle of assembly 2, and the bottom of assembly 3. Changing the length of the cylinder has a noticeable impact on the velocity and pressure contours of each model variation. Assembly 1 has the shortest cylinder length; therefore, the turbulence created from the cube has the largest impact on the cylinder and sphere. Assembly 3, on the other hand, has the largest cylinder length, and therefore the turbulent interaction between the cube and the cylinder and sphere is the smallest. Even though the three models only differ in one dimension, the pressure and velocity contours still vary significantly. The fluid flow around the cube, shown in the velocity contours for each

Table 4.2: Meshing Metrics

Meshing Metric	Value	Units
Element Size	10	mm
Max Element Size	30	mm
Body of Influence Element Size	2.5	mm
Mesh Curvature Normal Angle	28	mm
Mesh Curvature Min Size	0.9	mm

Table 4.3: OpenMeta CFD Test Bench Execution Times

CFD Analysis	Time To Complete (Minutes)
Assembly 1	179.2
Assembly 2	183.0
Assembly 3	224.0
Cube	125.3
Cylinder 1	13.0
Cylinder 2	16.4
Cylinder 3	16.6
Sphere	103.3

assembly, undergoes the same fluid interactions. Assembly 1 has a slightly wider low-velocity region in the flow field directly behind the sphere, opposed to assembly 3's narrower low velocity region. In assembly 1 the sphere is in the wake of the cube, whereas in assembly 3 the sphere is far enough from the cube that the effects of fluid flow around the sphere are seen separately from the cube. The pressure contours show the same general trends. The most noticeable difference is seen in the high pressure region preceding the sphere. Assembly 1 has the smallest of the high pressure regions, as a result of the close proximity to the low pressure region behind the cube. As the length of the cylinder increases, the length of the high pressure region also increases. In assembly 3, a more prominent negative pressure region is seen around the middle of the sphere.

4.2.5 Drag, Lift, and Moment Calculations

The CFD mesh is composed of tetrahedral mesh elements that represent the fluid domain and the interface between the fluid and solid domain. Figure 4.5 shows a single tetrahedral mesh element. Each tetrahedral element consists of four faces connected by four vertices. Each face of the tetrahedral element has a normal vector that represents the magnitude of the force on that face. The magnitude of a vector is the root sum squared of its x, y, and z components. The direction of the normal vector is determined using the right hand rule. In other words, if the vertices are numbered in a clockwise ascending order then the normal vector will point towards the element face. If the nodes are ordered in a counter-clockwise ascending order the

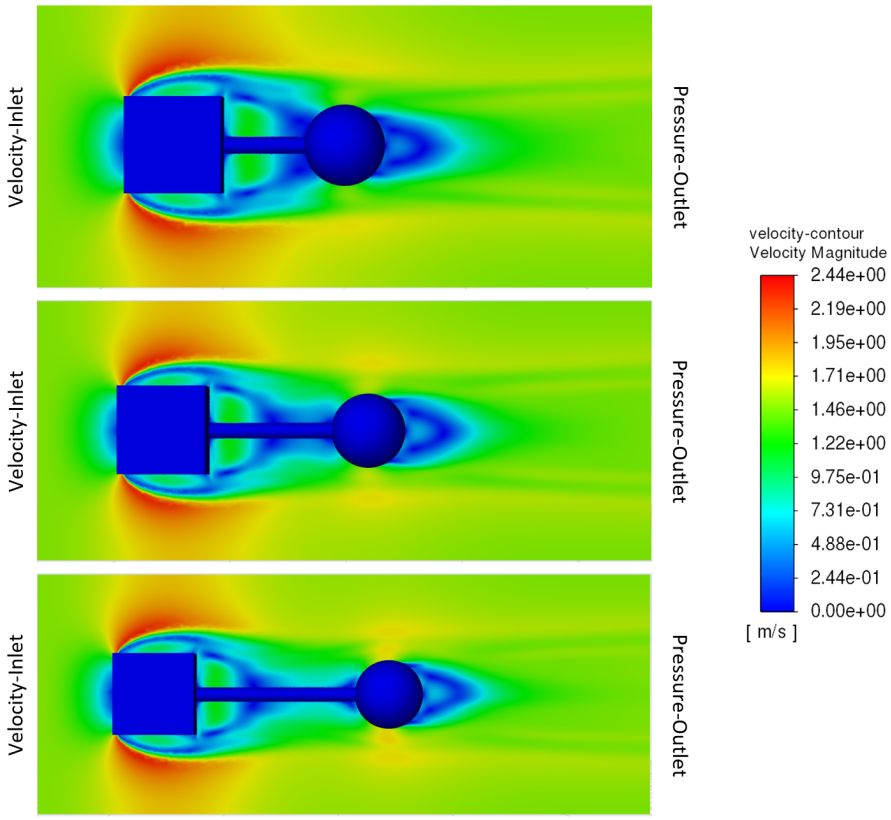

Figure 4.5: Velocity Contours

normal vector will point away from the element face. The drag and lift forces on an assembly are found by summing the forces on the surface of the model with respect to the desired axes. For example, the drag forces for assembly 1 would be computed by summing the forces on the surface of the model in the x-direction. Likewise, the lift force on assembly 1 would be computed by summing the forces on the model in the y-direction. The forces on the model can be derived from the surface pressures that the surrogate predicts. Pressure is a force distributed over an area. The force on each face is computed by taking the product of the pressure and surface area of the face, which results in a force in the direction of the normal vector. Computing the drag and lift forces can be done by writing a program to sum the forces in each direction or by using CFD post-processing software.

4.3 Data Collection

Data samples were collected based on the number of surface points in each model from the CFD meshes. Due to the unstructured nature of the CFD meshes, each assembly contains a different number of surface points. Samples were collected across a quarter section of the model. Since the models are symmetric, a

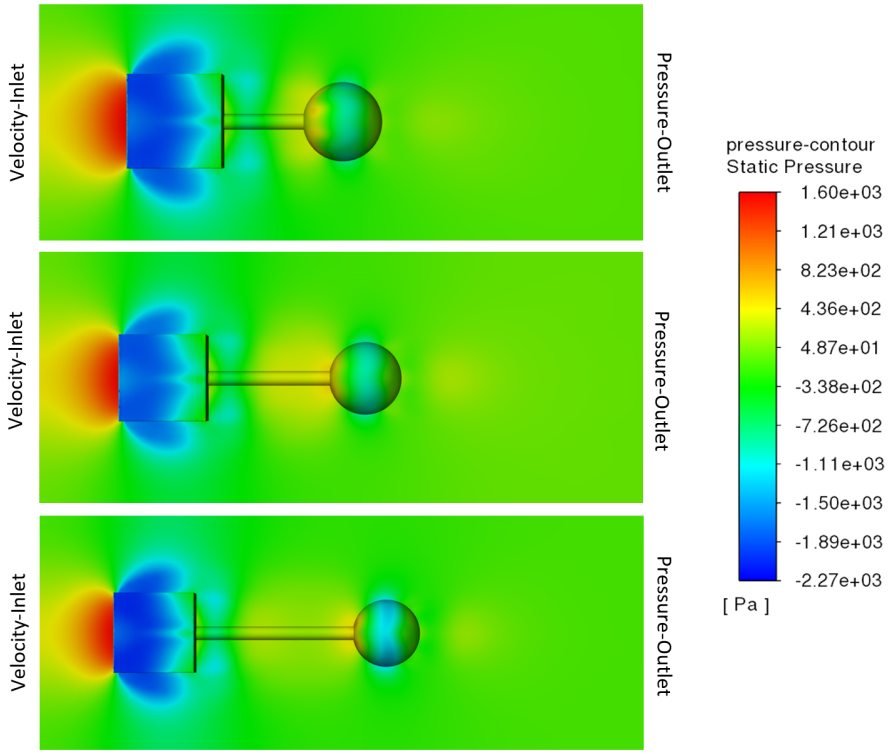

Figure 4.6: Pressure Contours

quarter section is used to reduce computational time. 80% of the data was used for training and the remaining 20% was used for testing. Python was used for all of the data collection and pre-processing. Due to the large amount of data, the data was stored in tables within a database [1]. Using a database allows the data to be stored in database files for future use.

4.4 CFD

CFD analysis was performed for each of the three models (assembly 1, assembly 2, and assembly 3), and for each individual part (cube, cylinder in each of its three characteristic dimensions, and sphere). All of the mesh points in the fluid and on the surface of the model are exported from the Ansys Fluent solver. Each point contains the x, y, z coordinates, x, y, z velocities, and pressure. The goal is to predict fluid properties on the surface of the model, so only the surface pressure values will be predicted. As a result of the no-slip boundary condition on the surface of each model, all velocity values on the surfaces are 0 $\mathrm{m/s}$. The CFD grid for the model assembly is centered about the $(0,0,0)$ coordinate point of the CAD assembly. This is also true for the CFD grids of the individual cube, cylinder, and sphere. To allow the coordinate points selected from the cube, cylinder, and sphere parts to reflect their positions in the assembly, each mesh is translated without

[1] SQLite3

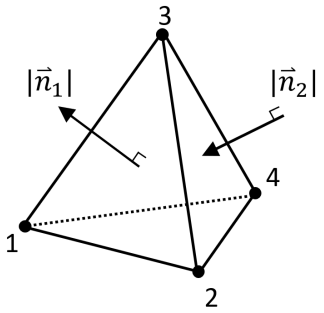

Figure 4.7: Tetrahedral Mesh Element

rotation into the position it would occupy in the model assembly.

4.5 Sampling

Samples are collected at every n surface point in the assembly. Where n is the specified increment between selected surface points. The increment size is selected based on the desired number of the training and testing points in the datasets. This allows for a representative sampling of the pressures at the surface of geometry. Evenly spaced $16 \times 16 \times 16$ Cartesian grids, referred to as probe volumes, are generated around each selected surface point. A probe volume is generated for each part in the assembly. In this case, probe volumes are created for the cube, cylinder, and sphere for each selected surface point on the assembly. The probe volumes are an interpolation method to convert a non-uniformly spaced mesh into a uniformly spaced grid. An $n \times n \times n$ box that encompasses the probe volumes is created around the coordinates of the selected surface point. The pressure values in that region are then collected from the cube, cylinder, and sphere CFD component grids. The pressure values from each part are then projected onto each respective probe volume. The probe volume stores each point's x, y, z coordinate relative to the assembly and the pressure value at each coordinate. The points are projected by selecting the points in the CFD grid within a specified radius of each evenly spaced point. A weighted sum, based on the distance from the probe volume point, is used to approximate the value on the evenly spaced grid. The CFD points closer to the probe volume grid points will, therefore, have a larger influence on the projected values than those farther away. Using the probe volume approach provides the CNN with a consistent number of input points. Figure 4.8 shows probe volumes created at various surface points throughout the geometry. The purple points at the center of the squares represent the selected surface points, and the purple square surrounding the point shows a 2D depiction of the size and shape of the 3D probe volume. Since the probe volumes are centered around the surface point, portions of those volumes are composed of the fluid and the other portion of the solid region. The pressure data is only present in the fluid domain and on the surface of the model. The lack of pressure data in the solid

domain can be used to identify the solid and fluid regions.

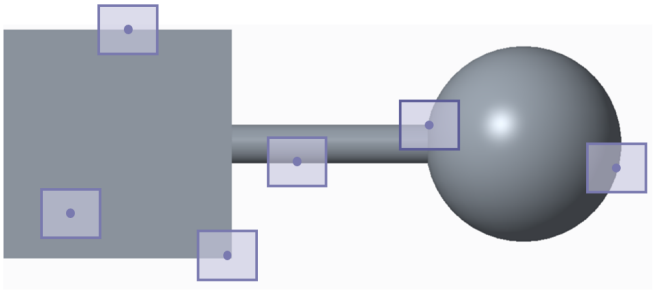

Figure 4.8: Probe Volumes

The top row of Figure 4.9 shows the points selected from the CFD meshes of the cube, cylinder, and sphere grids within the $n \times n \times n$ box around the surface point. In this example, the selected surface point lies on the sphere in the assembly, so only the outline of the sphere is shown in the sphere's plot. The point lies in the fluid flow region of the cube and cylinder grids, which explains why the sphere outline is not shown in those CFD mesh plots. In the model assembly, the flow around the cube and cylinder affects the flow around the sphere because both objects are upstream from the sphere. Therefore, the cube and cylinder CFD points are still relevant for the machine learning model. The second row shows the CFD grid points projected onto the evenly spaced probe volume. The probe volumes maintain the pressure gradients seen in the CFD grids with fewer points. Projecting the points onto the sparser grid will lose some of the detail found in the original grid, but should provide enough information for the machine learning model. The curvature of the sphere is shown in each of the probe volume plots because the goal is to evaluate the surface point relative to the assembly. In this case, the area inside the sphere does not contain pressure values.

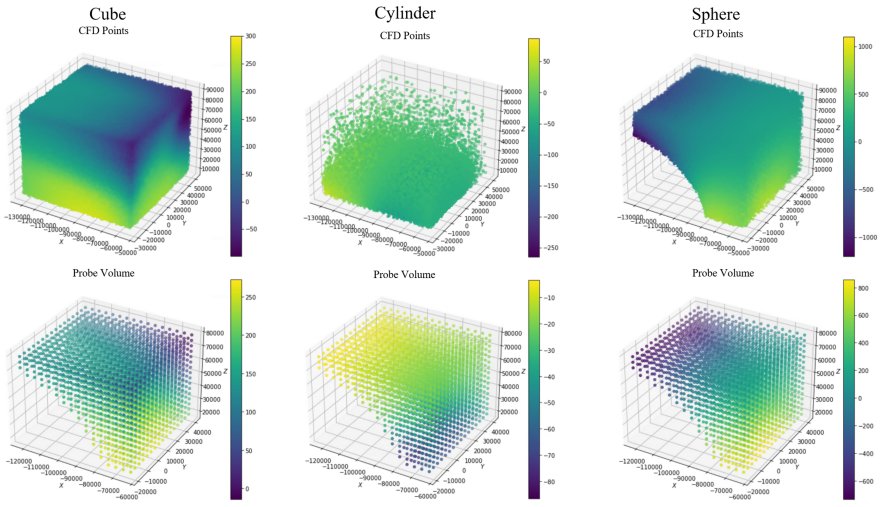

Figure 4.9: CFD and Probe Volume Grids

The cube, cylinder, and sphere fluid domains overlap, but not in all regions in the assembly fluid domain. The fluid domains of each part are shown in Figure 4.10, where the yellow region indicates the cube fluid domain, the blue region is the cylinder fluid domain, and the green region shows the sphere fluid domain. A surface point selected on the front face of the cube would only exist within the cube fluid domain. For this point, probe volumes will be created for all three parts, but only the cube probe volume will be assigned pressure values. Similarly a surface point selected on the back half of the cube would only lie within the cube and cylinder fluid domain. If the surface point is located on the edge of the cylinder domain there may not be sufficient mesh points to represent pressures in that region. To accommodate for this, a buffer region is created at the outer portion of each component's fluid domain. The buffer region ensures that the points will not be projected onto the probe volume for any surface point inside the buffer region of each component's fluid domain. In other words, only surface points inside the component's fluid domain but not within the component's fluid domain buffer will be projected onto the probe volume. In the case where the surface point is outside of one or more of the part's CFD grids, the probe volume is created with NULL values at each point in the probe volume. This prevents inaccurate pressure values from being collected in regions of the CFD mesh containing few to no grid points. The pressure fields of the component in that scenario would have minimal impact on the pressure at the selected surface point.

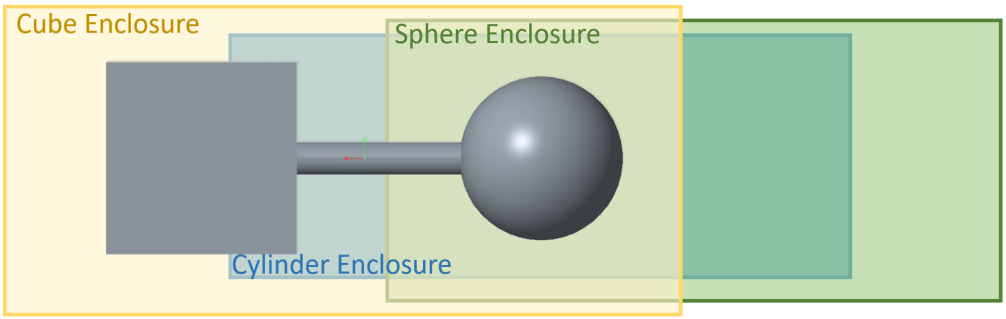

Figure 4.10: Overlapping Grids

4.6 Data Pre-Processing

CNNs take images as input and use a series of convolutional filters to process the images and identify key patterns. Typically, CNNs are used on 2D images; however, the CNN for the purpose of this paper will use 3D images. The input images are composed of three probe volumes, one for the cube, one for the cylinder, and one for the sphere. An additional masking image is provided as input, which indicates whether or not a point is in the fluid domain or within a component/solid region. Each probe volume and masking image is first

converted into a 16x16x16x1 grayscale image. The conversion process between probe volumes and grayscale images is explained further in the following section. Each image is then stacked on top of each other to make the final 16x16x16 image with four channels. The surface pressure values from the CFD assembly models are treated as the ground truth values. The CFD pressure values at each surface point, corresponding to a probe volume set, are collected and stored in an array for training and testing of the CNN model.

4.7 Image Generation

An image is essentially an array of pixel values. To store the desired 16x16x16x4 input image, an empty $n \times 16 \times 16 \times 16 \times 4$ array is created. Where n is the number of data samples collected. Four channels were selected to store information on the masking layer and the three components in the assembly. The first channel in the image holds the grid mask. The mask is an image used to identify the regions of the grid that contain points in the solid and the fluid domain. A pixel value of one is assigned to all points in the fluid region and a value of zero is given to all points in the solid region. Channels one, two, and three contain the cube, cylinder, and sphere probe volume images respectively. To convert each probe volume into an image, the probe volumes are first normalized to have a mean of zero and a standard deviation of one. This is a necessary step to help reduce the losses in the CNN. Each point in the probe volume with NULL values is set to zero. This indicates the point is within the part and therefore would not have a pressure value. Since these volumes are being treated as images, each point must have a value. Setting all NULL values to zero satisfies this constraint and creates a dark region in each image that the neural network can use to identify the space that the component occupies. The pixel values in the fluid regions are assigned the normalized pressure values at each respective point in the probe volume. The probe volumes have the same dimensions as the input images, so each point in the probe volume maps to a single pixel in the input image. The masking image and each probe volume image are then copied into the empty array in the desired channels and sample position. This process is repeated until images have been generated from each sample and added to the input data array. A second $n \times 1$ array is created to store all of the CFD ground truth surface point values. The input to the CNN is the $n \times 16 \times 16 \times 16 \times 4$ image containing four channels. A 2D grayscale image only contains one channel. Typical multiple channels are used in RGB images, where the image contains three channels. The first channel is for the red rendering of the image, the second for green, and the third channel for blue. In this case, the channels are used to store the mask and the different parts in the assembly.

4.8 Machine Learning

The architecture of the CNN plays a significant role in the model's performance. A shallow, but wide network could lend itself to underfitting of the model. On the other hand, a deep and narrow network could result in

overfitting of the model. Underfitting occurs when the model encounters high bias and low variance (O'Shea and Nash, 2015). This can cause the model to perform poorly on the training and testing data. Overfitting occurs when the neural network learns characteristics specific to the training dataset that do not generalize to the testing dataset. This presents itself as high accuracy on the training data and low accuracy on the testing data (Jabbar and Khan, 2015).

4.9 CNN Architecture

The CNN architecture was implemented in Python using the TensorFlow Keras library. A CNN with an input layer, four convolutional layers, two pooling layers, two fully connected layers, and an output layer was selected for use in predicting the pressure at a surface point on the model. Figure 4.11 shows this architecture in more detail. The first two convolutional layers have 16 filters with a 3x3x3 kernel size. These layers are followed by a pooling layer with a pool kernel size of 2x2x2. The pooling layer decreases the dimensionality of the image. The pooling layer is followed by two more convolutional layers with 32 filters each. The first convolutional layer has a kernel size of 3x3x3 and the second has a kernel size of 1x1x1. These convolutional layers are followed by another pooling layer with a 2x2x2 pooling kernel size. The images are then normalized using batch normalization and flattened. The flattened data is then fed into three fully connected dense layers, with drop out layers in between. The final layer is the output layer which produces the prediction of the surface pressure value of the point of the assembly. To train the model, the ADAM optimizer is used with a learning rate of 0.001. Additionally, a batch size of 128 and 1,000 epochs are used to fit the data.

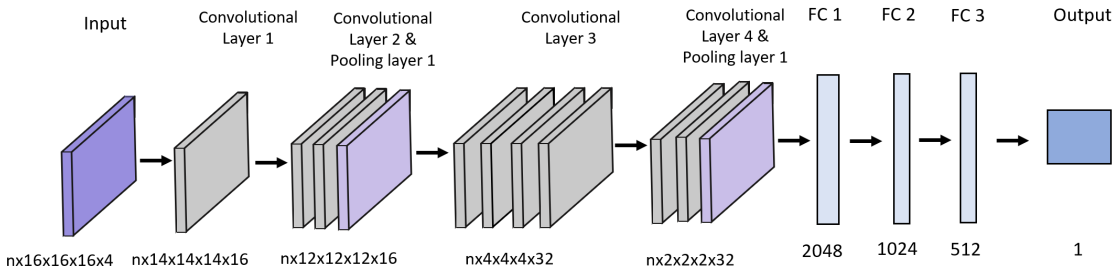

Figure 4.11: CNN Architecture

4.10 Information Flow

The information flow is shown in Figure 4.12. The OpenMeta CFD test bench exports files containing the pressure values at each mesh element in the fluid domain and on the surface of the geometry. The CFD solution files are loaded into the python environment. The solution data is then pre-processed, and probe volumes are generated for training and testing the CNN. The probe volumes are randomly shuffled and

divided into training and testing datasets. The CNN is then trained using the training dataset. The trained CNN can then be used to make surface pressure predictions in the testing dataset.

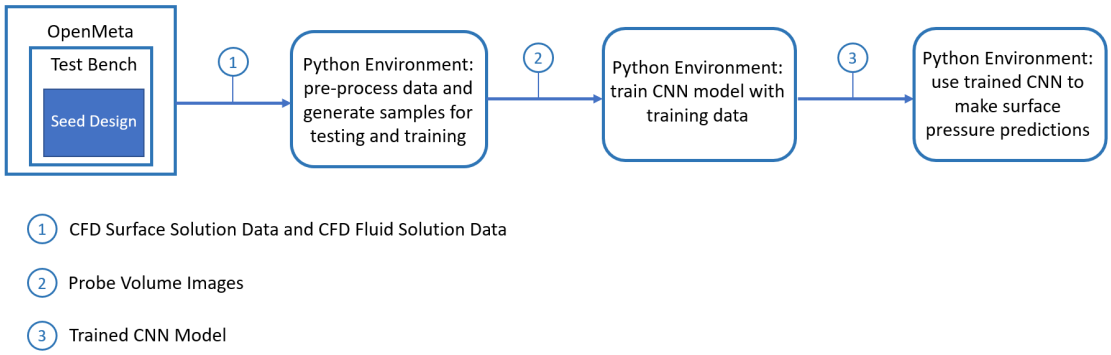

Figure 4.12: Information Flow Diagram

4.11 Results

The results of the CNN surrogate with and without the masking layer included in the probe volume are reported in the subsequent sections. The model's prediction accuracy is evaluated by the percent error between the actual and predicted pressure values. The percent error is given by Equation 4.1 where x_i is the predicted surface pressure value and $\hat{x}_i$ is the ground truth surface pressure values. The root mean squared deviation (RMSD) is an additional parameter used to evaluate the models performance that quantifies the similarity between the true value and the predicted values. The RMSD is shown in Equation 4.2 where x_i is the predicted surface pressure value, $\hat{x}_i$ is the ground truth surface pressure values, and N is the number of testing data points. The CFD surface pressure plot shows the pressure values for all points on the surface of the model computed from CFD analysis. The CFD surface pressure values are the ground truth values from which samples were taken. The actual surface plot shows the CFD surface pressure values for only the surface points included in the testing dataset. The predicted surface pressure plot shows the CNN's predictions for each point in the test set. The first error plot shows errors between the actual and predicted values for each point in the testing set with the same scale as the CFD pressure, actual surface pressure, and predicted surface pressure plots. The second error plot shows the same error values displayed with a smaller scale in order to bring attention to the high error regions that are often not seen in the first error plot.

$$PercentError = |\frac{x_i - \hat{x}_i}{\hat{x}_i}| * 100 \tag{4.1}$$

$$RMSD = \sqrt{\frac{\sum(x_i - \hat{x}_i)^2}{N}} \tag{4.2}$$

Table 4.4: Hardware Configuration

HARDWARE CONFIGURATION
AMD RYZEN 7 5800X CPU, 8 CORE, 3.81 GHZ 128 GB MEMORY

The hardware configuration of the computer used to run all CFD analyses and machine learning models are detailed in Table 4.13

4.11.1 Original Approach

The original approach for predicting the surface pressures did not include the masking layer in the input image. The input image used for this model was an $n \times 16 \times 16 \times 16 \times 3$ image where n is the number of samples and the three channels held the cube, cylinder, and sphere probe volume images. The solid regions of each image were represented by a pixel value of zero. The pixels in the fluid regions contained the representative pressure values. The hope was that the CNN would identify the regions of the image containing pixel values of zero as the solid region without additional information. The drawback to this approach is that the pressure values in the fluid region can be zero or close to zero. This may make it difficult for the CNN to differentiate between the solid regions and fluid regions with pressure values close to zero. Figure 4.13 shows the CNN's predictions compared to the actual CFD surface pressures and the associated prediction errors. The actual and predicted surface pressure plots appear to have the same high pressure region on the front surface of the cube and follow the same patterns in the pressure contour throughout the rest of the geometry. Table 4.5 shows the average percent difference between the actual and predicted surface pressure values and the percentage of the testing samples that have a percent difference above and below 25% error. The normalized data category represents the testing data that was normalized to have a mean of zero and standard deviation of one. This normalization is done to improve the CNNs predictions. The non-normalized category represents the values of the testing data after being converted back to its original pressure values that are measured in pascals. The average percent error for the normalized and non-normalized data was 20.96% and 27.68%. 89.53% of the testing data points had percent errors less than 25% for the normalized data and 93.01% of the testing data points had errors less than 25% for the non-normalized data. Most of the prediction error from this model was concentrated on the cube surfaces.

4.11.2 Improved Approach

Improvements in the previously described approaches are detailed in this section.

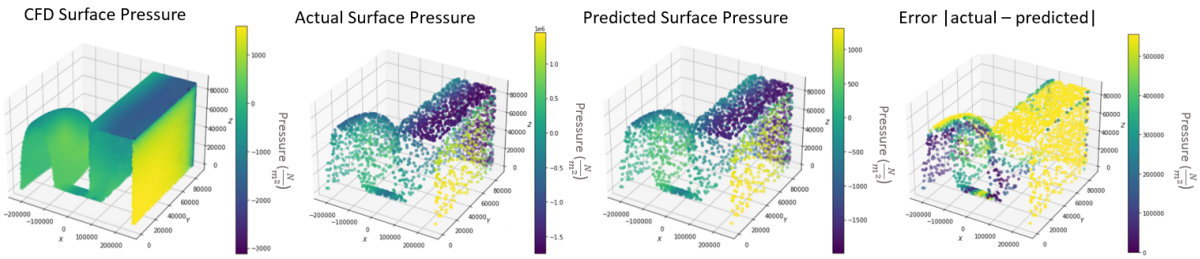

Figure 4.13: Train and Test on Assembly 1 (Without Masking Image) Predictions

Table 4.5: Train and Test on Assembly 1 (Without Masking Image) Results

	Normalized Data	Non-Normalized Data
Average Percent Error	20.96%	27.68%
Percent of samples with errors below 25%	89.53%	93.01%
Percent of samples with errors above 25%	10.47%	6.99%

4.11.2.1 Train and Test on Assembly 1

The previous approach to predicting the surface pressure values was improved by adding a masking image to the input image. Masking layer provides additional information to the CNN model to aid in discerning between the solid and fluid regions of the images. Figure 4.14 shows the actual pressure values, predicted pressure values, and the error. From inspection, both the actual and predicted pressures look similar. Both plots show the high pressure on the front surface of the cube, and the gradual increase in pressure along the top surface of the cube. The two error plots have different scales to show different perspectives of the error. The majority of the error is seen on the front and top surfaces of the cube. Relatively lower error values are seen on the cylinder and sphere. Table 4.6 shows the quantitative error measurements. The average percent error between the actual and predicted values for the normalized and non-normalized data was 16.64% and 9.17% respectively. 86.82% of the normalized samples in the testing set had less than a 25% error and 94.56% of samples in the testing set had less than a 25% error when the pressure values were converted back into Pascals. The normalized and non-normalized data had RMSD of 0.079 and 72.16 respectively.

4.11.2.2 Train and Test on Assembly 2

Figure 4.15 and Table 4.7 show the results for the CNN model trained and tested on assembly 2. The actual and predicted pressure plots follow the same pressure gradient patterns. The majority of the prediction error is seen on the front face of the cube. The normalized data had an average percent error of 11.18% with 94.29% of the prediction having a percent error less than 25%. The average percent error for the non-normalized data is 9.71% with 93.96% of predictions having errors below 25%. The RMSD for the normalized and

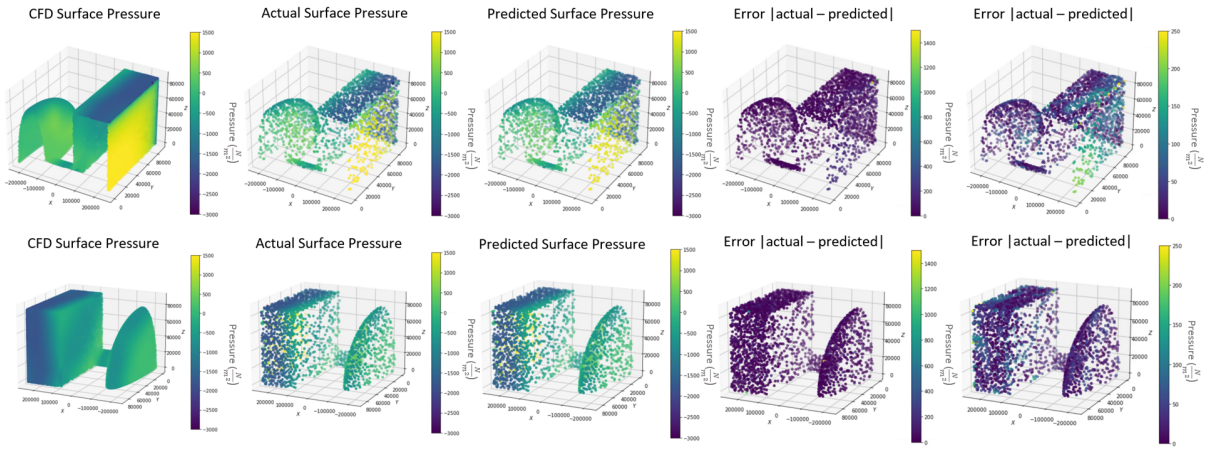

Figure 4.14: Train and Test on Assembly 1 Predictions

Table 4.6: Train and Test on Assembly 1 Results

	Normalized Data	Non-Normalized Data
Average Percent Error	16.64%	9.17 %
Percent of samples with errors below 25%	86.82%	94.56%
Percent of samples with errors above 25%	13.18%	5.44%
RMSD	0.079	72.16

non-normalized data was 0.046 and 42.07 respectively.

4.11.2.3 Train and Test on Assembly 3

Figure 4.16 and Table 4.8 show the results for the CNN model trained and tested on assembly 3. This assembly has the longest cylinder length. The actual and predicted surface pressure plots have no clear differences between them. The error in the predictions is seen toward the back corners of the cube. Low error is seen on the front surface of the cube, on the cylinder surface, and on the sphere surface. The normalized data had an average percent error of 20.98% with 84.40% of the prediction having a percent error less than 25%. The average percent error for the non-normalized data is 19.39% with 88.65% of predictions having errors below 25%. Additionally, the normalized data had an RMSD of 0.13 and the non-normalized data had an RMSD of 120.

4.11.2.4 Train on Assembly 1, Test on Assembly 2

To evaluate the generalizability of the model, the CNN was trained on assembly 1 and tested on assembly 2. The results are shown in Figure 4.17 and Table 4.9. The trained CNN showed no noticeable differences between the actual and predicted surface pressure plots. The majority of the errors in the model predictions

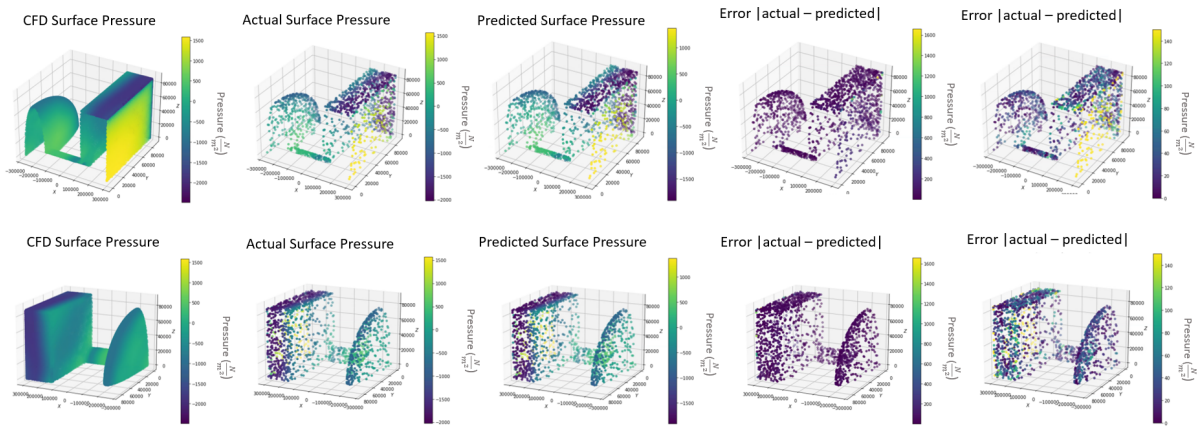

Figure 4.15: Train and Test on Assembly 2 Predictions

Table 4.7: Train and Test on Assembly 2 Results

	Normalized Data	Non-Normalized Data
Average Percent Error	11.18%	9.71 %
Percent of samples with errors below 25%	94.29%	93.96%
Percent of samples with errors above 25%	5.71%	6.04%
RMSD	0.046	42.07

are seen on the back half of the cylinder and front surface of the sphere. These are more reasonable results as the change in the length of the cylinder affects the fluid flow in these regions. The average predictive error for the normalized and non-normalized testing data was 147.43% and 70.45% respectively. 68.03% of the normalized testing points had percent errors less than 25%. and 73.75% of the non-normalized data had errors less than 25%. The normalized and non-normalized data had RMSD of 0.19 and 176.38 respectively.

4.11.2.5 Train on Assembly 1 and Assembly 2, Test on Assembly 1

A CNN model was trained on data points from assembly 1 and assembly 2 and tested on unseen data points from assembly 1. The goal is to assess the CNN model's performance when training on data from multiple assemblies. The results are shown in Figure 4.18 and Table 4.10. The only error seen in the predictions was localized to the front surface of the cube. The average percent error was 35.66% and 23.58% for the normalized and non-normalized data respectively. 90.66% of the normalized testing data points had prediction errors below 25% and 94.1% of the non-normalized testing data points had prediction errors below 25%. The normalized and non-normalized data had RMSD of 0.05 and 47.40 respectively. It can be concluded that training the CNN on multiple models does not decrease the CNN's performance when making predictions for just one of the assemblies that data points where sampled from for the training.

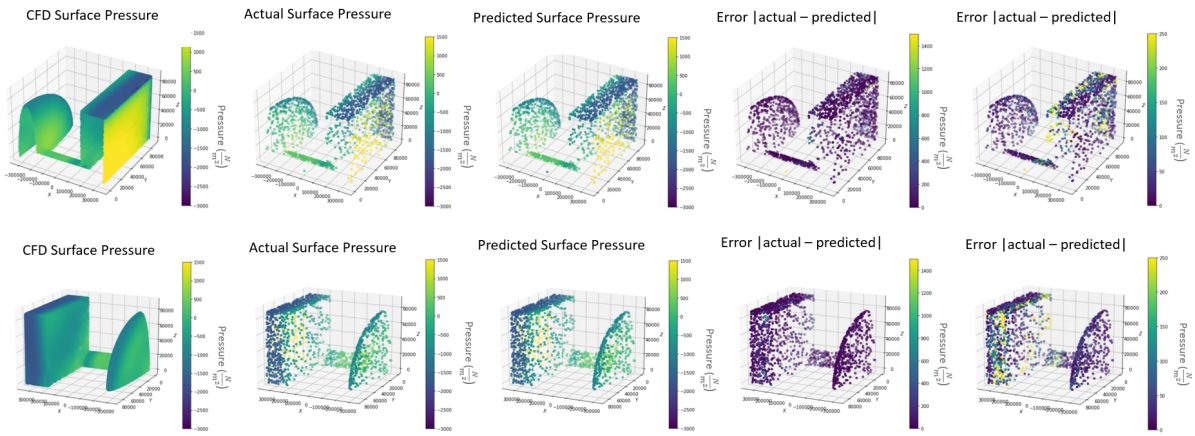

Figure 4.16: Train and Test on Assembly 3 Predictions

Table 4.8: Train and Test on Assembly 3 Results

	Normalized Data	Non-Normalized Data
Average Percent Error	20.98%	19.39%
Percent of samples with errors below 25%	84.40%	88.65%
Percent of samples with errors above 25%	15.60%	11.35%
RMSD	0.13	120

4.11.2.6 Train on Assembly 1 and Assembly 2, Test on Assembly 3

To further evaluate the generalizability of the model, the CNN was trained on samples from assembly 1 and assembly 2 and tested on data points from assembly 3. The results are shown in Figure 4.19 and Table 4.11. The actual and predicted surface pressure plots show the similar pressure values on each surface on the geometry. The highest prediction error is seen on the front surface of the sphere. There are also errors in the predictions on the cube. The average error in the prediction for the normalized and non-normalized data was 55.87% and 122.16% respectively. 68.25% of the normalize data had errors less than 25% and 76.91% of the non-normalized data had prediction errors less than 25%. The normalized data had a RMSD of 0.19 and the non-normalized data had a RMSD of 179.05.

4.11.3 Model Accuracy Analysis

The effect that the number of training samples has on the model accuracy was evaluated. Just over 12,000 probe volume samples were collected from assembly 2. The same CNN model was trained and tested using different train-test split ratios. The training sample size analysis results are shown in Figure 4.20. The CNNs trained with fewer data points had a higher prediction error than the CNNs trained with 6,000 or more data points. The error in the prediction accuracy decreases for CNNs trained with 0 to 6,000 data points. After

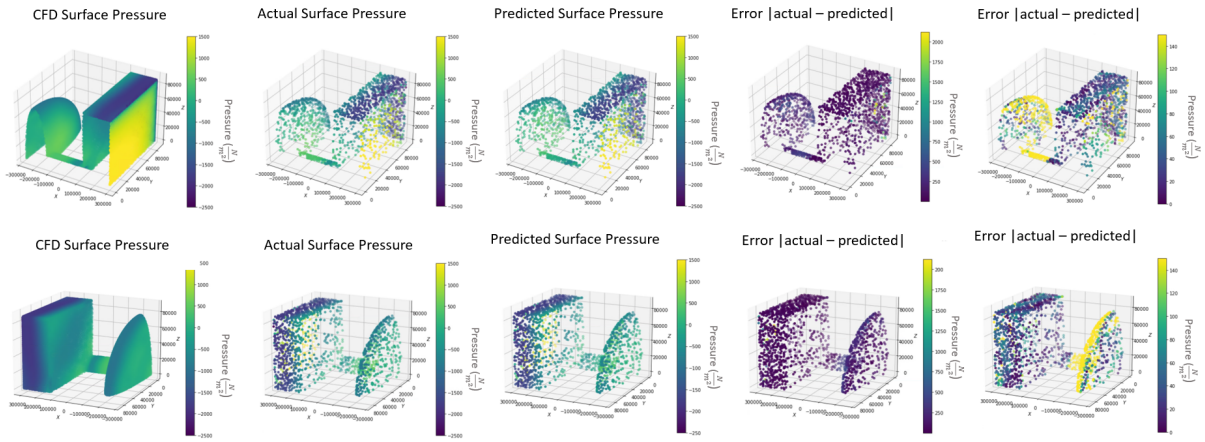

Figure 4.17: Train on Assembly 1, Test with Assembly 2 Predictions

Table 4.9: Train on Assembly 1, Test with Assembly 2 Results

	Normalized Data	Non-Normalized Data
Average Percent Error	147.43%	70.45%
Percent of samples with errors below 25%	68.03%	73.75%
Percent of samples with errors above 25%	31.97%	26.43%
RMSD	0.19	176.38

6,000 data points, the prediction error levels out. This indicates that 6,000 training samples is the optimal number of samples. The number of data points used to train the CNN model is directly proportional to the time to train the model. Using 6,000 training data points offers a balance between the time required to train the model and a lower prediction error.

4.11.4 Execution Time

The execution times for training and testing the CNN Surrogate are shown in Table 4.12. Training the CNN is the most time-consuming aspect of the surrogate. Training the CNN model with 6,000 data points takes approximately 129 minutes. This is a one-time cost as the model only needs to be trained once. Once trained, the model can make 6,000 surface pressure predictions in just under 3.5 minutes. Deriving the drag and lift forces requires surface pressure predictions at every mesh element included in the aggregation of CFD components for either a quarter section of the geometry or the whole geometry. Assembly 2 for example, contains approximately 98,000 mesh elements throughout the whole model and 24,000 mesh elements in a quarter section of the geometry. The trained CNN takes 56 minutes to make a prediction for every point in assembly 2 and approximately 14 minutes to predict every point in a quarter section of the geometry.

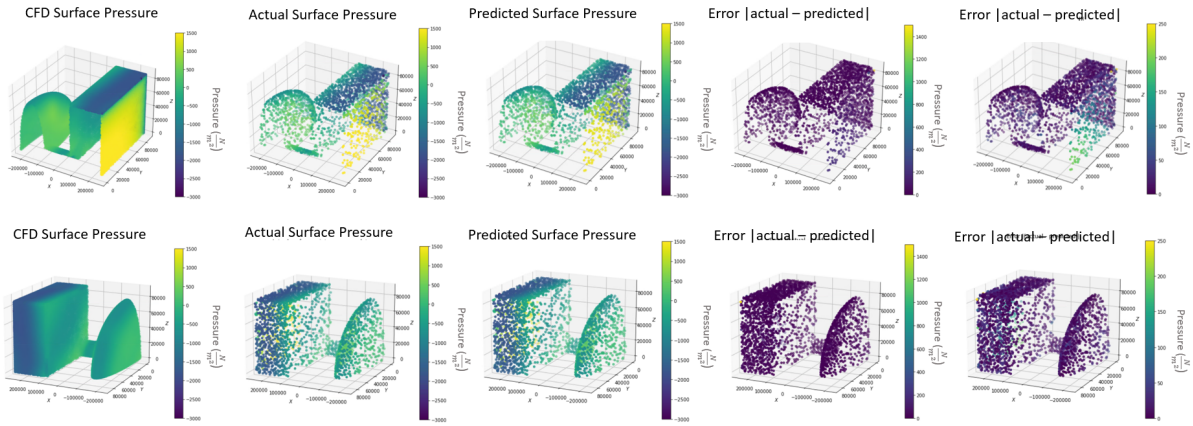

Figure 4.18: Train on Assembly 1 and Assembly 2, Test with Assembly 1 Predictions

Table 4.10: Train on Assembly 1 and Assembly 2, Test with Assembly 1 Results

	Normalized Data	Non-Normalized Data
Average Percent Error	35.66%	23.58%
Percent of samples with errors below 25%	90.66%	94.1%
Percent of samples with errors above 25%	9.34%	5.90%
RMSD	0.05	47.40

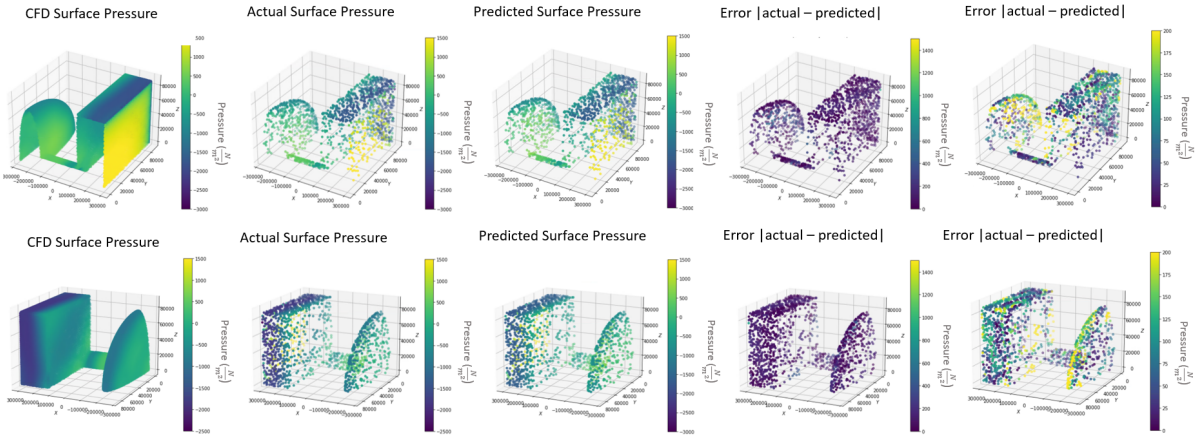

Figure 4.19: Train on Assembly 1 and Assembly 2, Test with Assembly 3 Predictions

Table 4.11: Train on Assembly 1 and Assembly 2, Test with Assembly 3 Results

	Normalized Data	Non-Normalized Data
Average Percent Error	55.87%	122.16%
Percent of samples with errors below 25%	68.25%	76.91%
Percent of samples with errors above 25%	31.75%	23.09%
RMSD	0.19	179.05

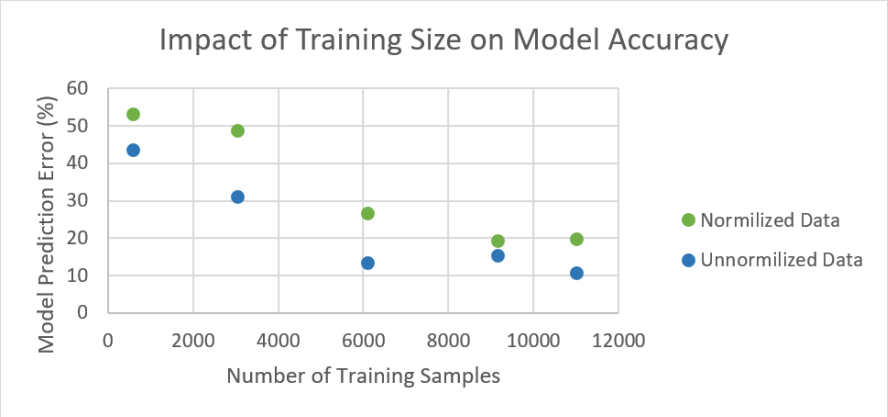

Figure 4.20: Training Size Analysis

Table 4.12: CNN Surrogate Execution Times

Task	Time to Complete (Minutes)
Train CNN with 6,000 Data Points	129
Predict 6,000 Data Points	3.43
Predict Data Points in Quarter Section	14.0
Predict Data Points in whole model	56.0

CHAPTER 5

Discussion

The two proposed CFD surrogates both have advantages and disadvantages that are explored in the following sections. Additionally, each method is usefully for different applications of CFD analysis in regards to parametric explorations of a design space given a corpus of components and seed designs. Each of the methods approach surrogate modeling from unique perspectives.

5.1 Polynomial Regression Surrogates

The results, limitations, and advantages of the polynomial regression surrogates are discussed in this section.

5.1.1 Results

The polynomial regression model had the highest performance when predicting the drag force with an R-squared score of 0.992 for the correlation between the actual and predicted values. This model also performed well when predicting the lift force with an R-squared score of 0.912. The moment prediction had the highest variance in predictions and the lowest accuracy with an R-squared score of 0.562. The prediction performances for each of the dependent variables indicate that the polynomial regression surrogate model can make accurate drag and lift predictions for a single parametric model under exploration. The moment value predictions, while less accurate, are still useful for a fast estimate of the system.

5.1.2 Limitations

The largest limitation was in collecting data for the polynomial regression machine learning model. This limitation is not related to the polynomial regression machine learning approach, but rather a shortcoming of the automated analysis tool. Traditionally, the use of CFD analysis software is performed manually, where mesh generation issues are resolved by the software user. Data is collected by running the PET with the desired parameters for ideally 1000+ iterations. The two errors encountered when doing this were either the mesh locking or the mesh failing to generate. The mesh locking result in no progress in the meshing process even when allowing the meshing to run for multiple days. If this happens, the PET run must be manually aborted and restarted. The mesh fails when the geometry is oriented in a manner that causes one or more components to fail to mesh. When this happens, the analysis does not produce any drag, lift, and moment values, but the PET will keep running. Collecting input data for machine learning requires a full CFD analysis for each variation of the model making it an inherently time-consuming process even when errors are

not encountered. The polynomial regression model knows the dimensions and rotation of each component in the system, but does not have a sense of how the components are oriented relative to each other. This means that the polynomial regression surrogate will not perform well on configurations of the geometry that were not seen in the training data. Additionally, the input features are tailored to the number of dimensions and rotation parameters applicable to the model under test. This makes generalizing to geometries with more or less parameters impractical.

5.1.3 Advantages

The time to train the polynomial regression surrogate is relatively quick. The training time is proportional to the number of training samples. Only a small number of training samples are needed compared to the large datasets required for deep learning models. Once the surrogate is trained, the model can make reasonably accurate drag, lift, and moment predictions extremely fast.

5.2 CNN Surrogates

The results, limitations, and advantages of the CNN surrogates are discussed in this section.

5.2.1 Results

The results of the CNN surrogates are discussed in this section.

5.2.1.1 CNN Without Masking Layer vs. With Masking Layer

The original approach to formatting the input data to the CNN did not include a masking channel. While this model did perform well if given enough training data, there was no clear distinction between the fluid and the solid regions. The solid regions of the probe volume were assigned pixel values of zero, and the pixels in the fluid regions were assigned the respective pressure values from the CFD meshes. This approach distinguishes between the fluid and solid regions as long as the pressure values are not zero or close to zero. Pressures could be close to zero in some cases; and thus, using pressure to identify solid geometry is error prone. It is important to distinguish between the solid and fluid regions because the goal is to only analyze the fluid flow around the geometry; and therefore, the internal structures of the geometry are irrelevant. When performing fluid analysis for this application, the pressure values within the solid region are ignored. Without this distinction being made in the input images there could be unrealistic flow patterns learned by the CNN.

5.2.1.2 Training and Testing on the Same Assembly

Training and testing the CNN model on points sampled from the same assemblies showed good surface pressure predictions. The average percent error ranged from 11.18% to 20.98% on the normalized data for

each of the three assemblies. The normalized RMSD ranged from as low as 0.046 to 0.13. This indicates that the CNN is able to learn the surface pressure values at a single point on the geometry from the pressure probe volumes of each competent in the assembly.

5.2.1.3 Training and Testing on Different Assemblies

The CNN model was trained on points sampled from assembly 1 and tested on points sampled from assembly 2. This is a higher error then seen when training and testing on points from the same assembly. The average error for the normalized and non-normalized data were 147.43% and 70.45% higher respectively than the average percent errors seen from the CNN model trained and tested on assembly 2. This indicates that training the CNN on one assembly may not be sufficient to generalize to unseen data. Next the CNN was trained on points sampled from assembly 1 and assembly 2 and tested on points from assembly 1. This CNN model had an average percent error of 35.66% on the non-normalized data. This indicates that including samples from the same assembly that the testing data was collected from in the training set improves the prediction performance. This additionally indicates that that a model trained on multiple assemblies can still perform well when making predictions on just one of the assemblies seen in the training data set. Finally, to test the generalizability, the CNN was trained on data points from assembly 1 and assembly 2 and tested on assembly 3. This model had an average percent error of 55.87% and RMSD of 0.19 on the normalized data. This is a slightly higher prediction accuracy than the model trained on assembly 1 and assembly 2 and tested on assembly 1. This CNN model is making prediction on unseen data, so it was reasonable to see a higher error in the predictions compared to a model making prediction that had a samples drawn from the same assembly in the training and testing set.

5.2.2 Model Accuracy

The number of data points used to train the CNNs has a large influence over the prediction accuracy of the models. An inadequate number of training points may not provide a sufficient representation of the pressure interaction between components. Additionally, more complex geometries or more complex flow conditions will likely require more training points to achieve high prediction accuracies. Using too many training points, on the other hand, will increase the time to train the models. Determining the optimal number of training points will produce a CNN model with a high prediction accuracy while still having a relatively low time to train the CNN model. The ideal number of training points for assemblies 1, 2, and 3 was found to be approximately 6,000 data points. Training a CNN model with 6,000 points takes approximately 2 hours. The model, however, only needs to be trained once. The trained model can make 6,000 surface point predictions in just under 3.5 minutes.

5.2.3 Execution Time

The full CFD execution times are shown in Table 4.3 and the CNN surrogate time are shown in Table 4.12. Training the CNN model with 6,000 points takes 129 minutes, which is less than the time to perform a full CFD analysis on a single assembly model. The model only needs to be trained once, so this is a one-time cost. After the model is trained, the surrogate can make predictions for the whole model and quarter section of the model in 56 minutes and 14 minutes respectively. Given a pre-computer set of component the surrogate model is approximately 3.5 times faster than the average time to perform CFD analysis on each of the three assemblies.

5.2.4 Challenges

Challenges encountered while implementing the CNN surrogates are discussed in this section.

5.2.4.1 Probe Volumes

The grid points in the generated CFD mesh are not evenly distributed. It is common to see the grid points close together near the geometry where the fluid flow undergoes the majority of changes such as turbulent eddies, detachment points, boundary layer formation, and potentially strong coupling. The mesh grid points farther from the model usually have larger spacing because these regions mostly experience slow changes in pressure and velocity. The non-uniform spacing poses a challenge when integrating machine learning with CNNs. CNNs require a constant number of input pixels. This can be achieved through two approaches. The first and most common is creating an evenly spaced Cartesian grid. This is the simplest approach, but comes with some known drawbacks. The first being that the decrease in the precision of the CFD representation is a result of interpolation or extrapolation. In order to capture a relatively accurate representation of the CFD domain, the evenly spaced grid must be extremely refined uniformly. This will undoubtedly increase the computational time. On the other hand, a coarser grid is computationally faster, but is unable to capture the curved surfaces of a geometry (Kashefi et al., 2021).

5.2.4.2 Data Storage

The data files exported from Ansys Fluent were too large to import into the Python environment directly. As a work around, a database was used to store all of the data. This allowed the data to be saved in database files for future use. Using databases in Python requires a cursor to execute all database operations, which is known to be slower then executing the commands directly. Each of the assembly and component models has data files, generated by Fluent containing an entry for every mesh element. The number of mesh elements in each model was roughly between 10 million to 20 million elements. The cursor, along with the large number of

data points made the traversal of the database extremely slow. Performing one iteration of the probe volume generation algorithm would take up to 24 hours. In order to make this a usable algorithm, the iteration time needed to be decreased significantly. To achieve this, indexes were created for the tables on the x, y, and z coordinates to speed up table searches. Additionally, temporary tables were created to store a subset of the mesh points with only the relevant values. This decreased the table search time by creating a smaller search space. The CFD fluid domain for each part extends further than necessary for this application. The large fluid domain is used to fully capture the fluid flow as standard practice in CFD applications. For predicting only the surface values, regions of the fluid domain far away from the component would never be used to generate probe volumes. For this reason, the outer regions of the domains were removed from the search space. To reduce the table sizes even further, quarter sections of the models were used to collect samples from the model. The algorithms to generate the probe volumes were also optimized. After these steps, approximately 2,000 samples could be produced in 17 hours. Likewise, collecting approximately 12,000 samples took around 3.5 days. Further algorithm optimization could be performed to further reduce the time to create probe volumes.

5.2.5 Limitations and Advantages

The limitations to the probe volume approach are discussed in this section.

5.2.5.1 Predictions

In order to make predictions, a masking layer and probe volumes must be generated relative to the surface point of interest for each part in the assembly. Generating probe volumes requires performing a complete CFD analysis for each part in the assembly. Performing CFD analysis on a single part should take less time than performing CFD analysis on the whole assembly due to the reduced complexity.

5.2.5.2 Unknown Number of Assembly Parts

One application for predicting the surface pressure might be to evaluate the effect that adding or removing a particular component has on the drag and lift force. The CNN model is currently set up to use input images of size $n \times 16 \times 16 \times 16 \times 4$. The four image channels are used to store the cube, cylinder, sphere probe volume images and the masking image. The CNN model only accepts images with these dimensions, which limits the geometries that can be used to ones comprised of three parts. One approach to enable use of geometries with more or less than three parts is to modify the input layer and convolution layer dimensions of the neural network. For example, increasing the number of channels in the input layer from four to six would allow training and testing on geometries containing five parts. The downside is that the new model

would now be limited to geometries containing five parts. Another work around would be to specify the CNN input dimension to maximum number of parts in the assembly. To evaluate the effect of removing a part in the assembly, a probe volume containing all zero values could be used as a place holder for the missing part. Additionally, the masking layer would need to reflect the solid region without the removed part. The drawback to this approach is the unnecessarily high dimensionality which will come at a computational cost. This is especially true from geometries containing a high number of parts.

5.2.5.3 Models with Large Numbers of Parts

The more parts in a model the more layers (e.g. the more 16x16x16 images) in the input image. An accurate model requires thousands of input data points. This increases the time to train the CNN model and make predictions. One approach would be to lump parts together. For example, if the goal was to determine the optimal fin placement on an underwater vehicle, then a simplified representation of the model could be used to reduce it to only two parts. All parts in the underwater vehicle assembly excluding the fin could be lumped together and represented as one part. This would leave the fin to be the other part used to generate probe volumes. Since the goal is to only modify the position of the fin, it would be the only part in the assembly under parametric exploration.

5.2.6 Probe Volume Approach Advantages

The polynomial regression model used with the Remus data was restricted to a single assembly configuration. The regression model was learning the effect of changing the dimensions, but has no knowledge of the orientation of the dimensions or components relative to each other. The probe volumes approach, however, uses a 3D image rendering of sections of the model to avoid this problem. Instead of learning the dimensions, the deep learning CNN model learns the patterns of pressure contours in different regions of the model. This provided a more robust surrogate model.

CHAPTER 6

Conclusion

The conclusions for polynomial regression surrogate and the CNN surrogate are presented in this section.

6.1 Polynomial Regression Surrogate

This paper demonstrates that a polynomial regression surrogate models can accurately predict the drag, lift, and moment values for parametric exploration of a single seed model. Training the polynomial regression model requires pre-computed CFD analysis for hundreds of permutations of the seed model. This approach would be useful when analysis is required for thousands of design points. After CFD analysis has been performed for a couple hundred design points, the polynomial regression surrogate can then be trained and employed for analysis on the remaining design space. The approach, however, is sub-optimal for parametric exploration of a design space in which the corpus of components that the model under analysis is composed of is not fixed. For example, if the goal of parametric exploration is to analyze the changes in drag, lift, and moment of using different propellers on a UUV, the dimensions alone may not be sufficient for describing differences between these complex components. The approach, however, is sub-optimal for parametric exploration of a design space in which point designs have varying number of components.

6.2 CNN Surrogate

This paper demonstrates that surrogates can be developed given pre-computed component solutions and system specifications to compute a fast estimate of the system model. The surrogate is composed of the CFD probe volumes and CNN generated algorithm to predict the surface pressure values on seen geometries with 91.72% of the points in the testing set having prediction errors less than 25% and 75.53% of the points in the testing set having prediction errors less than 25% on unseen geometries.

6.2.1 Applications

The surrogate for surface pressure prediction does not replace the need for CFD analysis, but instead greatly reduces the number of times CFD analysis is needed during the parametric exploration process. Previously, CFD analysis was performed for each parametric variation of the geometry. The number of times CFD analysis is performed is minimized with the use of the surrogate model. CFD analysis is initially performed on each part in the geometry and the assembly of components. These sets of CFD grids are used to train and test the models. Once the CNN model has been trained sufficiently, the surrogate model can be used to

quickly estimate the surface pressure on the model. The drag and lift forces can be derived from the predicted surface pressure values, providing a quick approximation of these forces on the model. At this point, the top parametric variations for optimum drag and lift force can be selected and additional CFD analysis can be performed if more precise values are needed. This approach cuts down on the size of the design space from performing CFD analysis on hundreds to thousands of models to only performing CFD analysis on a few of the top performing geometries determined by the surrogate model.

6.2.2 Surrogate vs CFD Execution Times

The number of points in the training dataset for the CNN influences the accuracy of the model and the time to train the model. Training a CNN with 6,000 training data points takes approximately 2 hours. The model, however, only needs to be trained once. The surrogate takes approximate 3.5 minutes to make 6,000 surface point predictions. Performing a full CFD analysis on one of the assemblies takes approximately 3.25 hours. Figure 6.1 shows the time to evaluate 100 design points in the parametric exploration space using the OpenMeta CFD test bench to perform the full CFD solution compared to using the surrogate model. Performing the full CFD analysis on 100 geometric permutations of the assembly took considerably longer than using the surrogate. The OpenMeta CFD test bench has automated the CAD generation and CFD analysis processes making it considerably faster than manually performing CFD analysis for each of the 100 models. The surrogate model has an initial startup time, which is represented by a non-zero time for zeroth design points. The startup time includes the time to run the prerequisite CFD analyses (each component separately and at least one assembly) and to train the CNN surrogate model. The time to generate the input probe volumes is not included in the startup time. Generating thousands of probe volumes can take hours to days, however, there is considerable room for improvement in the computational efficiency of these algorithms. The focus of this work was not to optimize the algorithms involved in generating the probe volumes. For that reason, the probe volume generation time was not included in the startup time for the CNN surrogate model. The full CFD solution does not have any initial overhead and thus the zeroth design points takes zero time. The CNN surrogate for the full and quarter models shows the time to predict surface pressure values for every point in the original CFD mesh and a quarter section of the CFD mesh. For example, assembly 2 contains approximately 98,000 surface points across the whole model, and approximately 24,000 surface points in a quarter section of the model. After approximately 10 design points, the surrogate model becomes more computationally efficient than the OpenMeta CFD test bench. Assuming CFD component solutions have been pre-computed, the surrogate model demonstrated a 3.5 times decrease in execution time compared to the full CFD analysis.

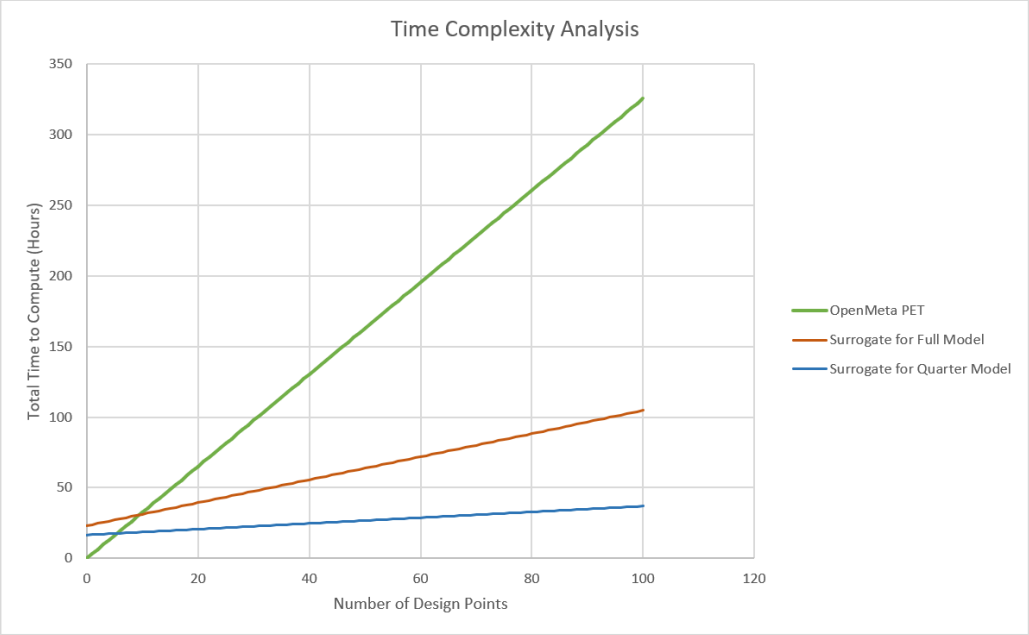

Figure 6.1: OpenMeta Vs. Surrogate Time Complexity

CHAPTER 7

Future Work

This section describes several areas where future work would potentially make significant improvements in the overall approach.

7.1 Improving Model Prediction Accuracy

The CNN model trained on points from assembly 1 and assembly 2 and tested on points from assembly 3 showed that 76.91% of the training points had prediction errors less than 25%. The average percent error between the actual and predicted values for all of the testing data was 122.16% on the non-normalized dataset. This indicates that the 23.09% of the points in the testing set with percent errors greater than 25% are much greater than 25%. To further improve the prediction results on unseen geometries, additional variations of the cube, cylinder, and sphere assembly should be added. It is likely that training a CNN model on three or more assembly variations would decrease the overall prediction error on unseen data. Ideally more than 90% of the testing data would have prediction errors less than 25%.

7.2 Testing on Arbitrary Shapes

The proposed CNN model and probe volume approach made reasonable surface pressure value predictions on an unseen model permutation after being trained on data points sampled from two model variations. Next steps for this project include testing the CNN trained on permutations of one model and testing it on either a completely different model or a model composed of the same parts, but in a new configuration. These set of tests will indicate the extent of the training data needed for the CNN to be able to generalize on unseen complex geometries. The scalability of this approach is discussed in Sections 5.2.5.2 and 5.2.5.3.

7.3 Testing on Range Velocities

The CFD analysis for assembly 1, assembly 2, and assembly 3 was performed with a constant inlet velocity of 1.5 m/s. The inlet velocity was held constant to simplify fluid flow conditions for the inputs to the machine learning model. The next steps include testing the CNN's ability to learn over a range of velocities. This would possibly require input data sampled from CFD models with varying inlet velocities. Introducing different inlet velocities adds a level of complexity to the flow regime. Velocity is directly proportional to the Reynolds number. Low Reynolds numbers are characterized by laminar flow, whereas, higher Reynolds numbers indicate turbulent flow. Turbulent flow undergoes irregular fluctuations and eddies, compared to

laminar which is more controlled and predictable. Velocity and pressure are inversely proportional to each other, so an element of the velocity is represented in the pressure data. The goal of testing the CNN on a range of inlet velocities is to identify additional limitations of the current CNN model and evaluate methods for improving the current approach.

7.4 Drag and Lift Forces

The main application for predicting the surface pressure of the model geometry is to derive the drag and lift forces from the predictions. Drag force is essentially the sum of all forces in the direction parallel to the inlet flow. Likewise, the lift force is the sum of forces normal to the inlet flow. For the assembly models, the drag and lift forces are the sum of forces in the x-direction and y-direction respectively. The pressure surface values indicate the force distributed over an area of the model. One possible approach to obtain the drag and lift forces from the surface pressure values would be to format the predicted surface pressure values into a file compatible to Ansys Fluent's CFD Post post-processing software. CFD Post computes the drag and lift forces in the function calculator tab. To calculate the drag and lift forces, the exterior wall of the model geometry is selected as the location. Next, the variable should be set to pressure and either x or y should be selected for the direction depending on which force calculation is desired. The CFD Post software would then be able to compute the drag and lift values. This would essentially cut out the time-consuming geometry set up, mesh generation, and Fluent solver steps of the traditional CFD analysis process. Automating the file formation and CFD Post drag and lift calculation would allow hundreds of geometric variations of the same model in order to find the optimal model configuration. Another approach to computing drag, moment, and lift would be to develop a program that performed the computations as described in Section 4.2.4.

7.5 Modifying the Window Size

One possible approach to improve the CNN's prediction accuracy would be to increase the probe volume size. Increasing the probe volume size would give insight into the other parts in the assembly around the surface point of interest. In the proposed approach, the probe volume sizes are small relative to the geometry. Generally, each probe volume only contains the solid region of one part in the model. The exception being surface points that are close to or on the intersection between parts. Using a small probe volume size relies primarily on pressure values of each part and does not provide the CNN with additional insight to where each part in the model is located relative to each other.

References

Ameri, A., Akhaee, M. A., Scheme, E., and Englehart, K. (2019). Regression convolutional neural network for improved simultaneous emg control. *Journal of neural engineering*, 16(3):036015.

Anderson, J. D. (1992). Governing equations of fluid dynamics. In *Computational fluid dynamics*, pages 15–51. Springer.

Bapty, T., Whittington, S., Walker, J., Hite, J., Swenson, B., Owens, K., Eisele, F., Scott, J., and Owens, R. (2022). Design oracle for ai-based cps design. In *2022 IEEE Workshop on Design Automation for CPS and IoT (DESTION)*.

Brunton, S. L., Noack, B. R., and Koumoutsakos, P. (2020). Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52(1):477–508.

Jabbar, H. and Khan, R. Z. (2015). Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). *Computer Science, Communication and Instrumentation Devices*, 70.

Kashefi, A., Rempe, D., and Guibas, L. J. (2021). A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries. *Physics of Fluids*, 33(2):027104.

Li, Z., Liu, F., Yang, W., Peng, S., and Zhou, J. (2021). A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21.

Moureau, V., Domingo, P., and Vervisch, L. (2011). Design of a massively parallel cfd code for complex geometries. *Comptes Rendus Mécanique*, 339(2-3):141–148.

O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *CoRR*, abs/1511.08458.

Ostertagová, E. (2012). Modelling using polynomial regression. *Procedia Engineering*, 48:500–506.

Prestero, T. T. J. (2001). *Verification of a six-degree of freedom simulation model for the REMUS autonomous underwater vehicle*. PhD thesis, Massachusetts institute of technology.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660.

Thuerey, N., Weißenow, K., Prantl, L., and Hu, X. (2020). Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal*, 58(1):25–36.

Wilding, M. (2022). Darpa: Symbiotic design for cyber physical systems.

Zhao, Y., Akolekar, H. D., Weatheritt, J., Michelassi, V., and Sandberg, R. D. (2020). Rans turbulence model development using cfd-driven machine learning. *Journal of Computational Physics*, 411:109413.