Predicting Volume of Fluid Interfaces with Neural Networks

by

Pranav Rajesh Pawar

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2023 by the
Graduate Supervisory Committee:

Marcus Herrmann, Co-Chair
Houlong Zhuang, Co-Chair
Huei-Ping Huang

ARIZONA STATE UNIVERSITY

August 2023

ABSTRACT

Computing the fluid phase interfaces in multiphase flow is a challenging area of research in fluids. The Volume of Fluid and Level Set methods are a few algorithms that have been developed for reconstructing the multiphase fluid flow interfaces. The thesis work focuses on exploring the ability of neural networks to reconstruct the multiphase fluid flow interfaces using a data-driven approach.

The neural network model has liquid volume fraction stencils as an input, and it predicts the radius of the circle as an output of the network which represents a phase interface separating two immiscible fluids inside a fluid domain. The liquid volume fraction stencils are generated for randomly varying circle radii within a 1x1 domain using an open-source VOFI library. These datasets are used to train the neural network. Once the model is trained, the predicted circular phase interface from the neural network output is used to generate back the predicted liquid volume fraction stencils.

Error norms values are calculated to assess the error in the neural network model's predicted liquid volume fraction stencils with the actual liquid volume fraction stencils from the VOFI library. The neural network parameters are optimized by testing them for different hyper-parameters to reduce the error norms. So as to minimize the difference between the predicted and the actual liquid volume fraction stencils and errors in reconstructing the fluid phase interface geometry.

ACKNOWLEDGEMENTS

The past two years as a student at Arizona State University have been challenging and rewarding. I am grateful to all the people who have been a part of my graduate school. First, I would like to thank my advisor, Dr. Marcus Herrmann, for trusting me to take me as his student in Fall 2022. Leading me on how to articulate research and optimistically address hurdles. I would also like to thank Dr. Houlong Zhuang for his valuable advice and for agreeing to be a part of this research.

I am happy to share the research platform with the Multiphase Research lab colleagues and thank them for their time. Lastly, I am thankful and lucky to have my family, specifically my grandparents, and friends in my life with their constant love, support, and encouragement. Thank you for everything.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

1.1 Overview & Challenges

Multiphase flow find applications like spray atomization (Sirignano (1999)), bubbly flows (Sato *et al.* (1981), Clift *et al.* (2005)), breaking waves (Melville (1996)) and rain formation (Shaw (2003)). To simulate such two-phase flows, it is crucial to understand the physics involved in liquid/gas flows and the complex phase interface tracking and capturing (Mirjalili *et al.* (2017)) methods. There is a vast amount of literature available on interface-capturing methodologies.

Exact analytical solutions exist for the simplest fluid flow problems. For example, in the flow around a sphere, the analytical result is limited to low and high Reynolds number (Prosperetti and Tryggvason (2007)). Experimental studies strive on measuring the properties of the two-phase flow phenomenon accurately. It becomes challenging when the flow length scales are small, time scales are short. In such scenarios, effective controlling of experiment parameters to collect data relating to droplet characterization and breakup is difficult (Prosperetti and Tryggvason (2007)). Moreover other than single-phase flow, creating experiments inside the laboratory for multiphase systems is also difficult.

With these challenges in mind, the need for numerical methods is explored by the multiphase research community to solve governing equations for actual fluid flow problems. To create methods that will consider the crucial fluid flow physics and are computationally cost-effective to solve. Thus aid in developing accurate and physics-representative simulations in interface-capturing.

Implicit phase interfaces are represented with numerical methods like Volume of Fluid (VoF) or Level Set methods for highly complex, evolving, and corrugated interface shapes (Popinet (2009)). However, Mirjalili *et al.* (2017) stated challenges in numerical methods. Such as - (1) Accurate representation of surface tension on the phase interface, (2) Model large discontinuous properties at the phase interface like density jumps (3) Implementing fundamental mass, momentum, and kinetic energy conservation principles. These can cause numerical instabilities, within incompressible limits, because of discrepancies in interface transport and momentum transport (Herrmann (2008)).

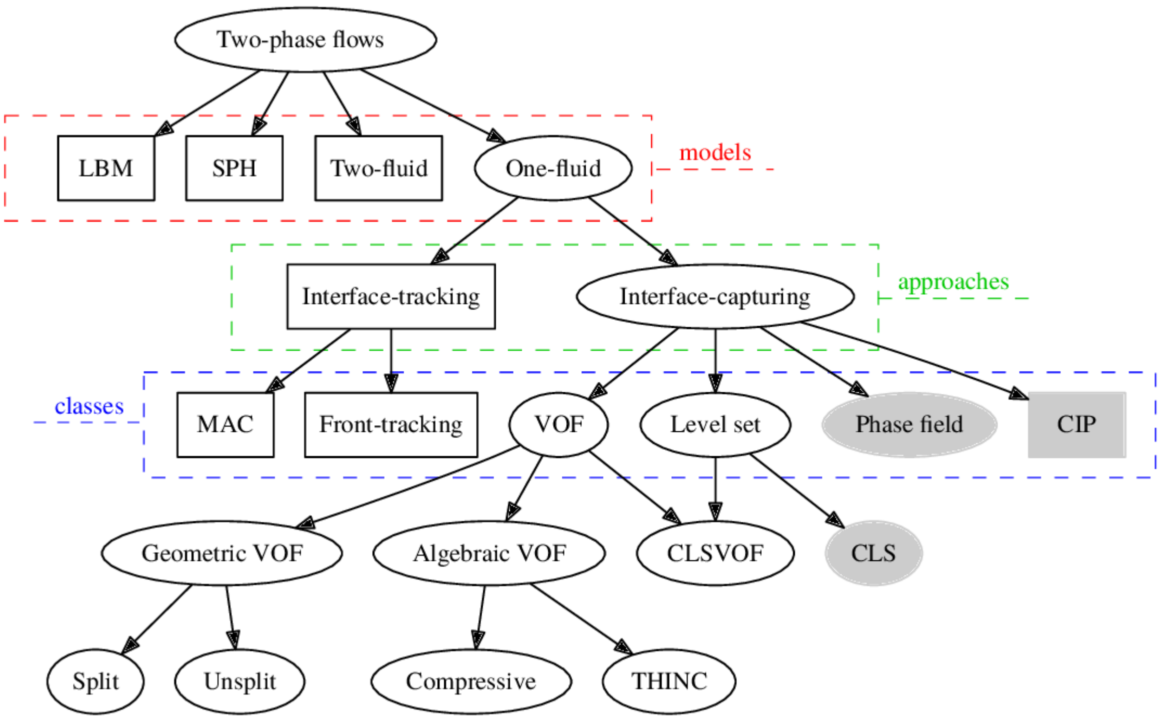


Figure 1.1: Numerical Methods for Two-Phase Flows (Mirjalili *et al.* (2017))

Figure 1.1 shows different methods researched by the multiphase community for modeling two-phase flow. The current thesis work uses the VOF method to generate liquid volume fractions inside a computational cell. Further, for the scope of this thesis research, the work is more focused on reconstructing the fluid phase interface for 2D circular droplets in two-phase immiscible flow.

1.2 Motivation & Scope

The numerical schemes like level set methods and volume of fluid (VOF) approximate the phase interface. The VOF scheme proceeds in two steps. First is the reconstruction of the interface and second is the advection of the reconstructed interface. It offers good characteristics like implicit formulation of topology changes, preserves mass, and can be scaled to 3D Cartesian geometry easily. The VOF-PLIC reconstruction approximates the line phase interface in 2D or planar phase interface in 3D.

The motivation for using a neural network is to test its ability to locally reconstruct the phase interface when it is trained on a dataset of 3x3 liquid volume fraction stencils. Further, the network is tuned for minimal training loss while testing it for different network architectures and parameters.

The current research work evaluates L_∞, L_1, and L_2 error norms on liquid volume fraction stencils. The performance of the neural network is evaluated by using L_∞, L_1, and L_2 error norms as a metric to assess how well it locally reconstructs the phase interface.

1.3 Thesis Outline

In this thesis, synthetic data is generated for circles by initializing circles with centers randomly with varying radii within a 1x1 computational domain. Next, different neural network architectures mainly dense neural network (DNN) and convolutional neural network (CNN) are explored with static and dynamic learning rates to assess how well the neural network can locally reconstruct the geometry of fluid phase interface. The current work tests the neural network architecture for different network parameters with the Keras library in Python. The parameters from the hyper-parameter space are selected such that the mean absolute loss is minimum. Thereby, the L_∞, L_1, and L_2 error norms in the predicted values are reduced for reconstruction of the fluid phase interface.

The thesis has three parts - (1) Generating VoF interfaces from an open source VOFI library (2) Integrating this data into a neural network to give radius prediction by neural network model and (3) Analyzing and comparing the two liquid volume fraction stencils, one from the open source VOFI library and the second from the neural network, thereby developing an error metric to assess network performance relative to VoF field values in stencils.

Chapter 2

VOLUME OF FLUID METHOD

2.1 Phase Interface

Different numerical algorithms are present in the literature to capture interfaces in two-phase flow. The volume-of-fluid (VoF) method is the most widely used method employed on fixed grid solvers. This chapter explores the volume of fluid method used to reconstruct the geometry of fluid interfaces.

An interface is a thin line in 2D or a plane in 3D that separates two fluid phases. Accurate locating of fluid interfaces has been always a challenging task in the multi-phase flow research field.

The volumetric formulation is explored on a larger scale in the current literature. This is because they achieve numerical stability in discretely balancing surface tension and pressure gradient terms, which is crucial as stated by Francois *et al.* (2006).

2.2 Governing Equations

All of the interface capturing methods, follow the fundamental momentum equation, coupled with local density, viscosity, and surface force terms. In sharp interface methods, these values have a jump across the interface. Compared to diffuse-interface methods, these scalars are functions of the local phase (Mirjalili *et al.* (2017)).

The phase interface moves causing phase change inside the cell, mathematically represented by the following advection equation –

$$\frac{DH}{Dt} = \frac{\partial H}{\partial t} + \mathbf{u} \cdot \nabla H = 0 \tag{2.1}$$

where H is the marker function for incompressible flow to identify different fluids, and u is the velocity of the fluid.

The Heaviside (step) function H is for fluid i at location x. Mathematically,

$$H_i(\mathbf{x}) = \begin{cases} 0, & if\ \mathbf{x} \text{ is reference fluid} \\ 1, & if\ \mathbf{x} \text{ is other fluid} \end{cases} \tag{2.2}$$

While following the flow, the approximations of H should be calculated. To represent several approximations of H, the color function C is defined as the average of H in each computational cell,

$$C_{i,j} = \frac{1}{\Delta x \Delta y} \int_V H(x,y)\ \mathrm{dxdy} = 0 \tag{2.3}$$

Once we integrate the equation 2.1 for a computational cell (i,j) in a 2D Cartesian grid of width h, the definition of color function or fractional value of the cell is,

$$h^2 \frac{\partial C_{i,j}(t)}{\partial t} + \int_\Gamma \mathbf{u} \cdot \mathbf{n} H(\mathbf{x},t)\, \mathrm{d}l = 0 \tag{2.4}$$

where Γ is cell boundary and $\mathbf{n}$ is unit normal to the phase interface.

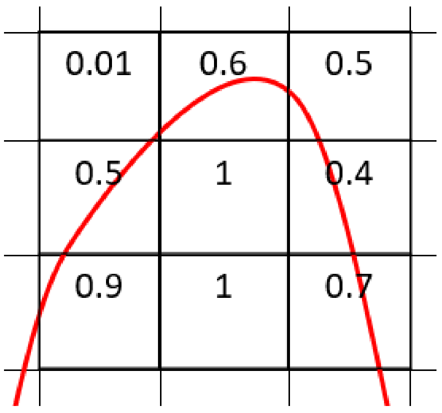


Figure 2.1: Multiphase Flow Fluid Interface in Structure Grid

Cell is full for $C = 1$ or cell is empty for $C = 0$. If the value of C is between 0 & 1, then the phase interface cuts the cell at a certain proportion. Figure 2.1 represents how fractional values $C(i,j)$ are computed inside the cell when a phase interface is present, indicating the amount of fluid present inside the cell.

2.3 Numerical Methods for Capturing Phase Interface

In the volume of fluid, accurate phase interface reconstruction is crucial to capture the spatial variation of two-phase flow. Extension of advection of marker function to capture phase interface was first done by Noh and Woodward (1976) with the method called as simple line interface calculation (SLIC). In this, the advection of fluid is done in one coordinate direction and next in another for 2D dimension flow. Hirt and Nichols (1981) introduced a slightly different method than SLIC. They chose on calculating the advection in one direction by observing the normal if it aligned with which coordinate direction (either vertical or horizontal). In the initial research period, the fluid phase interface geometry in 2D was still reconstructed as parallel lines.

2.3.1 Piecewise Linear Interface Construction (PLIC)

Gueyffier *et al.* (1999) presented the PLIC (Piecewise Linear Interface Construction) algorithm. It calculates the direction of normal to the interface. Then inside a cell where the phase interface cuts the cell, a fractional number is represented by a volume of fluid (VOF) ($0 < C(i,j) < 1$). Here it assumes the interface as a planar for 3D or a line for 2D case and orientation is selected arbitrarily with a coordinate axis. Then for phase interface reconstruction, it finds a normal representation of the phase interface with the help of fraction field values C of adjacent cells. In summary, collectively with the volume fraction value C and with the normal the location of the interface is determined.

Figure 2.2 represents an actual fluid phase interface geometry in comparison with the fluid phase interface geometry reconstructed by the PLIC algorithm. The area under the PLIC constructed line (yellow-shaded region) is equal to the area above the PLIC constructed line. Even though the circular arc is represented by a line, PLIC conserves volume by representing the total volume within the PLIC reconstructed interface geometry equal to the total volume of fluid inside of the actual phase interface geometry.

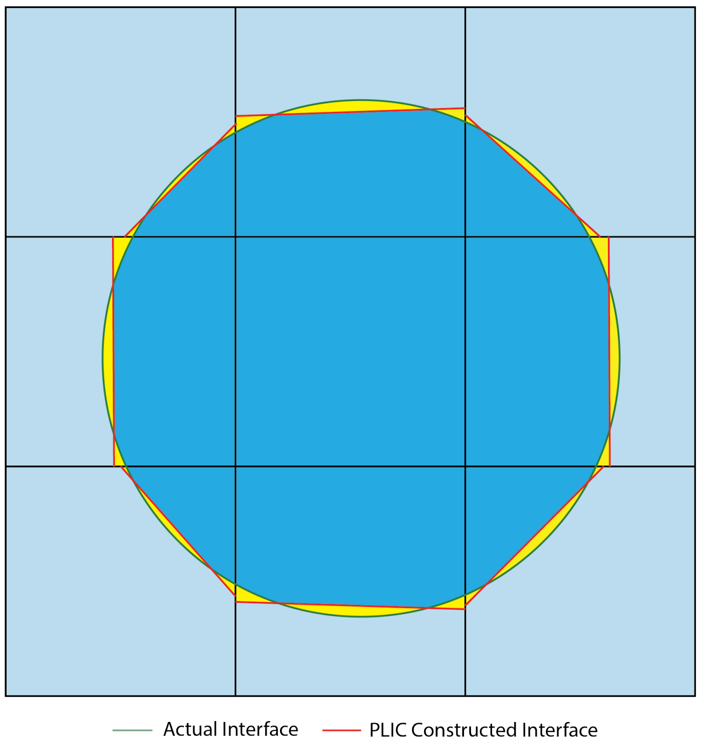


Figure 2.2: 3x3 Volume of Fluid (VOF) Stencils

2.3.2 Efficient Least-Squares VOF Interface Reconstruction Algorithm (ELVIRA)

The interface normal calculation is evaluated by finite difference approximation of volume-fraction gradient ∇C or by an error minimization equation. All of these algorithms are done for 3x3 VOF stencils.

In the first method called Youngs' finite difference method, the normal $\mathbf{m}$ is determined by local gradient of volume fractions ∇C, denoted as,

$$\mathbf{m} = -\nabla_h C \tag{2.5}$$

where $\mathbf{m}$ is the normal vector. The normal vector is taken as an average of components of m of four cell-corner values.

For the same 3x3 VOF stencil block, in the centered-columns difference method, the volume fractions are added in the vertical direction to define height functions $y = f(x)$ or horizontal direction to define width function $x = g(x)$. For the height function, the slope of the straight line is computed as,

$$m_{xc} = -\frac{1}{2h}(y_{i+1} - y_{i-1}) = -\frac{1}{2}\sum_{k=-1}^{1}(C_{i+1,j+k} - C_{i-1,j+k}) \tag{2.6}$$

The efficient least-squares VOF interface reconstruction algorithm (ELVIRA) was introduced by Pilliod and Puckett (2004). Along with the centered-columns method, in ELVIRA one can use forward or backward finite differences to compute the slope as done in equation 2.6. In forward and backward finite difference the estimated m_{xf} and m_{xb} formulation can be respectively written as,

For the forward finite difference in height function, the line slope $\mathbf{m_{xf}}$ is,

$$m_{xf} = -\frac{1}{h}(y_{i+1} - y_i) = -\sum_{k=-1}^{1}(C_{i+1,j+k} - C_{i,j+k}) \tag{2.7}$$

For the backward finite difference in height function, the line slope $\mathbf{m_{xb}}$ is,

$$m_{xb} = -\frac{1}{h}(y_i - y_{i-1}) = -\sum_{k=-1}^{1}(C_{i,j+k} - C_{i-1,j+k}) \tag{2.8}$$

There are 12 possible combinations of slope m (positive and negative slopes for backward, central, and forward differences from row & column sums of volume fractions) of the approximate linear interface in the 2D case (Pilliod and Puckett (2004), Fricke *et al.* (2020)). The criteria for selection of the slope is based upon one which gives the exact reconstruction of the linear phase interface. An error equation is minimized and defined as L^2 or L^∞ discrete error between the centered cell $C_{i,j}$ and adjacent 3x3 VOF stencil cells. The difficulty with the ELVIRA method is that the method is slow and computationally costly for 3D cases (Tryggvason *et al.* (2011)).

2.3.3 Problem Statement

Previous methods use least square approximation to reduce the error norms while reconstructing the phase interface with straight lines. It becomes challenging to reconstruct the exact representation of a circular arc with straight lines using the above interface reconstruction methods. As an alternate method to this problem, the thesis investigates the ability of neural networks to reconstruct circular interfaces in 2D instead of straight lines.

Like a reverse problem, the volume fraction values are calculated back for 3x3 VOF stencil, from the neural network reconstructed phase interface, and compared with the volume fraction values of actual interface geometry. Lastly, the error norms are calculated with the least squares error criteria between the same 3x3 VOF stencils to gauge the performance of the neural network. The methodology is further stated in Chapter 4.

Chapter 3

MACHINE LEARNING

3.1 Introduction to Neural Networks

Machine learning is a system that is trained rather than programmed explicitly Chollet (2017). The algorithm learns from initial data sets to perform tagging of target values with specific input values. It applies the fundamentals of statistical rules, like minimizing the loss for predictions compared to its output. But, machine learning fundamentals differ from statistical mathematics, in which it takes complex data set inputs like text and images and finds particular empirical optimized parameters to match the predictions with output.

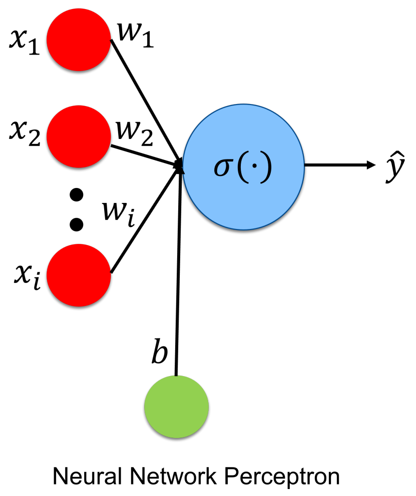


Figure 3.1: Neural Network Perceptron Illustration

The neural network algorithm of relating the inputs, outputs, loss functions, and optimizers gives more flexibility to automate tasks over millions of data sets, which is infeasible by current fundamental statistics. Secondly, it offers more flexibility to optimize the models with hyper-parameter tuning to improve its performance (Smith (2018)) and represents a gray area between mathematical theory and engineering-oriented ideas.

Figure 3.1 shows an illustration of how the model predicts the output values. The VoF inputs are taken as individual x values and multiplied by initialized weights. The dot product is summed and if necessary zero bias is added to it. A nonlinear function is applied to this value to yield the predicted y value. The choice of approximation is decided between linear and nonlinear methods (Brunton *et al.* (2020)). The mathematical formulation is represented as,

$$\hat{y} = \sigma\left(\sum_{i=1}^{N} w_i x_i + b\right) \tag{3.1}$$

The weights are initialized as uniform weights. However, different initialization methods like Xavier, Kaiming, Orthogonal, and so forth could be used (Glorot and Bengio (2010)). Weights are updated by adding the product of the learning rate and gradient of loss, which represents the updated weight. Mathematically, the weight update equation is,

$$W^{i+1} = W^{i} - \epsilon \nabla Loss_{W}^{i} \tag{3.2}$$

A dynamic/adaptive learning rate (ϵ) is used over constant learning rate, as the model performance for phase interface improved via faster convergence of mean absolute error as a loss function.

Chapter 4

METHODOLOGY

The methodical approach followed in this thesis work is explained in this chapter. The main focus is on finding a relationship between the radius of the phase interface and the volume fractions. The approach is divided into three sections -
(1) Generating a set of synthetic data sets for 2D circular phase interfaces.
(2) Fitting data using a Neural Network architecture to predict phase interface.
(3) Deriving the L norms as an error metric to evaluate the error in the volume fraction fields.

4.1 Volume of Fluid (VOF) Data Generation

Data sets are generated using an open-source VOFI library (Bnà *et al.* (2016)) compiled in MATLAB environment. This data is considered as a benchmark to compare the neural network model predictions, and indirectly compute the volume fraction field errors.

The VOFI library uses “vofi.c” function to generate specific volume fraction field values across a specified stencil. The C function is wrapped inside the MATLAB environment by using the ”mex” compiler in the MATLAB library.

The ”mathOpsIntegrated.m” file wraps the ”vofi.c” C function from the VOFI library and calls the ”vofi.h’ header file. It takes 5 inputs namely the x, and y centers of the circle, the interface orientation factor (assumed symmetric, no change if alpha, (α=0)), and the major and minor axis of the circle ((a=b=r), as the radius of the circle). Further outputs, the fraction field as an output for the function.

```
  $a_1$ : major axis, $b_1$ : minor axis, $x_c$ : x center, $y_c$ : y center,
  $i$ : i location of cell, $j$ : j location of cell
**Require:** $a_1$, $b_1$, $x_c$, $y_c$, $i$, $j$
**Ensure:** Compiling VOFI library with Matlab Function
  codegen mathOpsIntegrated.m -args$\{a_1, b_1, x_c, y_c, i, j\}$ -libvofi.so
```

Figure 4.1: Compiling VOFI Library in C in Matlab Environment

The ”mathOpsIntegrated.m” file is a MATLAB function file that calls the headers and the ”myvofi.c” C function file from the VOFI library in MATLAB environment. This function is called inside a MATLAB ”for” loop for randomly seeded circular phase interface parameters. This enables the generation of larger datasets, containing circular phase interfaces, to train neural network models.

The mesh grid resolution selected across the 1x1 domain is 20x20. Circles as a geometric shape interface are selected to be mapped through a neural network model. The centers of the circle and the radius are seeded randomly in the MATLAB script as shown in figure 4.2. Randomizing center of the circular phase interface in the range of -0.2 to 1.2. And radius randomized from 0 to 1.

A set of large synthetic data for circles with a radius ranging from 0 to 1, within a 1x1 domain, representing a 3 by 3 stencil for VoF are generated. To ensure sufficient possibilities of circle radii are captured, specific amounts of VoF stencils are generated for each circle radius range.

The data distribution for randomly generated circles' radius represented a Gaussian distribution. To use a uniform data distribution of VoF stencils, more data is generated within each bin. And R/Δx is considered to be a crucial parameter to predict phase interface radius values, where R is the radius of the circular phase

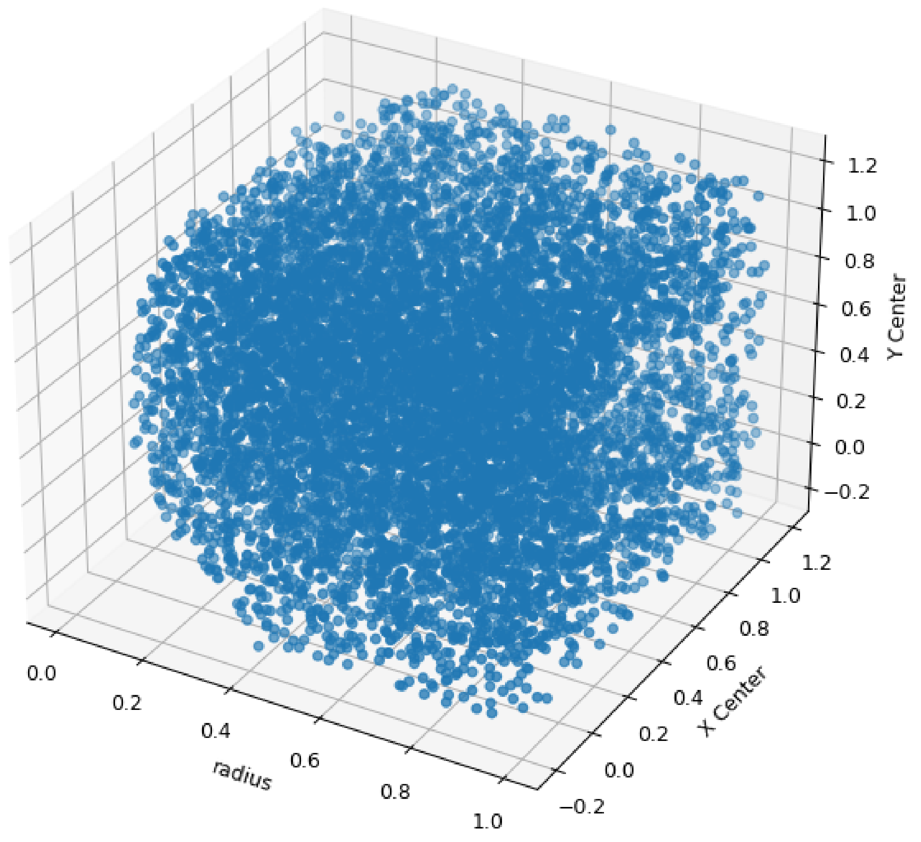


Figure 4.2: Random Data Distribution of Circular Phase Interface Radius

interface, and Δx is the grid spacing inside the domain.

4.2 Testing Qi *et al.* (2019) Neural Network Robustness

The recent developments to map curvature predictions with volume fraction field are done for fixed centers and radius values by Qi *et al.* (2019). The results give a good prediction of curvature represented in figure 4.3.

So to assess the robustness of the neural network used in the paper, the data generated by randomization in this thesis work is passed through the same network configuration.

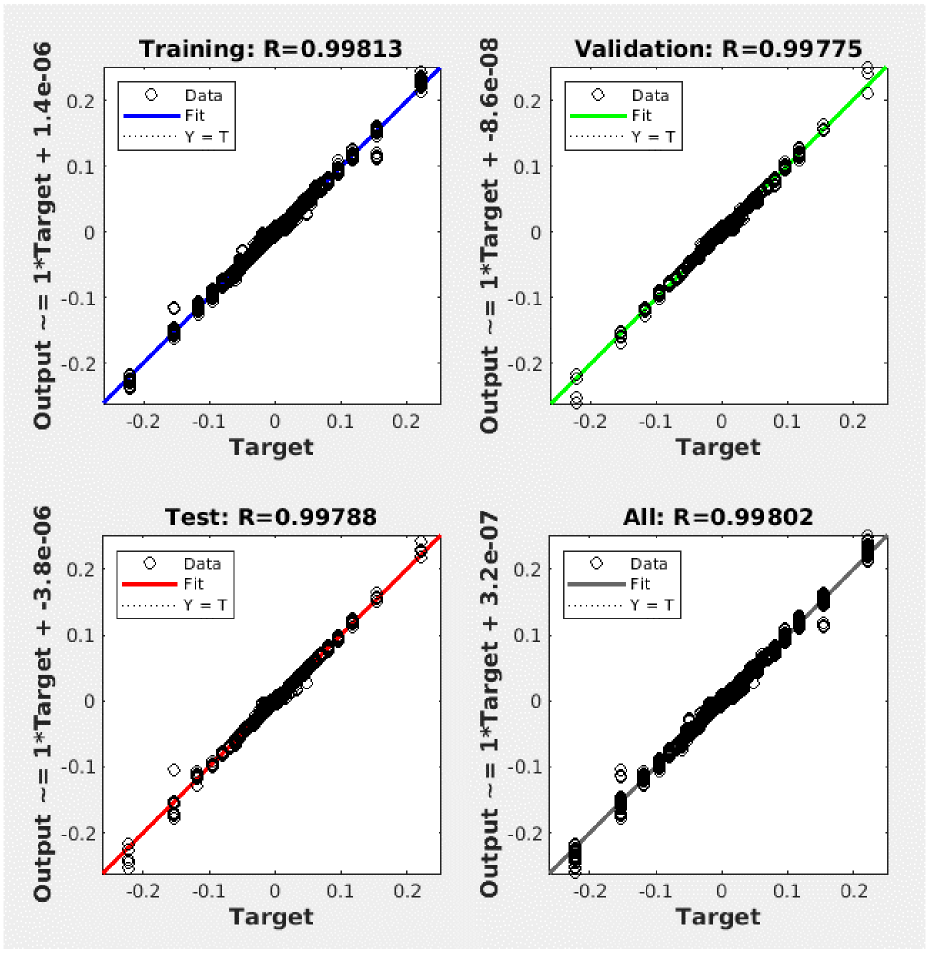


Figure 4.3: Fitted Curvature Predictions from Network Model to Target Values (a) Train (B) Test (C) Validation (D) Qi *et al.* (2019) Data

Qi *et al.* (2019) had examined a 2-layer MLP neural network architecture to have a fluid phase radii interface as an output from the network. The results are shown in figure 4.3, which shows the curvature predictions. It indicates that using the current Qi *et al.* (2019) approach with randomly generated circular interface radii with random centers, there are certain data points where the actual interface is not reconstructed by the neural network.

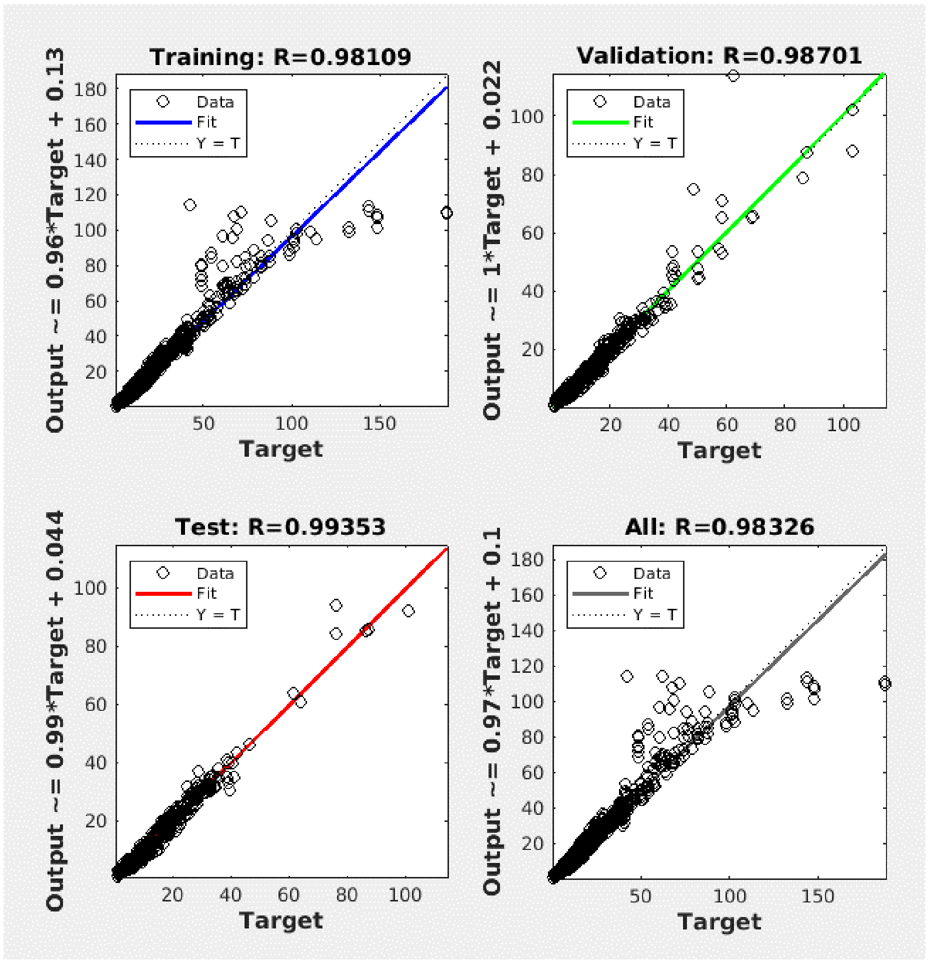


Figure 4.4: Fitted Curvature Predictions from Network Model to Target Values (a) Train (B) Test (C) Validation (D) Thesis Data

4.3 Empirical Testing of Neural Network Architectures

The models were tested and optimized for minimal losses to have closer phase interface predictions. The loss values were mapped and the model is trained till the loss values stay stagnated or if the model could not learn further.

The plots are generated for 1,000,000 circles initially which shows Gaussian distribution and more VoF stencils are added by generating more circles within the bins. Figure 4.6a & 4.7, represents Gaussian distribution for the test, train, and entire data sets.

In figure 4.6a, the data points in each divided bin are checked to make sure the model has enough data points (circular phase interfaces) for the model to train. For this purpose, more data points are added in each bin. Uniform data generation were carried out where within each bin random circular phase interfaces are generated till the maximum VoF stencil count is met within the entire data sets. The data distribution afterward is present in figure 4.6b.

Figure 4.5 shows the neural network architecture used for current empirical testing.

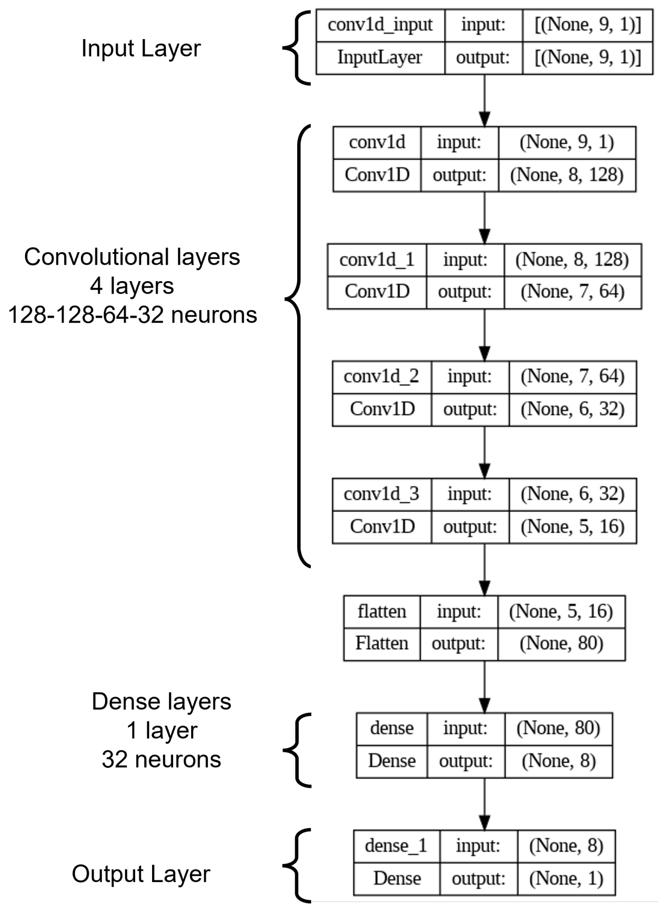
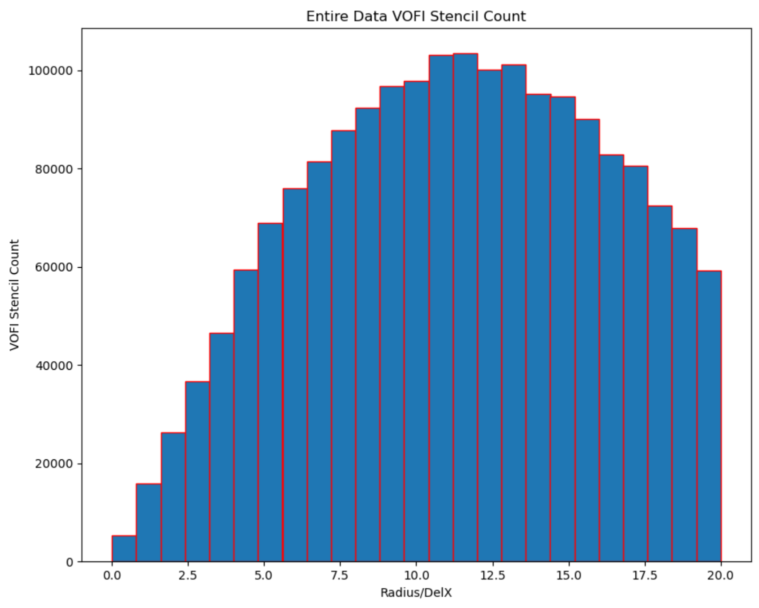
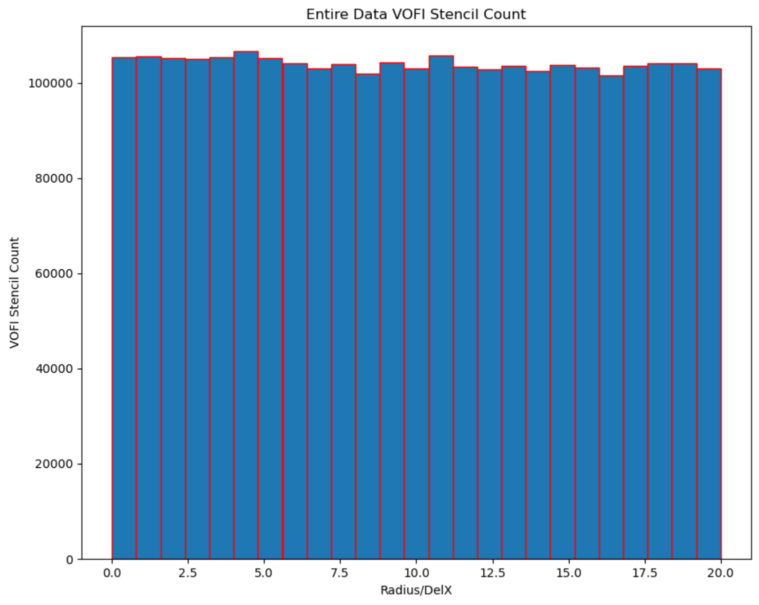

Input Layer
conv1d_input | input: | [(None, 9, 1)]
InputLayer | output: | [(None, 9, 1)]
Convolutional layers
4 layers
128-128-64-32 neurons
conv1d | input: | (None, 9, 1)
Conv1D | output: | (None, 8, 128)
conv1d_1 | input: | (None, 8, 128)
Conv1D | output: | (None, 7, 64)
conv1d_2 | input: | (None, 7, 64)
Conv1D | output: | (None, 6, 32)
conv1d_3 | input: | (None, 6, 32)
Conv1D | output: | (None, 5, 16)
flatten | input: | (None, 5, 16)
Flatten | output: | (None, 80)
Dense layers
1 layer
32 neurons
dense | input: | (None, 80)
Dense | output: | (None, 8)
Output Layer
dense_1 | input: | (None, 8)
Dense | output: | (None, 1)

Figure 4.5: Neural Network Architecture

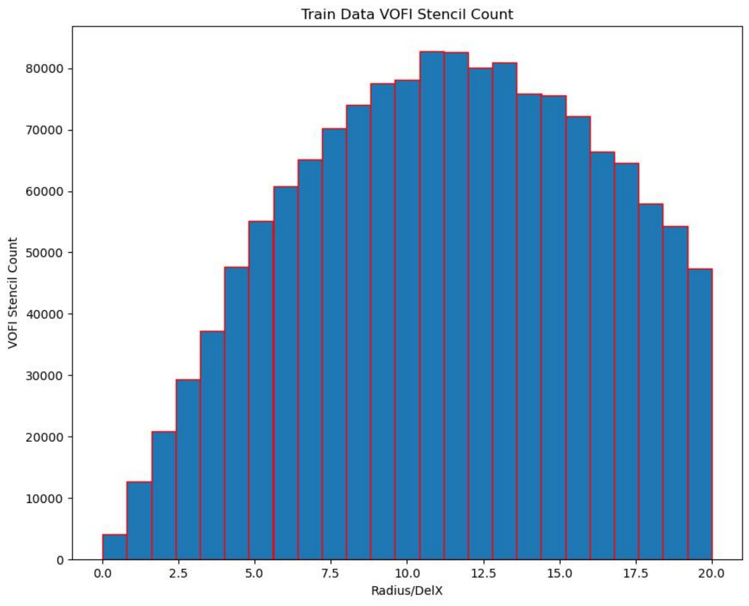


(a) 1.8 Million VoF Stencils Data Points

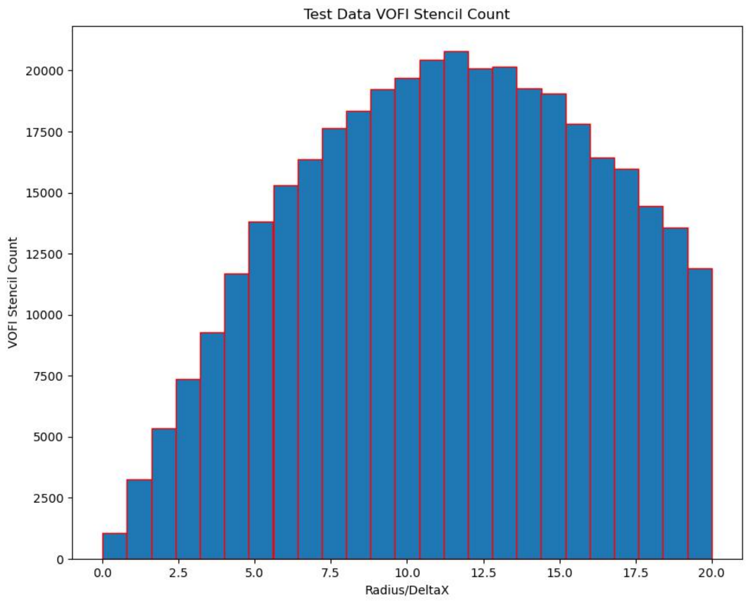


(b) 2.6 Million VoF Stencils Data Points for Uniform Data

Figure 4.6: VoF Stencils Count for 1M+ Circles

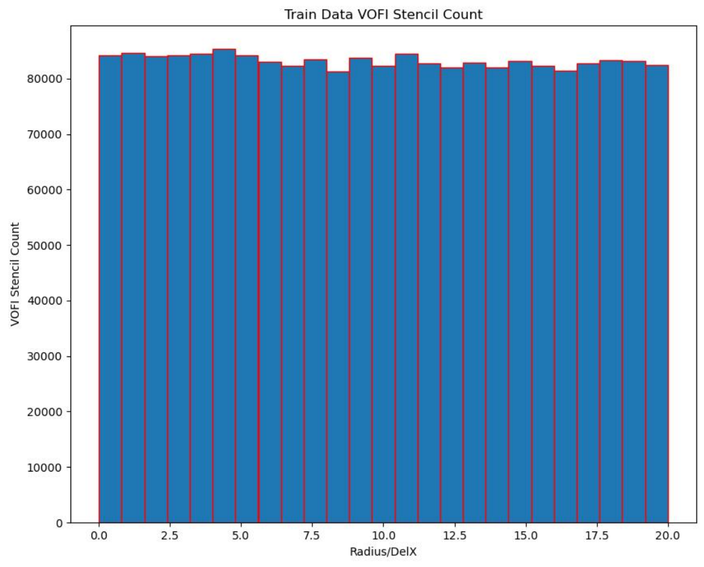


(a) Train Data

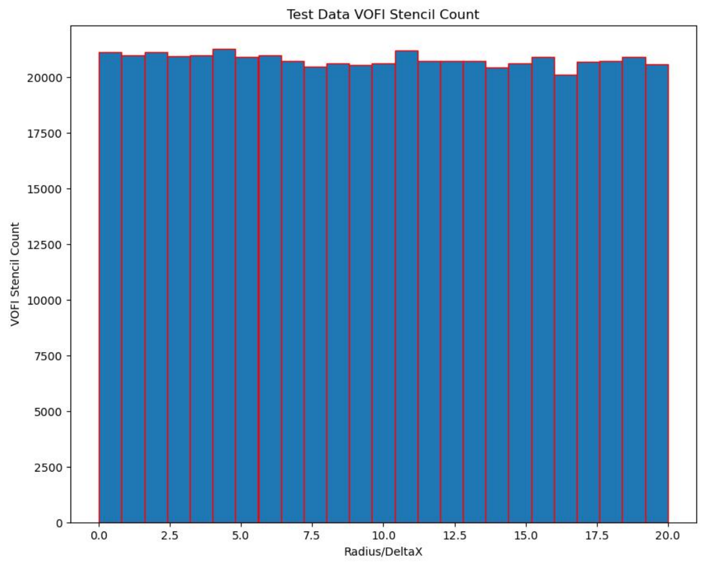


(b) Test Data

Figure 4.7: Train & Test VoF Stencils Count for 1M+ Circles (1.8M Data Points)



(a) Train Data



(b) Test Data

Figure 4.8: Train & Test Uniform VoF Stencils Count for 1M+ Circles (2.6M Data Points)

Chapter 5

RESULTS

5.1 Model 1 - 5x5 VoF Stencils with 15,000 Data Points

To test an optimal learning rate, the network losses are mapped to get lower loss convergence values. The loss value for train and validation data set in figure 5.1 converges in the range of 10^{-3} for train & validation data sets. The model training is performed on 15,000 data points for 5x5 VoF stencils.

The network architecture chosen here consisted of 4 convolutional layers and 1 dense layer. The network configuration was tested for different layers and neurons, learning rate, and batch size. The model which gave lower loss convergence values was selected to make VoF interface predictions. The selected optimal neural network parameter architecture is shown in figure 4.5.

Table (5.1) shows the L norm values for the neural network architecture of model 1 are -

L_{Inf}	L_1	L_2
2.8623	0.4376	0.5939

Table 5.1: Error Norms Metric for Model 1

Scaling the layers inside the neural network architecture beyond 5-7 layers, made the loss values overshoot for validation data as referenced in figure 5.2. Thereby, making the predictions worse. From this insight, the model was limited to less than 5 layers for further hyper-parameter testing of the network architecture.

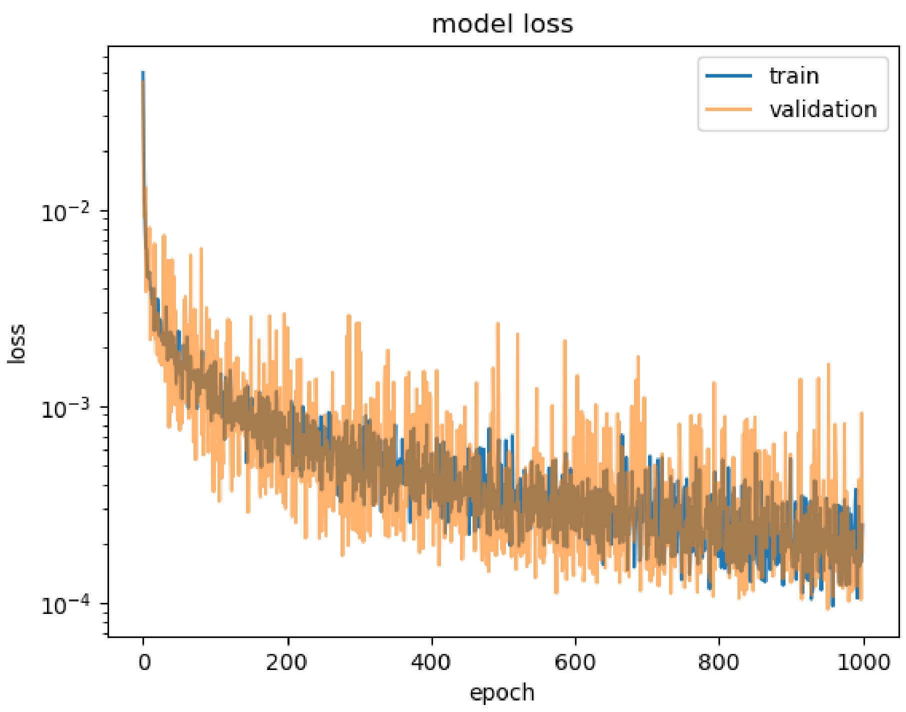


Figure 5.1: Model Loss Vs Epoch - Model 1, 15,000 Data Points, 1000 Epochs, Batch Size 5x5 VoF Stencil

5.2 Model 2 - CNN & DNN model, 3x3 VoF Stencils with 1.8M Data Points

Sometimes, the model does face an overshoot of loss values because of non-optimal hyperparameters for the model to converge. This is referenced in figure 5.2. Before calculating the error norm values, the error associated with the model predictions and actual phase interface is visualized in figure 5.3. Most of the predicted circular radius values lie around 10^{-2} error for test data of $368k$ data points of total $1.8M$ Vof stencil points generated for more than $1M$ random circle initialization. The overshooting of loss values is avoided by choosing a set of stable neural network parameters like small batch size, decaying learning rate, and suitable number of neurons, and layers for the test loss curve to overlap with train loss curves as the model is being trained on set batches of train data for a certain number of epochs.

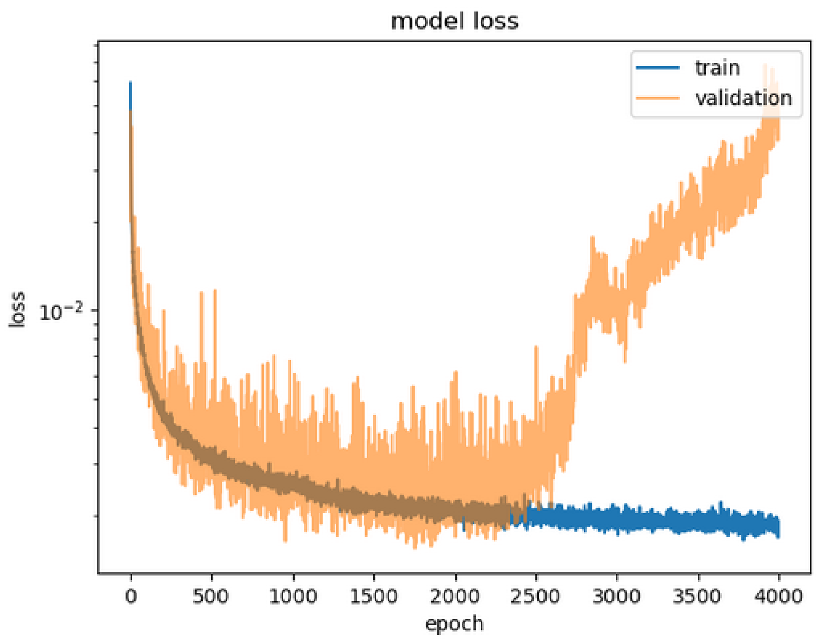


Figure 5.2: Loss Vs Epoch - Validation Data Overshoot with More than 6 Neuron Layers - Model 2, 31,000 Data Points, 4000 Epochs, Batch Size 16, 3x3 VoF Stencil

Table (5.2) shows the error norm values for curvature values predicted by the neural network architecture of model 2 are -

L_{Inf}	L_1	L_2
0.2265	0.017	0.0227

Table 5.2: Error Norms Metric for Curvatures - Model 2

The values of error norm are first evaluated in table 5.2 for curvatures rather than VoF fraction field data because of larger computational time. If the model gives lower error norm values for curvatures then they are evaluated on the VoF fraction fields. This is only done for larger test data sets, while for smaller size of test data sets the computational time for error norms is not too large.

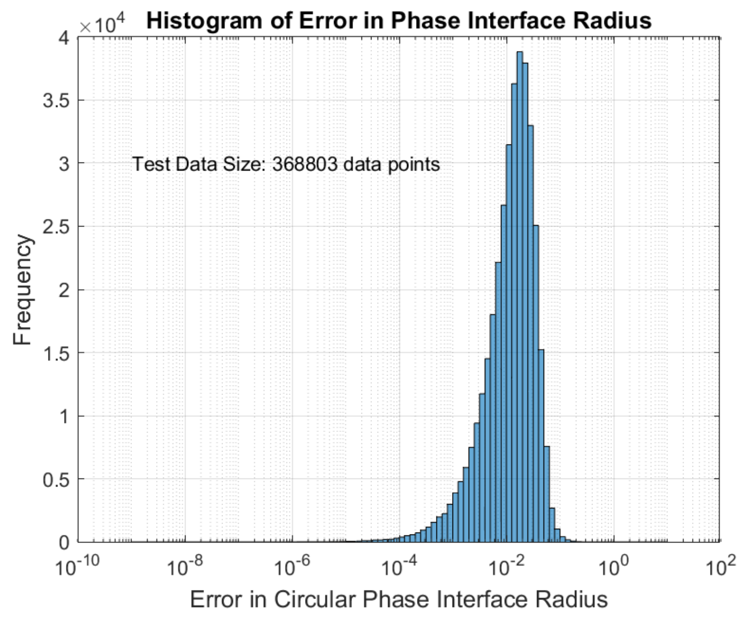


Figure 5.3: Error in Phase Interface Radius - Model 2, 1.8M Data Points, 4000 Epochs, Batch Size 10k, 3x3 VoF Stencil

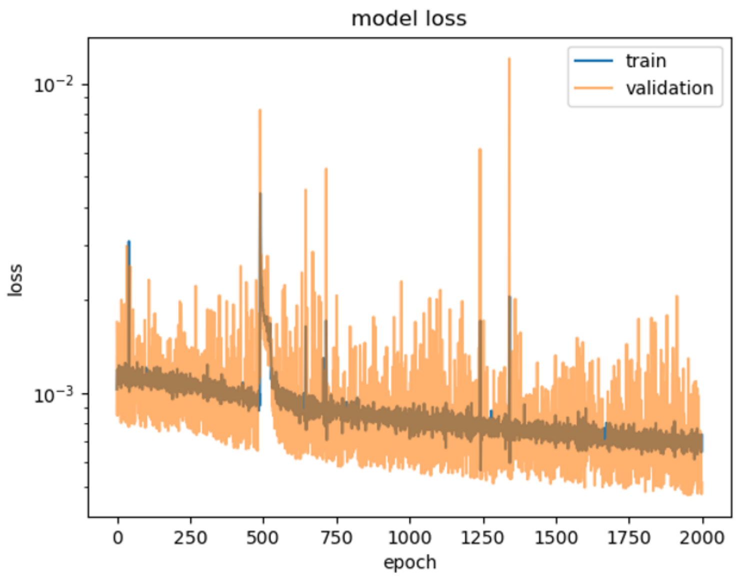


Figure 5.4: Loss Vs Epoch - Model 2, 1.8M Data Points - 368k Test Data Points, 4000 Epochs, Batch Size 10k, 3x3 VoF Stencil

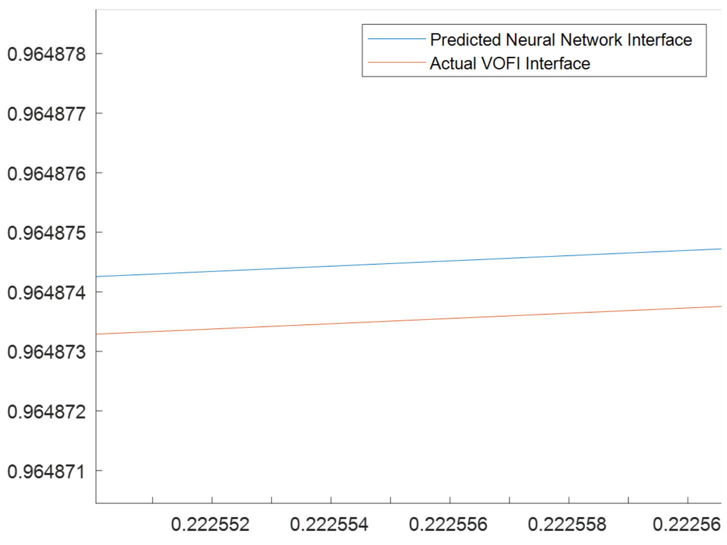


Figure 5.5: Best Circular Interface Phase Prediction by the Model - Model 2

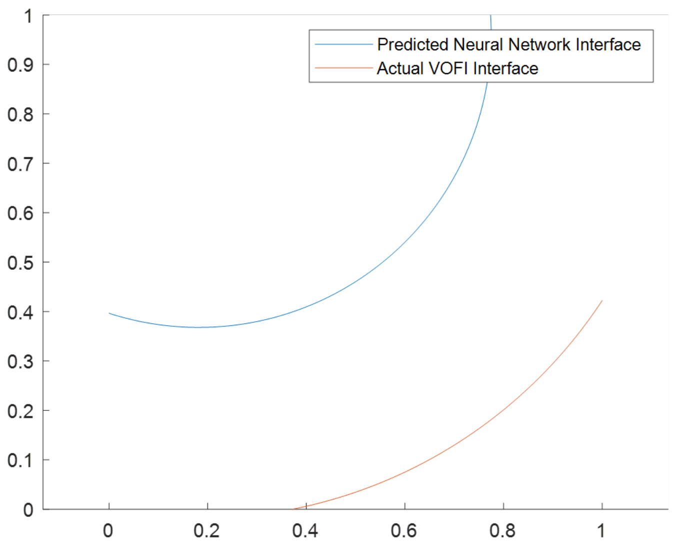


Figure 5.6: Worst Circular Interface Phase Prediction by the Model - Model 2

5.3 Model 3 - CNN & DNN, 3x3 VoF Stencils with 31,000 Data Points

The current combination of dense and convolutional neural layers with an optimal set of hyperparameters like 2000 epochs, 16 sample batch size, and 3x3 VoF stencil architecture shows loss value converging closer to 10^{-3}. The loss curves for the train & test data have a closer overlap of values shown in figure 5.7. The table (5.3) shows error norm values for the neural network architecture of model 3 are in the range of order 10^{-1} to 10^{1}-

L_{Inf}	L_1	L_2
1	0.1198	0.2525

Table 5.3: Error Norms Metric for Model 3

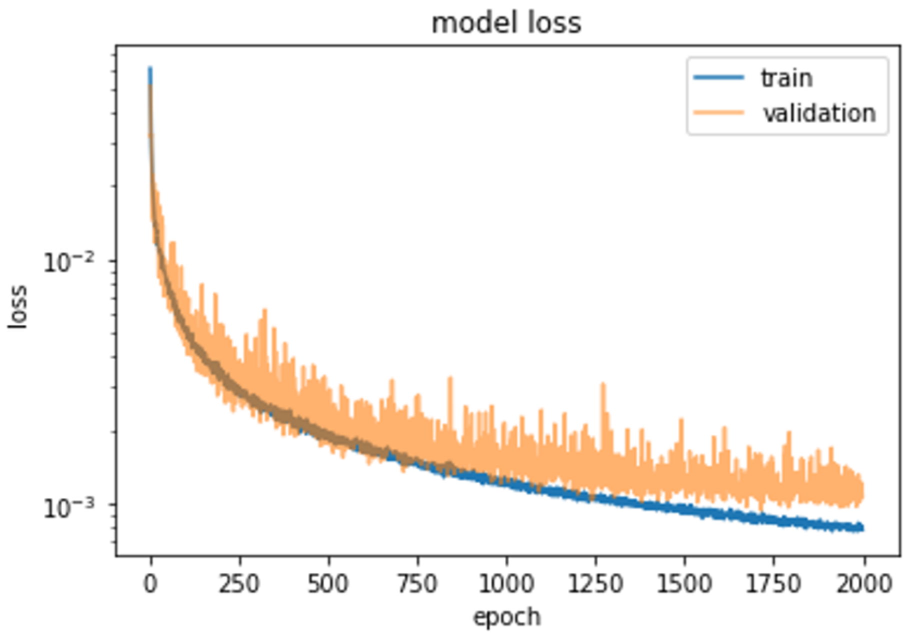


Figure 5.7: Loss Vs Epoch - Model 3, 31,000 Data Points, 2000 Epochs, Batch Size 16, 3x3 VoF Stencil

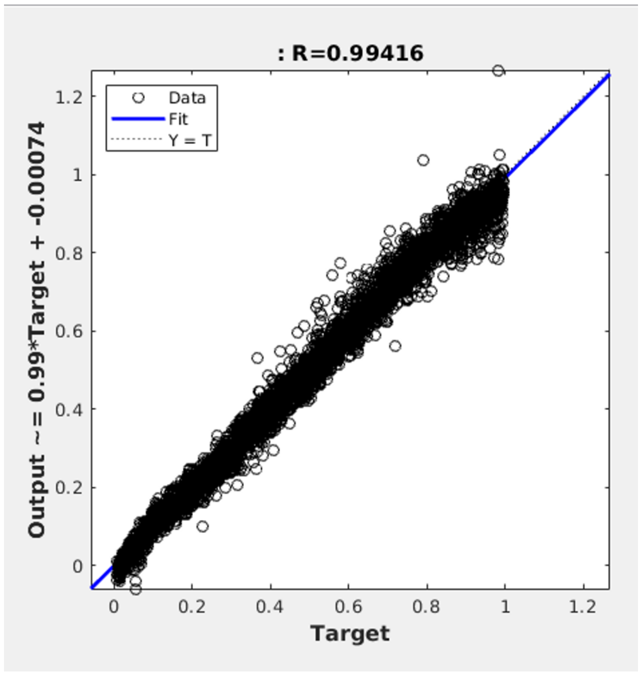


Figure 5.8: Model Fit for Actual Circular Phase Geometry Vs Neural Network Reconstructed Circular Phase Interface

In model 3, CNN and DNN dense layers are tested together for a small amount of data. The neural architecture implemented is shown in figure 5.12. This architecture was found to converge to minimal mean absolute error loss in the model when tested for the first 10 iterations. The model fit is tested by plotting the neural network reconstructed radii of circular interfaces against the actual phase interfaces geometry as referenced in figure 5.8. Model 3 gives the best predictions with a correlation coefficient value of 0.99416. The closer the value to 1, the predicted values are equivalent to the actual values. The model predicted the closest circular phase interface up to 10^{-8} decimal accurately. While the worst predicted value was up to 10^{2} digits accurate. This is very well captured by the L_{∞}, L_1, and L_2 error norms. Visual representation of the best and worst actual with predicted circular phase interface is post-processed in MATLAB shown in figure 5.9 and figure 5.10.

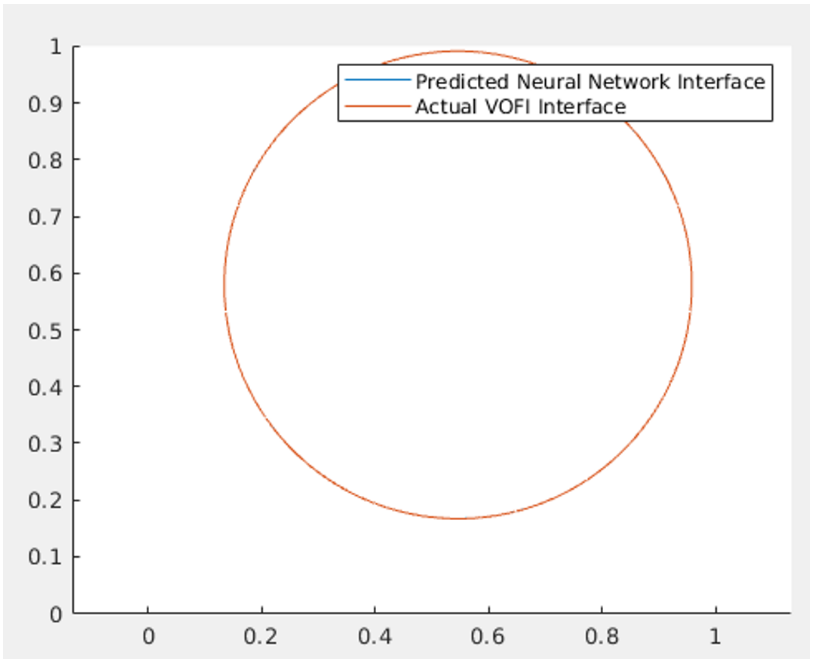


Figure 5.9: Best Circular Interface Phase Prediction by the Model - Model 3

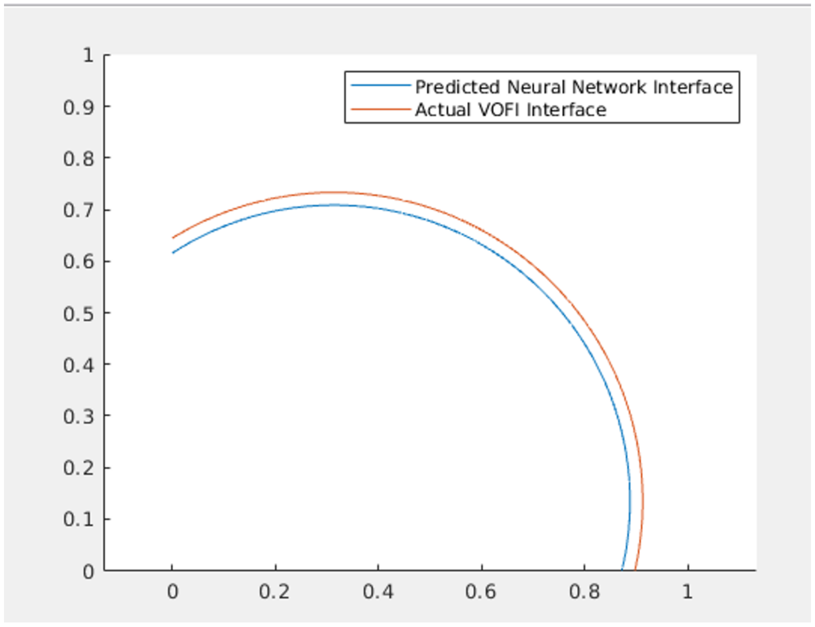


Figure 5.10: Worst Circular Interface Phase Prediction by the Model - Model 3

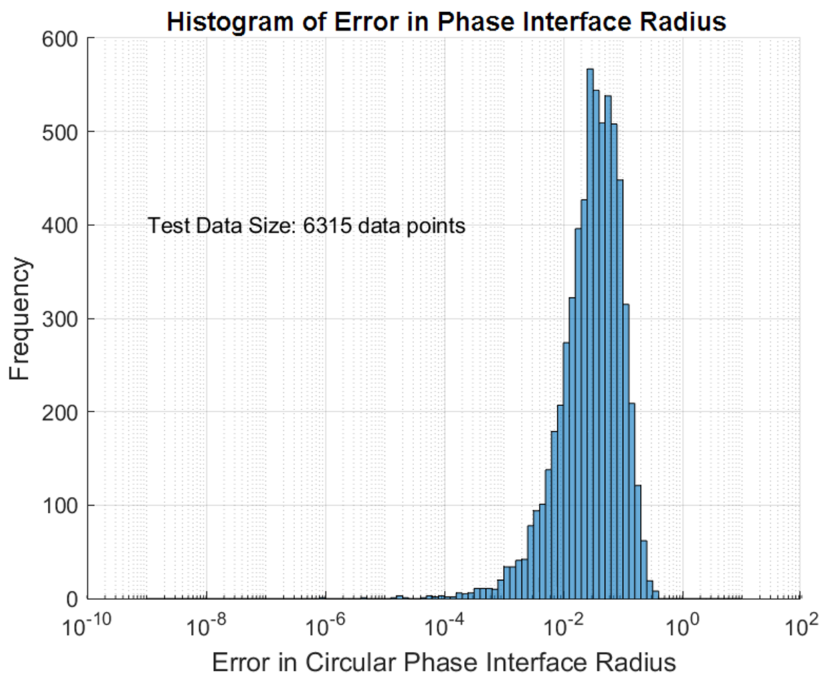
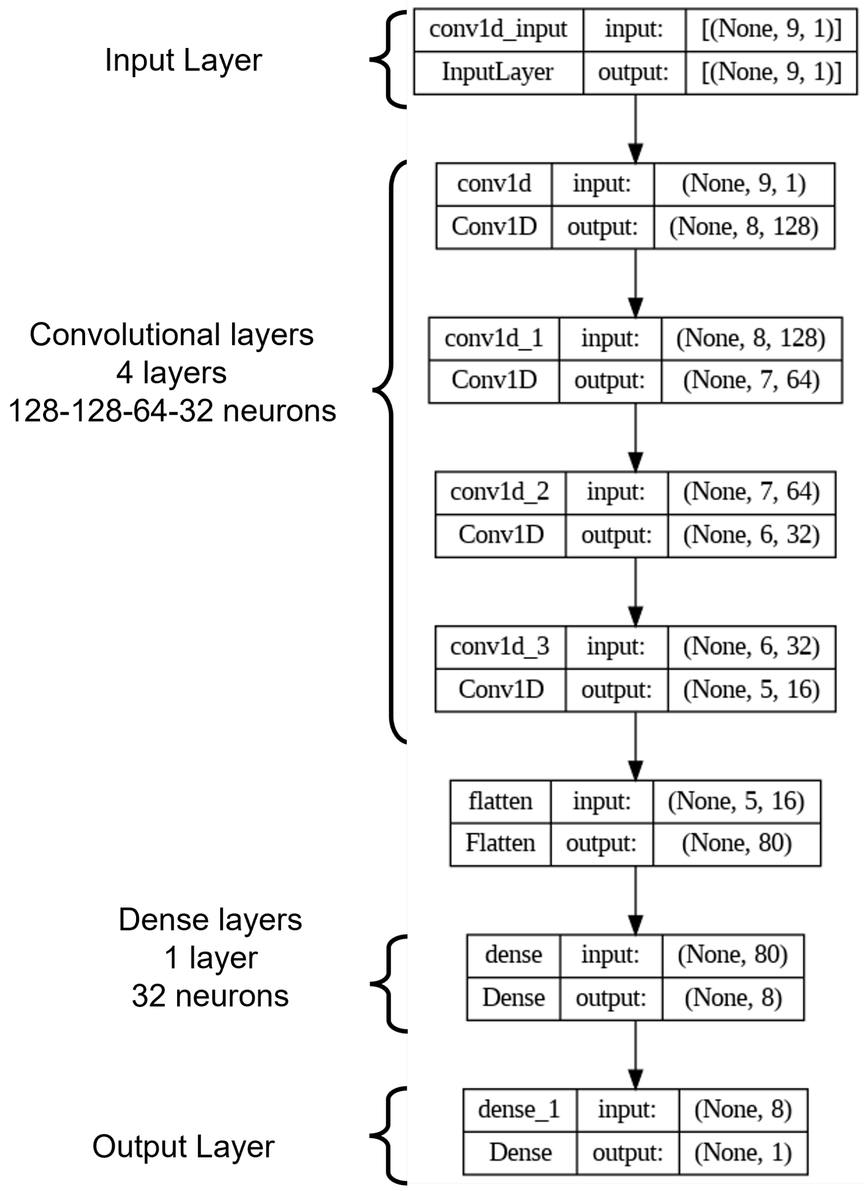


Figure 5.11: Error in Phase Interface Radius - Model 3, 31,000 Data Points, 2000 Epochs, Batch Size 16, 3x3 VoF Stencil



Input Layer
conv1d_input | input: | [(None, 9, 1)]
InputLayer | output: | [(None, 9, 1)]
Convolutional layers
4 layers
128-128-64-32 neurons
conv1d | input: | (None, 9, 1)
Conv1D | output: | (None, 8, 128)
conv1d_1 | input: | (None, 8, 128)
Conv1D | output: | (None, 7, 64)
conv1d_2 | input: | (None, 7, 64)
Conv1D | output: | (None, 6, 32)
conv1d_3 | input: | (None, 6, 32)
Conv1D | output: | (None, 5, 16)
flatten | input: | (None, 5, 16)
Flatten | output: | (None, 80)
Dense layers
1 layer
32 neurons
dense | input: | (None, 80)
Dense | output: | (None, 8)
Output Layer
dense_1 | input: | (None, 8)
Dense | output: | (None, 1)

Figure 5.12: Convolutional & Dense Neural Network Architecture for Model 3

5.4 Model 4 - DNN, 3x3 VoF Stencils with 31,000 Data Points

A neural network architecture of only dense layers was tested. However, the predictions for model 4 grew worse for the same model parameters like dynamic learning rate, batch size, and epochs. Table (5.4) shows the error norm values for the neural network architecture of model 4.

L_{Inf}	L_1	L_2
1.7012	0.3169	0.3646

Table 5.4: Error Norms Metric for Model 4

The L_∞, L_1, and L_2 error norms are in the range of order of 10^{-1} to 10^1. The best predicted compared to the actual circular phase interface by model 4 is presented in figure 5.13.

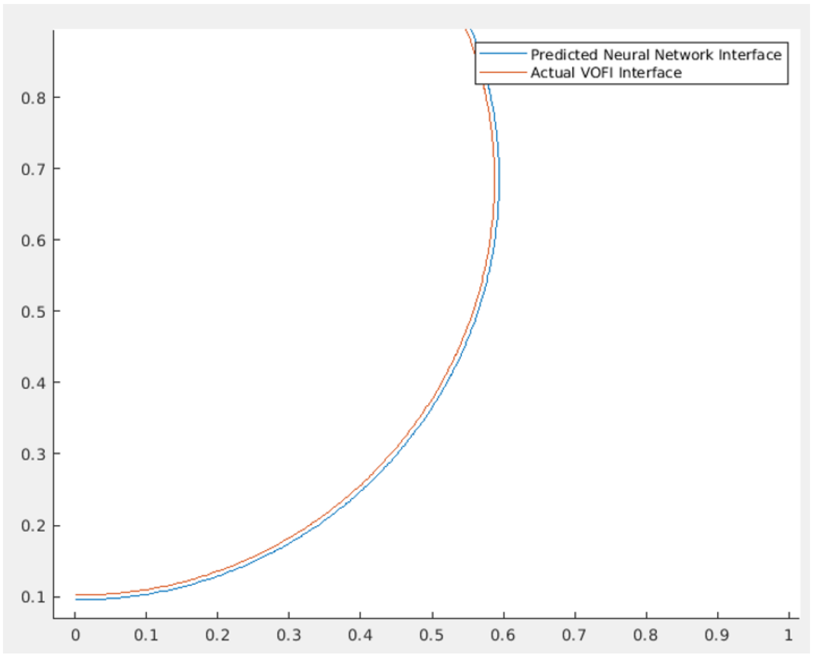


Figure 5.13: Best Circular Interface Phase Prediction by the Model - Model 4

The loss curves stagnate around 10^{-2} after 1000 epochs. The loss curves are presented in figure 5.14 and figure 5.15.

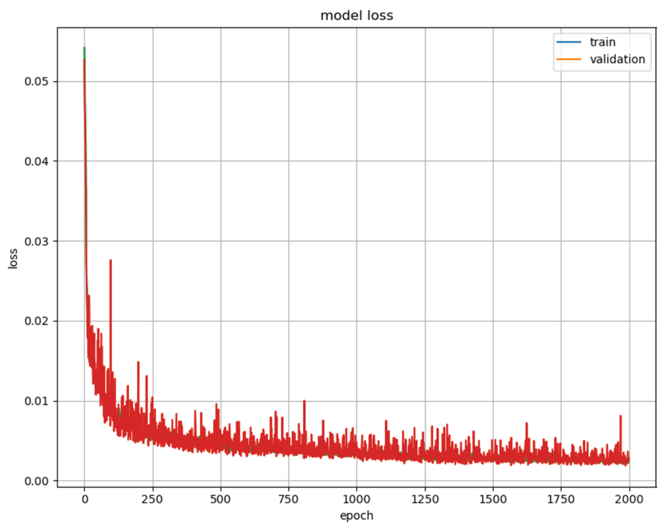


Figure 5.14: Loss Vs Epoch - Model 4, 31,000 Data Points, 2000 Epochs, Batch Size 32, 3x3 VoF Stencil

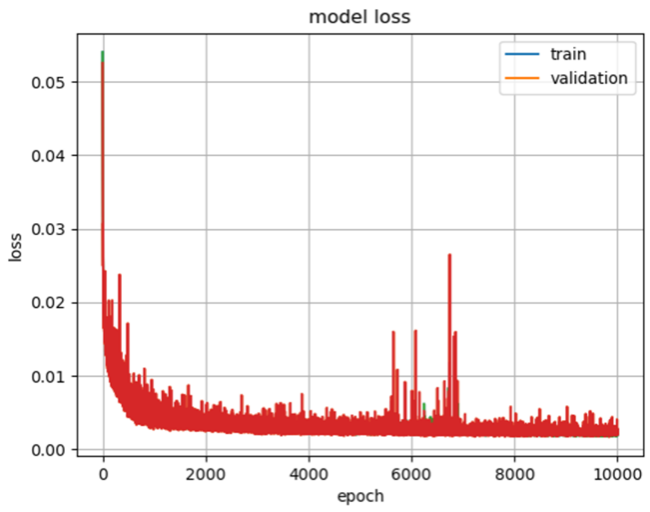


Figure 5.15: Loss Vs Epoch - Model 4, 31,000 Data Points, 10,000 Epochs, Batch Size 32, 3x3 VoF Stencil

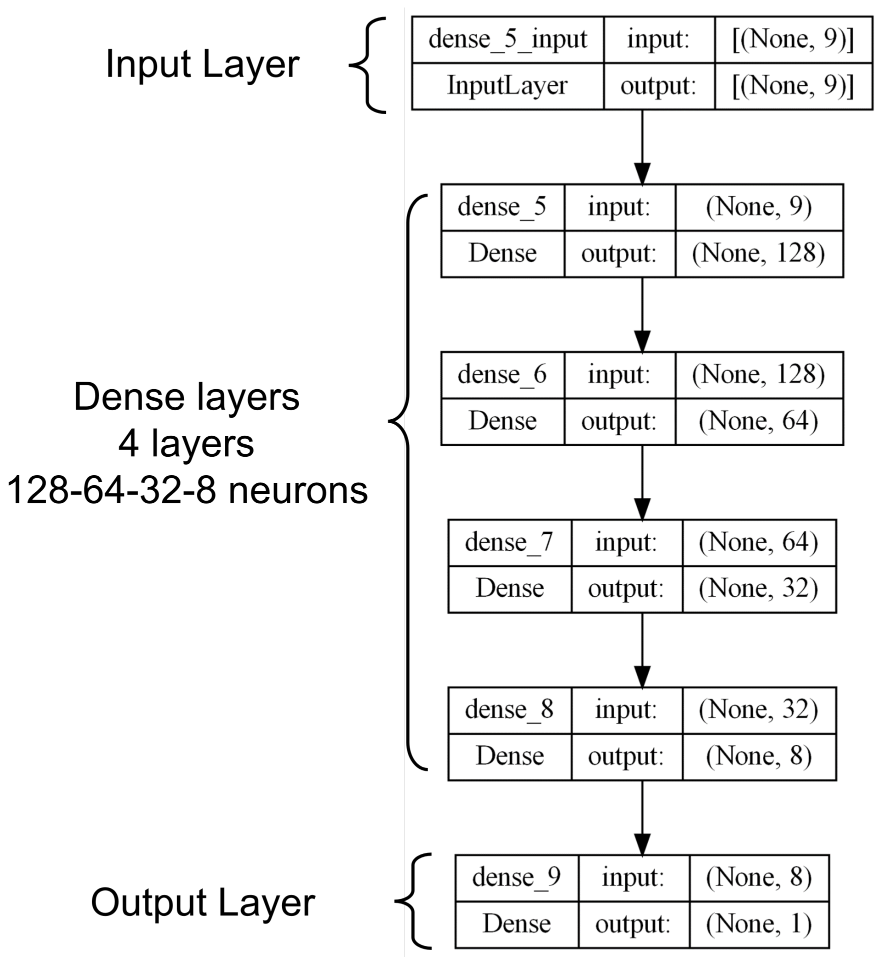

Input Layer
dense_5_input | input: | [(None, 9)]
InputLayer | output: | [(None, 9)]
Dense layers
4 layers
128-64-32-8 neurons
dense_5 | input: | (None, 9)
Dense | output: | (None, 128)
dense_6 | input: | (None, 128)
Dense | output: | (None, 64)
dense_7 | input: | (None, 64)
Dense | output: | (None, 32)
dense_8 | input: | (None, 32)
Dense | output: | (None, 8)
Output Layer
dense_9 | input: | (None, 8)
Dense | output: | (None, 1)

Figure 5.16: Dense Neural Network Architecture for Model 4

5.5 Model 5 - CNN, 3x3 VoF Stencils with 2.6M Data Points

In model 5, the architecture is tested with only convolutional layers. Model 5 experiences larger fluctuations of the validation loss curve compared to the train loss curve. This is because to initially test the model's performance with lesser computational time, a larger batch size is selected. Figure 5.17 shows the mean absolute error (MAE) loss for the model stagnates around 10^{-2} after 10,000 epochs. Table (5.5) shows the error norm values tested for 80% of the whole data for the model 5 architecture -

L_{Inf}	L_1	L_2
1.5524	0.0894	0.1992

Table 5.5: Error Norms Metric for 230k/520k Data Points for Model 5

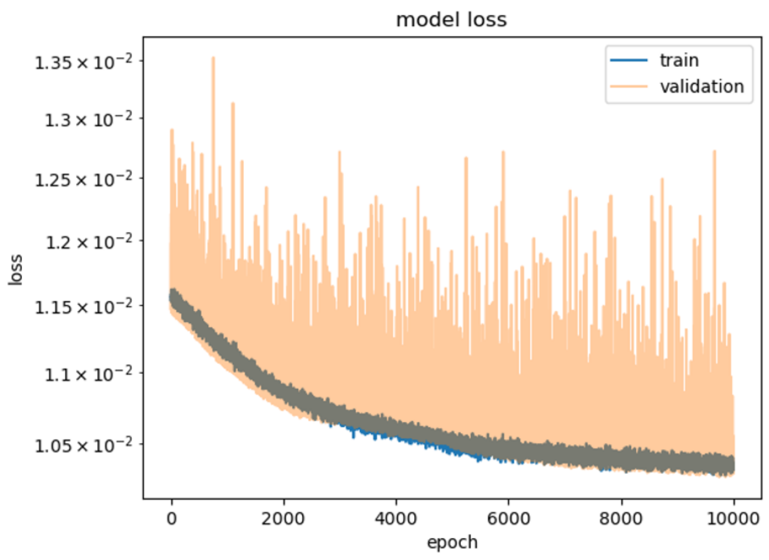


Figure 5.17: Loss Vs Epoch - Model 5, 2.6M+ Data Points, 1000 Epochs, Batch Size 10k, 3x3 VoF Stencil

The error distribution for the range of predicted reference value is shown in figure 5.18. The best prediction of the network has an order of error of 10^{-9}, while the worst prediction has an order of 10^2. The large error norms capture poor reconstruction of the circular phase interface by the neural network. It is difficult to attain lower error norms in the order of 10^{-9} for all the data points. Further strategies like increasing the data size to train the model for different corrugated and geometrical surfaces and building effective search methods to seek for more optimal neural network parameters within the search space at the cost of computation, will certainly help to lower the error norms distribution and achieve the exact reconstruction of circular fluid phase interfaces from the neural network model. Moreover, this will shift the mean of error between the neural network reconstructed phase interface and the actual phase interface geometry. Instead of having a left-skewed histogram, it will be right-skewed distribution.

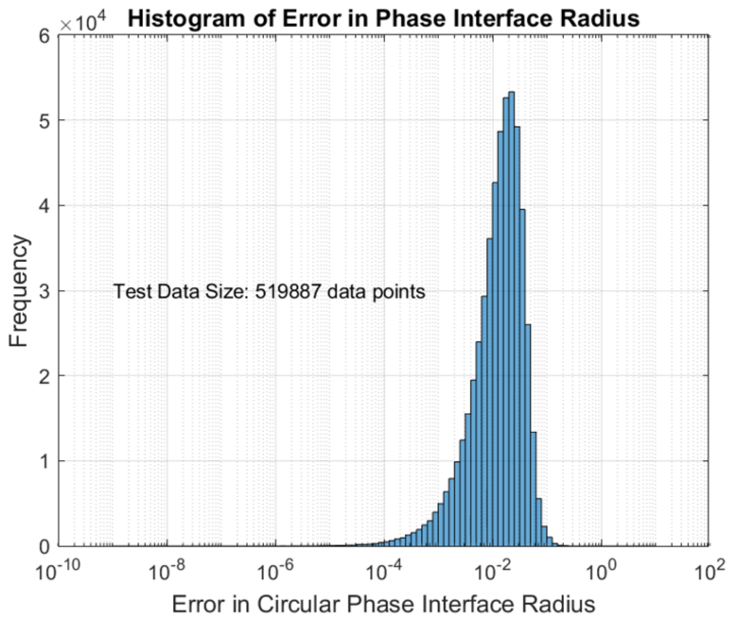


Figure 5.18: Error in Phase Interface Radius - Model 5, 2.6M+ Data Points, 1000 Epochs, Batch Size 10k, 3x3 VoF Stencil

Chapter 6

CONCLUSION

The current results indicate that the neural network model struggles to reconstruct certain circular radii in the test dataset. But, the ease of data generation for training helps it to make the reconstruction of a few circular phase interface radii in the test dataset with lower error norms. Improvements in the area of machine learning on regression problems can make this approach a viable option in establishing a relationship between the radius of the circular phase interface and volume fraction fields. The model does have some outliers, which account for the bad predictions captured by the large error norm value. Looking at the performance of the neural network, the predictions for circular phase interface radii have a mean error norm value of 10^{-2}. The worst prediction error norm value is 10^{1} and the best prediction error norm value is 10^{-8}. Overall, this thesis work provides insight into the potential of using neural networks to reconstruct the phase interface in multiphase flows. Offering a complementary data-driven approach alongside the current interface capturing methods. From this thesis work, the following is concluded -

1. Circular phase interface randomly generated within a 1x1 domain, the neural networks give a range of good and bad predictions with lower model loss values but large error norms values as the error metric captures bad predictions

2. The error in the predictions of reconstructed circular phase has a log-normal ranging from 10^{-10} to 10^{1} error values

3. The model mean square error loss values converge around 10^{-3} for network architecture with convolutional (CNN) and dense (DNN) neural network layers

6.1 Future Work

To improve the neural network mapping of volume fraction field values with accurate phase interface predictions by -

1. Machine learning algorithms have an empirical approach and need to test the best optimal configuration for a regression-supervised problem from a space of convergence and model parameters

2. Scale the VoF stencil inputs to 5x5 or 7x7 stencils points with more data points for each circular phase interface

3. For the loss to converge, larger data sets with varying phase interface shapes can be used to train the model. For the higher sets of phase interface predictions can achieve closer to machine precision values

4. Hybrid approaches like modifying the loss function associated which can converge the loss values to closer to machine precision values to achieve non-gaussian error distribution over a range of test data. Referring to figure 6.1 the loss function can include another loss that calculated the errors involved in the physics partial differential equation being solved.

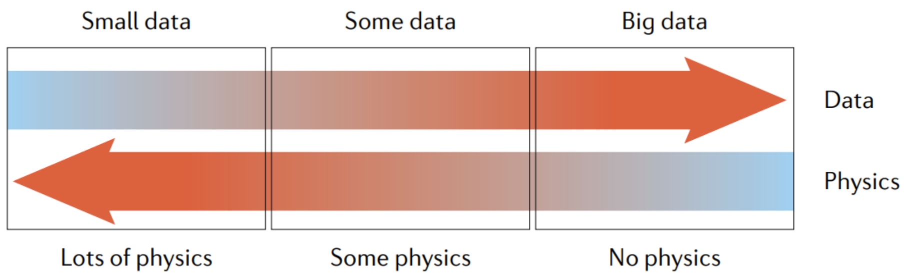


Figure 6.1: Physics Vs Data Consideration (Karniadakis *et al.* (2021))

REFERENCES

Bnà, S., S. Manservisi, R. Scardovelli, P. Yecko and S. Zaleski, "VOFI - A library to initialize the volume fraction scalar field", Computer Physics Communications **200**, 291–299 (2016).

Brunton, S. L., B. R. Noack and P. Koumoutsakos, "Machine learning for fluid mechanics", Annual Review of Fluid Mechanics **52**, 1, 477–508, URL `https://doi.org/10.1146/annurev-fluid-010719-060214` (2020).

Chollet, F., *Deep Learning with Python* (Manning Publications, New York, NY, 2017).

Clift, R., J. Grace and M. Weber, *Bubbles, Drops, and Particles*, Dover Civil and Mechanical Engineering Series (Dover Publications, 2005), URL `https://books.google.com/books?id=UUrOmD8niUQC`.

Francois, M. M., S. J. Cummins, E. D. Dendy, D. B. Kothe, J. M. Sicilian and M. W. Williams, "A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework", Journal of Computational Physics **213**, 1, 141–173 (2006).

Fricke, M., T. Marić and D. Bothe, "Contact line advection using the geometrical volume-of-fluid method", Journal of Computational Physics **407**, 109221, URL `https://www.sciencedirect.com/science/article/pii/S002199911930926X` (2020).

Glorot, X. and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", Journal of Machine Learning Research **9**, 249–256 (2010).

Gueyffier, D., J. Li, A. Nadim, R. Scardovelli and S. Zaleski, "Volume-of-fluid interface tracking with smoothed surface stress methods for three-dimensional flows", Journal of Computational Physics **152**, 2, 423–456, URL `https://www.sciencedirect.com/science/article/pii/S002199919896168X` (1999).

Herrmann, M., "Multiphase flow - overview", Center for Turbulence Research Annual Research Briefs (2008).

Hirt, C. and B. Nichols, "Volume of fluid (vof) method for the dynamics of free boundaries", Journal of Computational Physics **39**, 1, 201–225, URL `https://www.sciencedirect.com/science/article/pii/0021999181901455` (1981).

Karniadakis, G. E., I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang and L. Yang, "Physics-informed machine learning", Nature Reviews Physics **3**, 6, 422–440 (2021).

Melville, W. K., "The role of surface-wave breaking in air-sea interaction", Annual Review of Fluid Mechanics **28**, 1, 279–321, URL `https://doi.org/10.1146/annurev.fl.28.010196.001431` (1996).

Mirjalili, S., S. S. Jain and M. Dodd, "Interface-capturing methods for two-phase flows: An overview and recent developments", Center for Turbulence Research Annual Research Briefs **2017**, 117-135, 13 (2017).

Noh, W. F. and P. Woodward, "Slic (simple line interface calculation)", in "Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics June 28 – July 2, 1976 Twente University, Enschede", edited by A. I. van de Vooren and P. J. Zandbergen, pp. 330–340 (Springer Berlin Heidelberg, Berlin, Heidelberg, 1976).

Pilliod, J. E. and E. G. Puckett, "Second-order accurate volume-of-fluid algorithms for tracking material interfaces", Journal of Computational Physics **199**, 2, 465–502, URL `https://www.sciencedirect.com/science/article/pii/S0021999104000920` (2004).

Popinet, S., "An accurate adaptive solver for surface-tension-driven interfacial flows", Journal of Computational Physics **228**, 16, 5838–5866, URL `https://www.sciencedirect.com/science/article/pii/S002199910900240X` (2009).

Prosperetti, A. and G. Tryggvason, *Computational Methods for Multiphase Flow* (Cambridge University Press, 2007).

Qi, Y., J. Lu, R. Scardovelli, S. Zaleski and G. Tryggvason, "Computing curvature for volume of fluid methods using machine learning", Journal of Computational Physics **377**, 155–161, URL `https://www.sciencedirect.com/science/article/pii/S0021999118307046` (2019).

Sato, Y., M. Sadatomi and K. Sekoguchi, "Momentum and heat transfer in two-phase bubble flow—i. theory", International Journal of Multiphase Flow **7**, 2, 167–177, URL `https://www.sciencedirect.com/science/article/pii/0301932281900033` (1981).

Shaw, R. A., "Particle-turbulence interactions in atmospheric clouds", Annual Review of Fluid Mechanics **35**, 1, 183–227, URL `https://doi.org/10.1146/annurev.fluid.35.101101.161125` (2003).

Sirignano, W. A., *Fluid Dynamics and Transport of Droplets and Sprays* (Cambridge University Press, 1999).

Smith, L. N., "A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay", arXiv preprint arXiv:1803.09820 (2018).

Tryggvason, G., R. Scardovelli and S. Zaleski, *Direct Numerical Simulations of Gas–Liquid Multiphase Flows* (Cambridge University Press, 2011).