PREDICTION OF STRESS CONCENTRATION FACTOR FROM IN-LINE INSPECTION DATA USING CONVOLUTIONAL NEURAL NETWORKS

A Thesis

by

EMMANUEL VALENCIA

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee, Richard Malak

Committee Members, Douglas Allaire

Ivan Damnjanovic

Head of Department, Bryan P. Rasmussen

August 2022

Major Subject: Mechanical Engineering

ABSTRACT

It is becoming increasingly harder to install new oil and gas pipelines, the need for fuel at a reasonable cost is at an all-time high, and the public tolerance for pipeline failure is zero to none. To help existing pipelines maintain these rising energy demands and meet their fitness for service assessments, operators rely on in-line inspection (ILI) data to identify dent defects and calculate their stress concentration factor (SCF). These ILI runs typically identify a large number of dent defects, making it difficult for engineers to process individually. This study developed a Convolutional Neural Network (CNN) trained on 4,667 raw ILI data files containing dent shapes, along with a data pre-processing algorithm to convert ILI data into pseudo-images, to address the need of prioritizing dented pipeline segments based on their SCF. The final CNN model successfully predicted SCFs ranging from 1.04 to 10.69 with an RMSE of 0.418 and R2 of 0.929, therefore showing potential as the framework for a pre-assessment tool used by pipeline operators.

DEDICATION

"Fear of the LORD is the foundation of true wisdom.

All who obey his commandments will grow in wisdom.

Praise him forever!"

Psalm 111:10

I dedicate this thesis to my parents, who supported me throughout my entire college career, and reminded me that wisdom is putting knowledge to practice.

And most importantly, I thank God for His magnificent creation that I can study. May I never lose that curiosity and desire to learn throughout the rest of my life.

ACKNOWLEDGMENTS

I would like to thank Dr. Richard Malak, Chairman of my committee, for his continued support and brilliant insight throughout my graduate career. I was first introduced to Machine Learning in his class on Spring of 2020 – since then I have been greatly inspired to learn more. I would also like to recognize and thank my committee members, Dr. Douglas Allaire and Dr. Ivan Damnjanovic, for their guidance in previous classes and willingness to challenge me in my education.

I am greatly thankful to ADV Integrity, Inc. for funding my graduate studies and providing subject matter expertise on the content of pipeline in-line inspection (ILI). I extend my gratitude to the pipeline company that provided me with the ILI data necessary to complete this study.

This thesis was conceived within the first year of the COVID-19 pandemic. Despite the tumultuous time it has been, I am grateful to have found a community I may call my friends at Texas A&M University, and a community I may call my home at Grace Bible Church: Midtown.

Last, but certainly not least, I would like to thank my father, my mother, and my brother. My parents instilled in me a passion to lead, serve my community, and honor Christ. Without their relentless sacrificial love, I would not be where I am today. *Este es para todos ustedes, mi familia.*

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Richard Malak of the Department of Mechanical Engineering, Professor Douglas Allaire of the Department of Mechanical Engineering, and Professor Ivan Damnjanovic of the Department of Civil Engineering.

Rhett Dotson of ADV Integrity, Inc. supported with the dent assessment history covered in Section 2 and the development of the Data Smoothing in Section 4.2.3. David Santos, PhD candidate at Texas A&M University, provided insight on the machine learning model development covered in Section 2.1. All of the raw in-line inspection data was provided by a pipeline operator company that requested to remain anonymous.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was supported by a contract with ADV Integrity, Inc. for the financial assistance of the Master's graduate program.

NOMENCLATURE

ADV	ADV Integrity, Inc.
AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DL	Deep Learning
FEA	Finite Element Analysis
FFS	Fitness for Service
ID	Inside Diameter
ILI	In-Line Inspection
ML	Machine Learning
NDE	Non-Destructive Examination
NPS	Nominal Pipe Size
OD	Outside Diameter
PHMSA	Pipeline and Hazardous Materials Safety Administration
SCF	Stress Concentration Factor
UTS	Ultimate Tensile Strength
WT	Wall Thickness

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND MOTIVATION

It is becoming increasingly harder to install new pipelines, the need for fuel at a reasonable cost is at an all time high, and the public tolerance for pipeline failure is zero to none. These growing exigencies imply that the existing pipeline infrastructure is expected to last longer, deliver more product, and perform better in order to meet the rising energy demands. These factors enforce proper maintenance and threat detection in pipelines, an area where involving artificial intelligence (AI) stands the opportunity to benefit the oil and gas industry the most.

1.1 Oil & Gas Pipelines in the United States

The largest source of energy in the United States (U.S.) is petroleum, including oil and natural gas. Combined, they supply 65% of the energy used as of 2018 [4]. There exist four ways to move oil and gas around the country: truck, railroad, boat, and pipeline. As of 2018 in the U.S. alone, 100% of the natural gas and 70% of crude oil and petroleum products were shipped by pipeline [5]. The nation's pipelines are a massive, interconnected transportation system as shown in Fig. 1.1, that are made up of three categories: pipeline gathering systems used for transporting crude oil from the wellhead to the processing facilities; pipeline transmission lines for supply areas to markets; and distribution pipelines, which are most commonly used to transport natural gas to medium or small consumer units [6].

Pipelines have become the dominant means of transporting oil and gas due to their safe movement of the products at extraordinary quantities and reduced cost. The volume of energy products transported by pipelines is so significant, that "it would take a constant line of tanker trucks, about 750 per day, loading up and moving out every two minutes, 24 hours a day, seven days a week, to move the volume of even a modest pipeline" [4]. Similarly, this would be the equivalent of a railroad train with 225 tank cars each carrying 28,000 gallons of product. Essentially, a single pipeline can easily exceed the capacity of other forms of transportation. As of 2018, there were more than 2.6 million miles of pipelines in the U.S. alone [4].

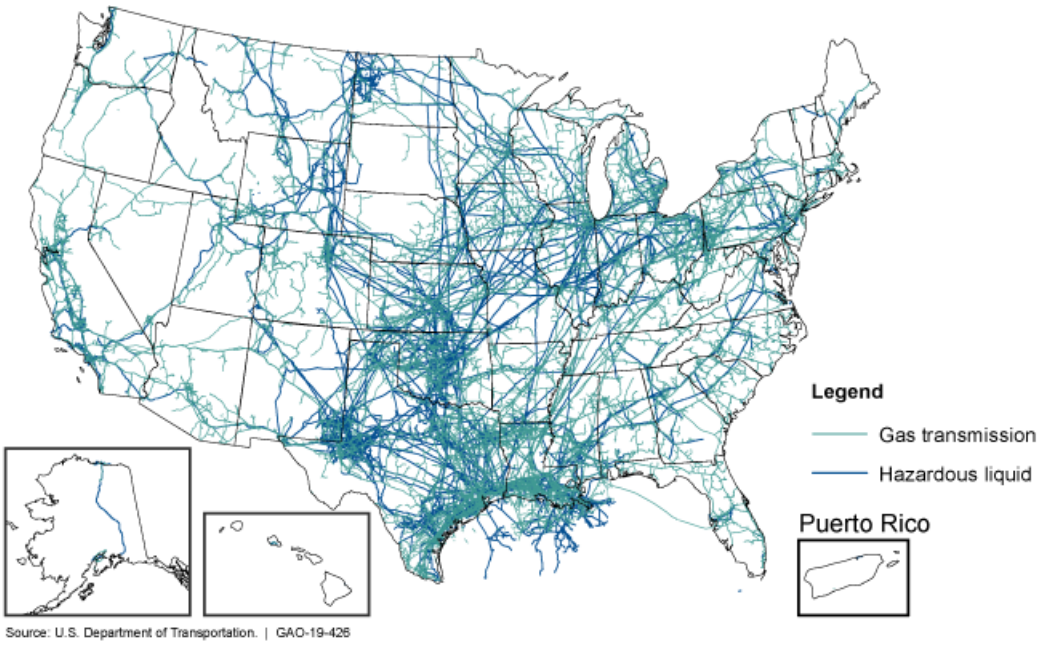


Figure 1.1: Hazardous Liquid and Natural Gas Transmission Pipelines in the United States, September 2018 [1]

Since the U.S. infrastructure heavily depends on oil and gas pipelines, it is critical that pipelines remain operational as long as possible and any necessary downtime be minimized. In a study performed by Enbridge[1], the company studied the impact of a simulated shutdown of a single but major pipeline going through the state of Michigan. This pipeline, known as Line 5, has been a vital piece of energy infrastructure since 1953 that has delivered the light oil and natural gas liquids that heat homes, fuel vehicles, and power industry. As of 2021, Line 5 transports up to 540,000 barrels per day (bpd), or 22.68 million U.S. gallons per day, of light crude oil, light synthetic crude, and natural gas liquids, which are refined into propane. To put this into perspective, Line 5 supplies 55% of Michigan's statewide propane needs. Therefore, shutting down this pipeline – whether for maintenance, failure, or decommissioning – would severely impact the economy. According to the study, the complete shutdown of Line 5 would result in $20.8 billion loss in economic activity and 33,755 lost jobs, among other major annual economic losses.

[1]2021 The Regional Economic and Fiscal Impacts of an Enbridge Line 5 Shutdown. Prepared for Consumer Energy Alliance by Weinstein, Clower & Associates. Report URL: `https://consumerenergyalliance.org/cms/wp-content/uploads/2021/05/CEA_LINE5_REPORT_2021_DIGITAL_FINAL.pdf`

But it does not need to be a complete shutdown to result in a significant impact, even a temporary shutdown could be catastrophic. The current average annual Brent crude oil price cost is $91.82 per barrel [7], therefore if Line 5 were to shut down for a single day in the year of 2022, Enbridge would lose $49.6 million per day on potential revenue. This does not account for the impact on the economy, nor the additional costs if the shutdown were incurred by a failure, which would exponentially increase the charges added for cleanup, restoration, and potential lawsuits.

Due to the growing demands of energy consumption, oil and gas production continues to increase. The International Energy Outlook of 2016 estimated that the worldwide consumption of fossil fuels would increase from 90 million barrels per day in 2012 to 121 million barrels per day in 2040 [8]. Unless more pipelines are installed, the existing infrastructure will have to work more efficiently in order to adjust to a greater load in capacity. The existing transmission system made up of high-strength steel pipelines is continuously moving large amounts of oil and gas thousands of miles from the producing regions to clients, with operating internal pressures typically ranging from 200 to 1,500 pounds per square inch (psi) [9]. This internal pressure applies a force on the internal lining of the pipe wall, as shown in Fig. 1.2. The resultant operating pressure of an oil and gas pipeline value depends on various factors, such as the pipe size and rating (i.e., the pipe diameter, material properties, and safety factor), the area of operation (i.e., the geographical landscape and urban population define the level of operation criticality), and variation in demand periods (i.e., time of day or year will cause fluctuation in product consumption, such that it will not always be completely steady-state).

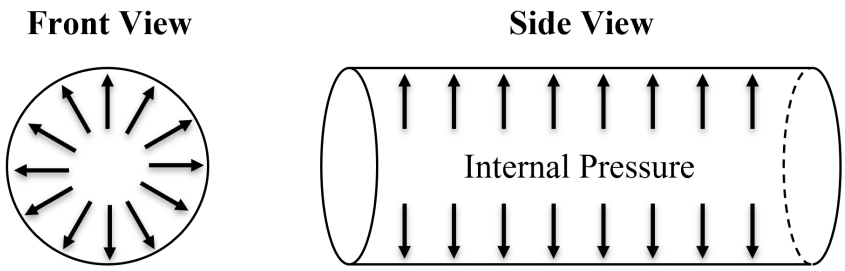


Figure 1.2: Front and side view of a pipe's internal wall shape with internal pressure.

The variability in internal pressure over time create cyclic loads that deteriorate the life of a pipeline. These cyclic loads, in addition to other events such as corrosion, natural force damage, excavation/third-party damage, faulty maintenance, and other external hazards to the pipeline [10], can deteriorate the reliability, safety and performance of the asset by generating defects along the pipe wall. Common defects that adversely affect a pipeline's integrity are dents, metal loss, punctures, and wrinkles. For the purposes of this thesis, pipe wall dents will be the sole focus when describing a pipeline defect.

Therefore, if a defect is identified during inspection, there are a series of assessments that must be made by pipeline operators to ensure regulatory bodies[2] that the condition of the pipeline is clearly understood and that proper future actions are planned [12]. Fitness-for-service (FFS) assessments are detailed, technical inspections and engineering evaluations performed to ensure the pipeline's ability to operate is in accordance with regulation standards, and provided that the conditions to reach failure are not reached. As oil and gas assets continue to age, a FFS assessment offers a comprehensive analysis of a pipeline's current and future integrity [12].

There exist various methods to assess a pipeline's FFS and reconfirm its structural integrity safety. However, there are only four (4) approaches permitted by federal minimum pipeline safety regulations to assess pipeline integrity [13]: 1) in-line inspection (ILI), often called "smart pigging"; 2) hydrotesting; 3) direct assessment for external corrosion; and 4) an equivalent technology that must be noticed and demonstrated to the regulatory bodies before it is used. Each of these assessments have their specialized applications and benefits, but for the purposes of this study, the remainder of this thesis will focus only on ILI as the method of assessment for FFS. The following section summarizes the current use of ILI tools for pipeline maintenance, but a thorough review of pipeline dent assessment procedures is covered in Section 2.2.

[2]Interstate pipelines are managed by the Federal Energy Regulatory Commission (FERC) and the U.S. Department of Transportation (DOT). Once pipelines are operating, the DOT's Pipeline and Hazardous Material Safety Administration (PHMSA), acting through the Office of Pipeline Safety (OPS), regulates, monitors and enforces safety [11].

1.2 In-Line Inspection

The use of in-line inspection (ILI) to assess pipeline integrity has become increasingly popular among pipeline operators, such that over the past decades, many pipelines have been designed and installed with the mindset of handling them [13]. These tools operate on a periodic maintenance inspection schedule and usually on a pipe with internal flow. The use of ILI is commonly employed as a non-destructive evaluation (NDE) to assess the structural health of a pipeline [14]. They are complex devices consisting of sensors that gather data about the internal wall of a pipe, and different ILI tools are designed for various types of specific pipeline threats, such as corrosion, dents, and cracks. Because of their diversity of purpose, they can vary in length, complexity, and size. This indicates that a single ILI tool may not be applicable for all pipes or purposes, but consequently, certain ILI tools are better suited for a specific task.

The most common types of ILI tools are seperated into three categories: 1) magnetic flux tools, 2) ultrasonic tools, and 3) caliper/mapping/geometric tools. While each tool has their strength and weaknesses, this thesis will focus only on caliper tools. Caliper and geometry tools are used interchangeable when referencing to ILI caliper tools. An example of an ILI tool with a segment of caliper arms is shown in Fig. 1.3, where multiple mechanical arms are closely packed together making a complete, 360° revolution around the tool. These mechanical arms measure the distance from the center of the pipe (or the pipe axis) to the inside of the pipe wall (or the internal radius), essentially creating a contour of the internal surface. The figure displayed is an example of an ultra-high resolution ILI tool, indicating that it is using a highly dense sensor array that covers the complete circumferential lining of the pipe in order to best represent the true geometry of the pipeline internal wall [2].

The data gathered from an ILI is then used for the pipeline's FFS assessment. Engineers who are highly experienced with evaluating ILI data are then capable of identifying dent defects amid the pipe wall contours and prioritizing them based on their perceived threat and risk to the pipeline integrity [12]. If dents are identified, then the FFS assessment involves determining the remaining fatigue life of the dents. Fatigue is a degradation process that promotes damage and potentially

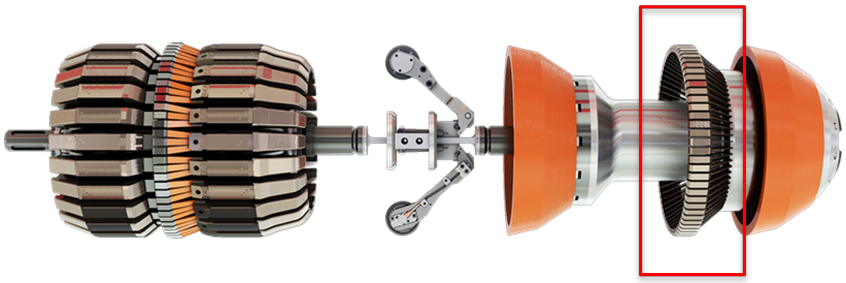


Figure 1.3: Example of a high-resolution ILI tool with a caliper arm segment (in red box) [2].

failure of a component when it is subjected to repeated cyclic loading [15], therefore as the pipeline continues experiencing variations in internal pressure, a dent defect existing within the pipe will deform and increase in severity over time. This remaining fatigue life of dents is estimated by using two critical components: by calculating the stress concentration factor (SCF) of the dent shape, and by incorporating the pipeline's pressure cycle history. A SCF on its own does not embody a complete assessment of a pipe's FFS assessment, but it administers a fast pre-assessment and prioritization of dents.

1.3 Stress Concentration Factor

A stress concentration factor (SCF) is a dimensionless value representing the ratio of the highest stress to the nominal stress of a mechanical component. It is located in a region of significantly greater stress, otherwise known as a stress concentration, which is typically in a point of discontinuity where a shape abruptly changes. These stress concentrations occur as a result of irregularities in the geometry [16]. In this case, a pipeline dent typically becomes a point of stress concentration. It becomes important to understand the factor of stress, or SCF, in a dent in order to properly assess the integrity of a pipeline.

These SCFs can be determined through experimental, analytical, or computational methods. Experimental methods quickly become incredibly expensive, as it requires removing the pipeline segment containing the dent and testing it to failure to measure the stress and strain of the concentration point. It is also not guaranteed, since the point of highest stress is not easily determined. Analytical methods can be used by measuring the dent geometry and applying the theory of elas-

ticity [16], which makes its own assumptions and requires ideal dent shapes. This is unrealistic for real-world usage where dent shapes are irregular and unique. However, computational methods tap into the power of finite element techniques in order to assess a dent's SCF. This involves generating a numerical model of the dent shape that is then exposed to stress simulations in order to determine a realistic stress concentration. Software utilizing finite element analysis (FEA) and computer-aided engineering such as Abaqus employ engineers with the skills necessary to complete the FFS assessments mentioned in the previous section. An example of an FEA model of a pipeline dent is demonstrated in Fig. 1.4 below.

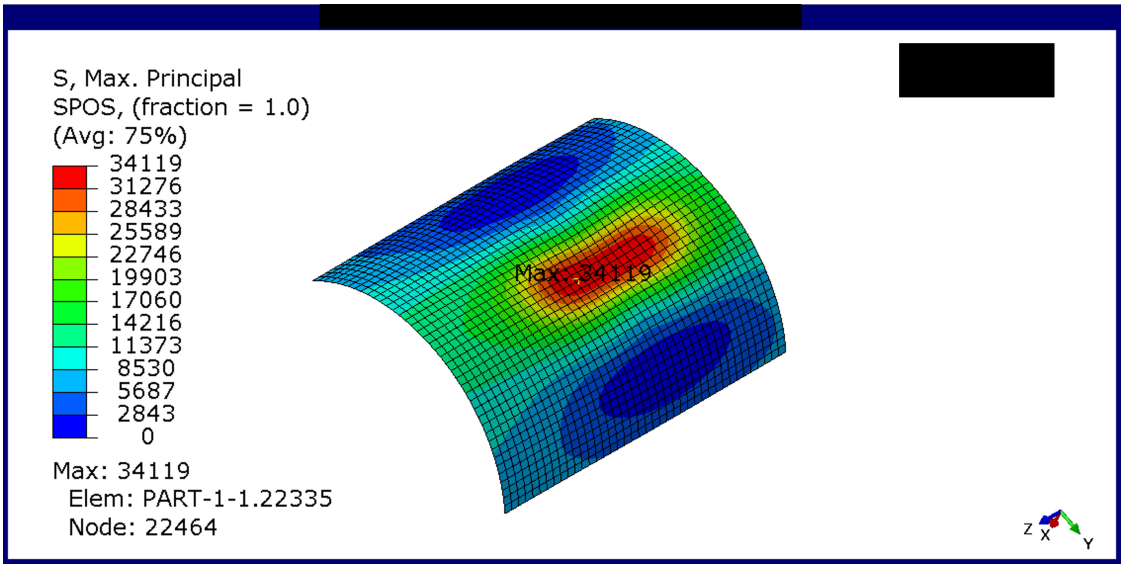


Figure 1.4: Example of a pipeline segment FEA model with Maximum Principal Stress values and a SCF of 3.41 in dent region.

Considering the significant number of dents that potentially exist on a single pipeline, performing a FFS on multiple pipelines can quickly become time costly and financially expensive. According to Rhett Dotson [17], Chief Engineer at ADV Integrity, the cost for SCF assessments can range from $250 to $1,000 per dent. Because of this, some technologies have been developed in the oil and gas industry to streamline the process of assessing the SCF of pipeline dents, such as the FE-DAT created in collaboration with Stress Engineering Services and ROSEN [18]. While

such tools exist to cut down on costs and expedite the process, they still require access to a license from Abaqus (costing about $20,000 per year), an engineer employed with expertise on FEA, and a computer with significant processing power to reduce the analysis times from days to minutes. This is notable progress, but it only sets the momentum to find improved ways to determine pipeline dent SCFs and integrate new problem-solving methodologies with AI.

1.4 Fast and Effective SCF Assessments

There exist massive databases over the pipeline infrastructure in the U.S. alone. According to the U.S. Department of Transportation's Pipeline and Hazardous Materials Safety Administration (PHMSA), pipeline operators are "required to submit annual reports to PHMSA... [including] information such as total pipeline mileage, facilities, commodities transported, mileage by material, and installation dates" [19]. However, it does not contain data from ILI runs, as that information is not required to be disclosed by operator companies. But pipeline dent defects are not unique to a single pipeline operator – every single oil and gas operator faces this challenge. Considering that a pipeline failure not only hurts the responsible operator, but the entire oil and gas pipeline industry, it would be beneficial for pipeline companies to collaborate in this area as part of their risk mitigation strategy. There is a need for joint partnerships in developing pipeline dent databases to use for fast and effective SCF assessments. As data collection accelerates, performing SCF assessments can become more of an AI Machine Learning (ML) challenge than an empirical problem.

Using ML to predict the SCF of a pipeline dent creates various opportunities for fast and effective assessments that contribute to a pipeline's overall FFS. To the best of our knowledge, the current solutions in the oil and gas industry do not offer an immediate SCF assessment of pipeline dents. These solutions depend on an external system that reviews and processes the ILI data, then calculates an SCF per dent. As explained in Section 1.2, an SCF assessment alone is only half of the information necessary for a proper FFS as the pressure cycle history is also necessary. Nevertheless, integrating software to existing ILI tools using ML to predict SCFs would serve as a pre-screening tool capable of giving operators guidance to prioritize dents. This level of prioritization can help save time, money, and mitigate potential catastrophic failures.

1.5 Research Goal and Objectives

The work described in this thesis is motivated by the need to develop a cost-effective, dependable, and quick assessment of pipeline dent SCFs, thus giving insight into risk-based decisions. This thesis has the goal of developing a pre-assessment tool used on pipeline dents that predicts SCF values based on raw ILI data. This goal is fulfilled by working on the following objectives:

- Develop a pre-processing algorithm that prepares raw ILI data into a usable format for training, validating, and testing a ML model.
- Build a ML model using Convolutional Neural Networks (CNN) that best predicts SCFs.
- Demonstrate the intended implementation of the final ML model and its potential impact for real use scenarios.

1.6 Thesis Outline

This thesis is organized into seven (7) sections: **Introduction and Motivation** covers the motives, research goal and objectives behind this thesis; **Literature Review** briefly examines existing peer-reviewed papers demonstrating the current assessments in this area, and the lack thereof; **Overall Methodology** introduces a high-level flow diagram of the overall procedure as well as the metrics for success; **Data Collection and Pre-Processing** carefully covers the extensive process of selecting the raw ILI data and pre-processing it into a usable format for the purposes of training, validating, and testing a potential ML model; **Machine Learning Model Development** explains the exhaustive tuning of hyperparameters to achieve a final ML model with suitable SCF value prediction capabilities; **Results and Discussion** provides commentary on the final product and tests for generalization; **Conclusion and Recommendations for Future Study** summarizes the essential findings of this thesis and presents a trajectory for future research.

2. LITERATURE REVIEW

There exists a large body of work on the use of empirical, analytical, and numerical methods to assess the stress concentration factor (SCF) of pipeline dent defects. Over the past few decades, the parameters of high correlation with dent severity have changed, and authoritative standards providing guidance on estimating the remaining life of a pipeline have been updated accordingly. Industry-ruling standards, such as the American Society of Mechanical Engineers (ASME) B31.8 "Gas Transmission and Distribution Piping Systems" [20], indicate what is accepted for these assessments, but leave space for subjective implementation. As a result, various methods have been developed within the oil and gas industry to attempt at estimating the SCF of a pipeline dent.

The presence of these geometrical discontinuities, otherwise dent defects, on a pipe surface act as stress concentration points that can be represented with a SCF relative to the nominal stress experienced by the rest of the pipe wall. These dents are inevitable in real pipeline components as assets deteriorate over time due to cyclic internal pressure loads, excavation damage, soil movement, and other external hazards to the pipeline [10]. Since more than half of all United States (U.S.) pipelines are almost 50 years old [19], it is becoming increasingly necessary to understand which of these dents have a greater risk of failure, and thus, prioritize the appropriate response to assess them based on their level of severity. With over 2.6 million miles of pipelines in the U.S. alone, effective and reliable assessments of these dents becomes a high priority.

In performing the literature review for this thesis, the reviewed documents were grouped into 4 different categories that were within the scope of this thesis:

- Brief explanation of Machine Learning (ML) and a review of different ML methods.
- History of pipeline dent assessment procedures for determining the SCF of a dent.
- Current leading development of ML models for mechanical assessments.
- Using regression convolutional neural networks (CNN) to make predictions from image data.

2.1 Review of Machine Learning

In the recent years, artificial intelligence (AI) has been solving various challenges of the oil and gas industry, and one of those major challenges has been defect detection and enhanced quality assurance [21]. The use of AI addresses the effort to automate intellectual tasks normally performed by humans. Many incredible assortments of real-world applications exist under the umbrella of AI, but this study reviews the use of machine learning (ML) which arises from the question: could a computer go beyond what we know how to order it to perform and learn on its own how to perform a specified task? [3]. As shown in Fig. 2.1 below, classical programming (otherwise known as symbolic AI) has an understood set of explicit rules for manipulating data in order to achieve a series of answers, while ML uses a set of data and examples of expected answers in order to derive the set of rules needed for that specific application – essentially *learns* the rules for the data.

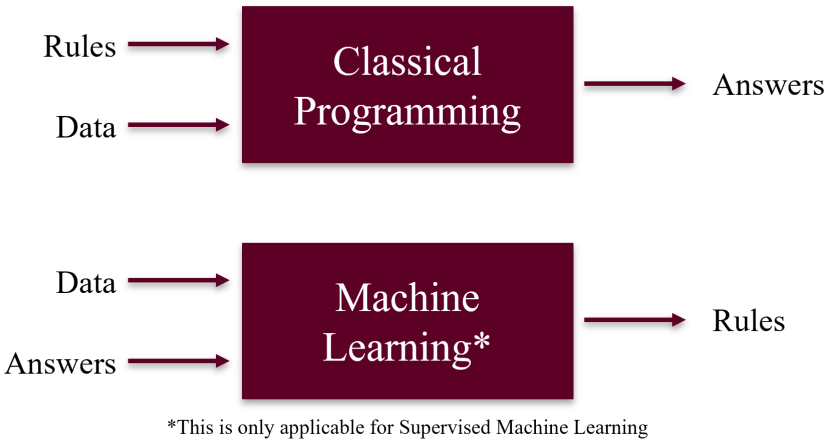


Figure 2.1: Comparison between a classical programming vs machine learning approach.

To develop a ML model for this study, there were three things needed: input data points, examples of the expected output or target output, and a way to measure whether the algorithm is doing a good job [3]. As will be covered in Section 4, all of the input data points and target outputs were prepared and organized in order to begin training a ML model. The goal was to develop a ML model capable of predicting the stress concentration factor (SCF) from raw in-line inspection

(ILI) dent data. For a model to be considered suitable, it must have met the metrics for success covered in Section 3.2. The objective of this section is to provide a brief explanation for methods of ML, such as Deep Learning (DL).

2.1.1 Artificial Neural Networks

An artificial neural network (ANN) is a computational model and subset of ML that was established based on the characteristics of a natural neural network. Its structure is composed of multiple computational units called nodes, or artificial neurons, that are stored in node layers, containing an input layer, one or more hidden layers, and an output layer. In the model, the nodes are interconnected to each other by means of unidirectional or bidirectional communication links [22], each with an assigned weight and then a single bias, or threshold. If the output of any individual node exceeds the specified threshold value, it activates the node, permitting data to transfer to the next layer of the network. This activation is done by an activation function that is applied to the summation of weighted inputs, which represent information being used by the network. A representation of a single node is shown in Fig. 2.2. On the left side of the image are the input nodes, which consist of N independent variable inputs, x_i for $i = 1 \dots N$, and one bias input, $x_0 \equiv 1$. This bias input defines the y-intercept of the linear regression. The connecting lines from the input nodes to the single hidden layer contain the weights, w_i for $i = 1 \dots N$, and one bias weight, w_0. In the hidden layers, all of the weighted inputs are summed and pass through an activation function to produce an output. If that output exceeds a given threshold, it activates the node and passes the data to the next node in the network. For this example, there is only one artificial neuron displayed with a single scalar output. This is an example of supervised learning as the ML model will learn to map the input data to known targets, which are often manually done by humans.

A mathematical representation of an artificial neuron d would be as follows (eq. 2.1):

$$\hat{y}_d = \text{act}(net) = \text{act}(w_0 + \sum_{i=1}^{N} w_i x_i) = out \tag{2.1}$$

where act() is the activation function and $\hat{y}_d$ is the node's (d) predicted outcome, which is scalar

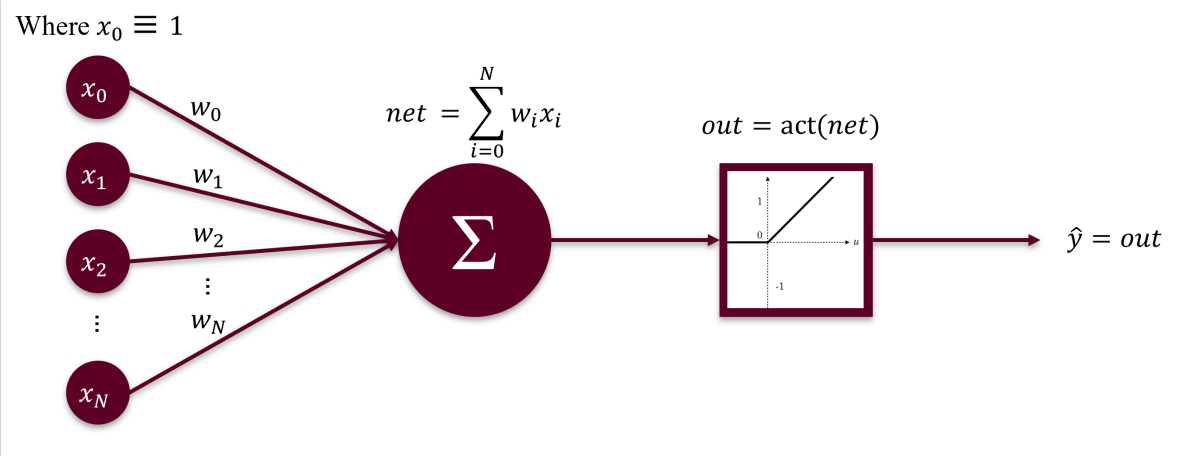


Figure 2.2: The model of a single node, or artificial neuron, used for neural networks.

value out. This prediction is compared to the target value, y_d, resulting in an error. In order to curve fit a single node, an error model and a procedure for updating weights based on error are necessary. With each training example, the weights and bias, which serve as the parameters of the model, are adjusted to gradually converge at the minimum of the determined error model (otherwise known as cost or loss function). Such an example is the Mean Squared Error (MSE) as seen in eq. 2.2, which takes the averaged squared difference between the model's predictions and target values. Taking the root gives the Root Mean Squared Error (RMSE), which follows the same units of SCF.

$$MSE = \frac{1}{D}\sum_{d=1}^{D}(y_d - \hat{y}_d)^2 \tag{2.2}$$

where d is the index of training inputs and targets, for $d = 1 \ldots D$. The advantage of using MSE is that it puts larger weight on larger errors due to the squaring of the function, thus ensuring that the trained model has no outlier predictions with huge errors.

Many activation functions exist, such as sigmoid, arctan, tanh, rectified linear unit (ReLu), and so on. This study mainly used ReLu activation functions as shown in eq. 2.3. These functions are meant to zero out negative values and has a range of $[0, \text{inf})$, whereas an activation function like sigmoid "squashes" arbitrary values into the $(0, 1)$ interval and is typically used to assign a probability. Since the desired output for this study is not a class label, a probability is not necessary.

$$ReLu(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases} \tag{2.3}$$

The loss and activation functions used are examples of **hyperparameters** of the model (different from model parameters, which are the weights and bias). These hyperparameters involve the settings for the model architecture (e.g., number of layers, number of nodes per layer, activation functions, etc.) and the settings for compiling the model (e.g., loss function, metrics, learning rate, epochs, etc.). The metrics measure whether the algorithm is doing a good job. This is different to the loss function, which is the measurement of performance on the training data used to steer the network in the right direction, i.e., metrics is for observation, loss function is for correction. A metric used in this study was the Mean Absolute Error (MAE), as shown in below eq. 2.4

$$MAE = \frac{1}{D}\sum_{d=1}^{D} |y_d - \hat{y}_d| \tag{2.4}$$

Other useful hyperparameters are: epochs, the number of iterations over the training data; learning rate (or step size), the amount that the weights are updated during training; and batch size, the size of input data batches on which to train the model before updating the weights [23].

These are adjusted to change the performance of a model, but the true difference between models is found at the architecture used and its ability to handle data. Different architectures are needed for different purposes; while one model may be excellent for classifying images, another would be better for predicting a scalar value from the input data. The following section briefly covers the type of architecture used and the application of these hyperparameters to tune the model.

2.1.2 Deep Learning

Within the umbrella of ML, there is a subset field called Deep Learning (DL) that specializes on models that learn representations from input data with an emphasis on learning successive layers of increasingly meaningful representations. As implied with the name, the "deep" in DL refers to the depth of layers in a neural network. Therefore, a neural network consisting of more than

3 layers (i.e., 1 input layer, 1 hidden layer, and 1 output layer) can be considered a deep learning algorithm [24]. As shown in Fig. 2.3, DL is a subset of Neural Networks, which is a subset of ML, and which is a subset of AI.

The use of DL is resembled as a "multistage information-distillation operation, where information goes through successive filters and comes out increasingly purified" [3], that is, useful for a specific task. A huge benefit of DL is that it makes problem-solving much easier due to its capability of automating one of the most crucial steps in a machine-learning workflow: feature engineering.

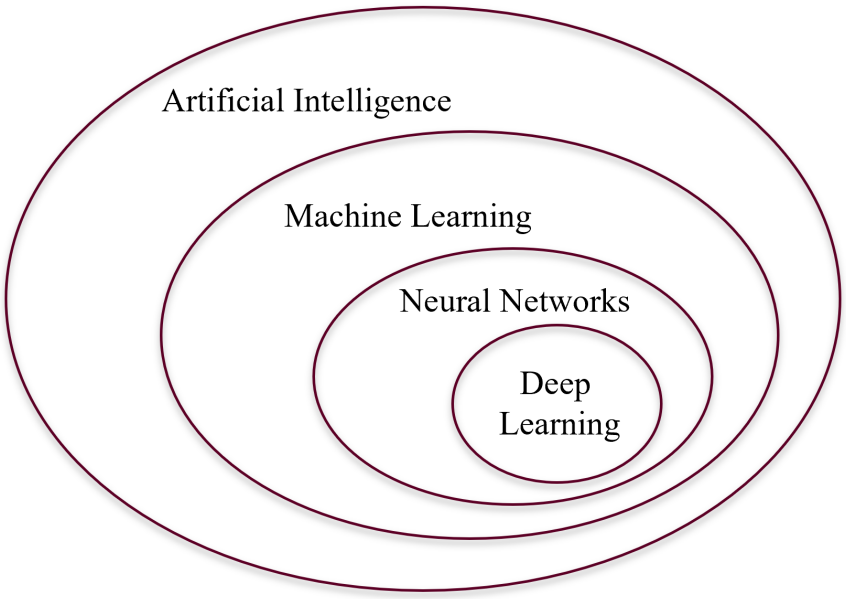


Figure 2.3: Umbrella of Artificial Intelligence with subsets of Machine Learning, Neural Networks, and Deep Learning.

Feature engineering is the initial effort of making input data more amenable to processing, therefore improving the performance of the ML models. This involves selecting, manipulating, and transforming raw data into features that can be used as any measurable input in the model [25]. For example, it could be the color of an object, or the contour of a shape. Traditionally, this can be part of the data pre-processing (as covered in Section 4.2), or it could be an additional step

taken to transform the data before its input to the model. Either way, it is an important step that can almost guarantee better results. Using DL greatly simplifies this step in the ML workflow, and can extract additional features from complex data that may have not been intuitive to a human.

2.1.3 Types of Deep Learning Architectures

This study saw significant value in implementing a DL architecture due to their ability of taking unknown elements to extract features, group objects, and discover useful data patterns [26]. Due to the non-linear relationship of the ILI dent shapes and their corresponding SCF values, a DL architecture was expected to perform well for this problem as they learn by example and through experience, thus discovering a relationship among the input variables that is difficult to understand. Some of the DL architectures considered were Multilayer Perceptrons (MLPs), Generative Adversarial Networks (GANs), and Convolutional Neural Networks (CNNs). It was quickly discovered that MLPs would not work sufficiently well, and due to the MLP's architecture containing fully connected layers, it would quickly grow into a very high number of parameters, thus running much less efficiently than comparable architectures. This study did not manage to implement GANs, but observed potential for it in future studies and is mentioned in Section 6.1 due to its ability to create new data instances that resemble the training data. However, a concern for that involved the possibility of making the training data even more biased to specific dent types and not generalize well with the world-wide population of dents. After all, as covered in Section 4.1, this study involved a limited dataset of 4,667 data points which are only representative of 8 distinct pipeline segments, out of the hundreds of pipeline segments in the United States alone.

2.1.4 Convolutional Neural Networks

Convolutional Neural Networks (CNN) is a DL architecture that has outperformed almost all of the state-of-the-art algorithms in different fields of computer vision starting with image classification [27]. The difference between another deep neural network and a CNN is the addition of the convolution layer, which learns local patterns throughout an image. As explained in [3], this gives convolutional networks two interesting properties: the patterns they learn can be translated

to anywhere in the image, and they can learn spatial hierarchies of patterns. These learned patterns can be recognized anywhere without needing to relearn the pattern if they appear in a new location. Additionally, these small local patterns can be combined through the layers in order to make features. As observed in Fig. 2.4, the first layer after the input image of a cat reveals several patterns, such as lines, curves, and edges. However, the second layer will learn "larger patterns made of the features of the first layers" [3]. This is easy to visualize for a trivial classification problem such as with a cat, but it becomes incredibly complex and useful for dealing with arbitrary patterns and features on an ILI dent shape.

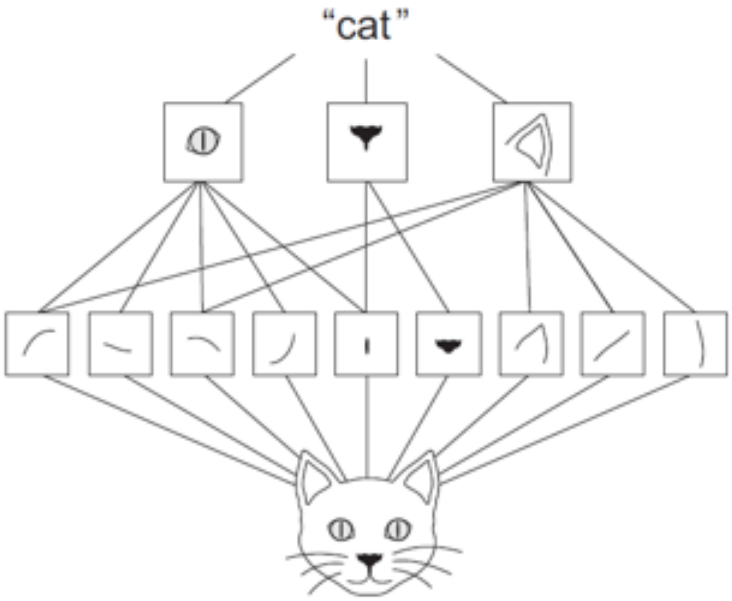


Figure 2.4: Display of feature extraction when classifying a cat. Courtesy of [3].

The architecture of a CNN is comprised of two parts: it starts with a series of convolutional and pooling layers, and then they end with densely connected layers [3]. The convolutional process involves the feature extraction to separate and identify various features of the image, while the pooling layers are occasionally introduced in between convolutional layers to reduce the spatial size of the feature maps and hence to reduce the number of parameters to be learned in the network [27]. Once the first part is completed, the 3D tensor outputs are flattened to 1D, and then passed through fully connected (Dense) neural layers as described in Fig. 2.2.

2.2 History of Pipeline Defect Assessment Procedures

There are various methods that oil and gas pipeline operators use to perform dent assessments. Federal regulatory standards require repair of dents with depths exceeding 6% of the pipeline diameter, however, failure has been known to occur with depths even less than 3% of the pipe diameter [15]. Therefore, dent depth alone is generally not an accurate predictor of the dent severity for fitness-for-service (FFS) assessments. Over time, other direct measurement parameters have been considered, including dent length, dent width, and ratios between these. These measurements are more characterizations of the dent features, as demonstrated in Fig. 2.5. In this example, two similar pipelines contain a dent of equal depth, but with varying length. This has driven the oil and gas industry to new methods that best closely represent the actual shape of pipeline dents.

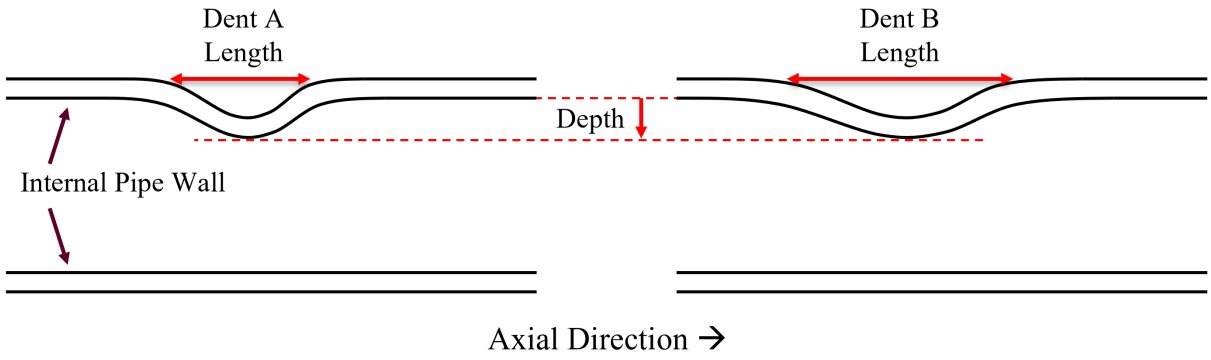


Figure 2.5: Comparison between two pipeline dent shapes containing the same depth but varying dent width. In this comparison, Dent B has a larger width than Dent A.

According to [28], gas operators prefer strain-based assessments, while liquid operators typically address dents meeting depth-based criteria. Some of the depth-based analytical solutions, such as the industry accepted method PRCI MD-4-9, account for "variations in diameter, wall thickness, and dent curvature" [28] in order to have an improved characterization of the dent shape for the remaining life assessment. However, these type of analytical solutions were developed using idealized dent shapes and finite element modelling, which are not a realistic representation of the real-world pipeline dent population which exhibit complex shapes or interacting dents. Dotson et al. [28] noted that rising methods "require full circumferential caliper data" in order to most accurately assess a pipeline dent.

In a follow-up study, Dotson et al. [29] doubled-down on the need to fully understand the three-dimensional shape of a pipeline dent in order to best assess the remaining life of the defect. Recent works reviewed in [29] demonstrated that dent depth alone is not the strongest indicator of dent severity. While this study was primarily focused on ILI measurement variability and "its impact on dent strain and remaining life assessments", it recognized that data filtering and smoothing may impact these assessments, but did not consider those influences to be significant sources of variation. Since data smoothing can be quite subjective and up to the discretion of an experienced engineer, it would be beneficial to examine the contribution of smoothing algorithms to the variation in dent assessments. According to ASME B31.8 [20], suitable data smoothing techniques are encouraged to minimize the errors induced by random error inherent with raw caliper ILI data. Yet, other approaches have attempted to estimate the strain in a dent by calculating the longitudinal and circumferential curvature of the dent shape [30].

As the technology employed in ILI tools has improved over the years, pipeline assessments have identified dents that were previously below the detection and reporting levels. In a study performed in 2010 by Race et al. [31], they developed an algorithm capable of prioritizing the dents for repair based on ILI results. Their strategy in the development of the algorithm took into consideration that it should "be easy to use with measurements that can be readily made in the field or with ILI tools, not require intensive and expensive FEA... [and] recognize the key parameters that contribute to stress" in the dent [31]. The criteria was based on the dent depth characterization, demonstrated as the percent of the outside diameter, and does not consider other features of the dent shape. Additionally, the results of this study required an ILI inspection to be coupled with magnetic flux leakage (MFL) and ultrasonic (UT) technology, and not just caliper data, in order for it to be a full dent assessment.

But comprehensive studies over the history of pipeline dent assessments point to the need of understanding the complex dent shapes in order to apply models to generalized cases. BMT Fleet Technology Limited performed a study [15] to develop a methodology for estimating the remaining life of pipeline dents based on 3D, non-linear dent shapes. In their findings, they recognized that

future studies should consider complex dent shapes in order to validate the dent models against real-world complex shapes. Many other studies have been performed to understand the impact of improved dent shape and geometry on a pipeline's integrity assessment [32, 33, 34, 35]. Therefore, there is area for improvement when it comes to developing methods for assessing pipeline dents that do not degenerate the dent shape only to a series of curve fits or parameters. This study aims to consider the entire dent shape to the closest representation of the real, physical deformation.

2.3 Development of ML Models for Mechanical Assessments

The use of ML to predict SCFs has been studied in multiple works that derive from the analytical methods of assessing a defect based on key dent features. Several of these studies employed the use of ML to solve highly-complex formulations to understand the relationships between these features. They were dependent on feature extraction of the pipeline dent data, which involves manual examination of the characteristics that will become the driving factors for the SCF prediction, and essentially, the input data to train, validate, and test a ML model.

Several studies exist that observed features with complex geometries that are necessary components of a mechanical system such as welded joints, unlike the undesirable defects of pipe wall dents. In a study by Dabiri et al. [22], they observed the geometry parameters of T-welded joints in order to train an artificial neural network (ANN) -based model and predict the SCF of the weld. The use of an ANN was proposed to fill a gap where empirical equations are not able to accurately calculate a SCF for a given weld profiles. In this study, the welded component acted as the geometrical discontinuity that resulted in a stress concentration. The weld components varied in geometry, and the study considered 6 different parameters: thickness, weld leg size, toe radius, undercut depth, connection height, and flank angle. This process of selecting specific parameters for consideration serves as an example of feature engineering to develop data appropriate for training. The study used Finite Element (FE) models as the source of truth data for SCF calculation. The ANN developed in this study consisted of a Multilayer Perceptron (MLP) with 4 inputs, 2 hidden layers, and 3 and 2 units in the first and second hidden layers, respectively. The ANN was able to predict the SCF values compared to the FE models with a coefficient of determination close to

100%. However, for certain weld profile configurations were outside the ranges implemented in the ANN-based model, the percentage error of the SCF predictions were between 12.5% to 26.7% compared to the FE model. This study relied on feature extraction and was limited to only select parameters that can apply only to specific welded joints, which in this case were T-welded joints.

Wang et al. [36] used an Extreme Learning Machine (ELM) to predict the fatigue SCF of a material notch. The model had 7 parameters that served as the inputs to the model: tensile strength, yield strength, fatigue strength, theoretical SCF, notch root radius, sample size and notch fatigue limit. The model output was the fatigue SCF of 46 different kinds of metal materials. Due to the complexity of the input features, and the lack of evident correlation between them and the output, using the ELM model was an unconventional solution. The relative error was used to evaluate the predicting performance, which indicated the uncorrelated degree between the predicted and the experimental fatigue SCF. This quantitative assessment was compared between different models, with the most accurate ELM model having a mean value of 4.63% relative error. Another study also used ELM in addition to back propagation neural networks (BPNN) and curve fitting models for their material assessment. Through this method, Raja et al. [37] managed to predict the fatigue crack growth behavior in ultrafine grained Al 2014 alloy.

While some testing involved observing relevant parameters among dent samples, other studies used feature selection upfront to generate dent shapes. Ji et al. [38] used Least-Squares Support-Vector Machines (LS-SVM) to predict the SCF of isolated elliptical corrosion pits on buried pipe using 3-D finite element analyses (FEA). The corroded pipe samples were generated using FEA and following the predetermined feature combinations. The LS-SVM models were developed to predict the SCF with reasonable agreement of similar corrosion patterns formed on the external surface of pipes. The models were trained using geometric properties of the corrosion pits: radius of the pipe, wall thickness, major and minor diameters, and depth of the elliptical corrosion pit.

Models did not need to rely on a single ML architecture to achieve best results. To classify corrosive defects on pipelines and determine the growth and failure time of a specific feature, Liu et al. [14] used a ML model to match corrosion from multiple ILI data sets. The ML model consisted

of an individual and ensemble use of various methods, involving a support vector machine, decision tree, and random forest. The models were trained on select features from the ILI data, such as odometer reading, defect orientation, length, width, and depth percentage, and achieved a matching accuracy higher than 90% when using the ensemble model. Their use of ensemble learning was a method of aggregating a group of predictors and selecting the class that got the most votes, resulting in a technique known as a majority-vote classifier or hard voting classifier.

Some studies held the objective of supporting the FEA process for certain mesh models by being able to make stress predictions of mechanical components with low fidelity mesh models. Such was the case with Yamaguchi et al. [39], who used neural networks to predict the stress concentration at fillets in order to minimize finite element mesh generation, effectively expediting a FEA using simple mesh models. Other studies also used ML and FEA to predict pipeline dent SCFs or the remaining lifetime by using other ML architectures such as extremely randomized tress [40]. These examples demonstrated the potential to enhance, if not replace, the use of computational expensive numerical assessments of FEA by applying ML models. A procedure that depends on high-fidelity mesh models will require significant time and computational power to generate and run those simulations. There are opportunities to expand the use of ML to minimize those direct and indirect costs to the procedure.

Studies demonstrate that ML models have the capability of predicting pipeline burst pressures [41, 42], the remaining fatigue life [40, 43], and stress concentration factors [32, 33, 22, 44], among many other characteristics of a mechanical feature. They also show that ML models can reduce the load on FEA assessments [39, 40], if not completely replace them. Therefore, the goal pursued in this study to determine a ML model that can delay, if not bypass, the need for costly FEA to predict an SCF as part of the dent assessment is plausible and has relevant research to support the legitimacy of this approach. This review also demonstrates the lack thereof of current solutions that focus on predicting the SCF of pipeline dent features. Moreover, the following section continues reviewing literature that uses ML models for mechanical assessments, but focuses on using CNN trained with image data instead of limited parameters.

2.4 Using CNN to Make Predictions from Images on Mechanical Assessments

The use of CNN is primarily tasked with enabling machines to view the world as humans do. This approach can take in an input image as data and be able to differentiate between one image from another, such as a human can distinguish a cat from a dog. Visual inspections are a practical way of assessing a mechanical system because it is non-intrusive to the procedure, unlike direct measurements or other assessments that may require to remove a mechanical system from its service. In the oil and gas industry, visual inspection is a form of non-destructive examination (NDE) procedure that can provide insight as to the integrity of a pipeline without removing the system from its service or placing a risk on the operators.

A study performed by Bastian et al. [27] created a vision based algorithm to detect for corrosion on pipelines. The process involved developing a custom deep neural network (DNN) architecture using convolutional neural networks (CNN) for the extraction of features specific for corrosion, and then a fully connected neural network to classify the images based on their features. The model was capable of classifying images based on 4 levels of corrosion intensity present on the pipe: high, medium, low, and zero level of corrosion. In addition, the model was extended to provide guidance on the precise localization of the corroded areas. The study consisted of 142,400 RGB images of water pipelines, and used a data-driven approach for feature extraction of the necessary information to classify corrosion. The custom built CNN model outperformed existing architectures with an accuracy of 98.2%. Such a model does not require the manual intervention for feature extraction, but was trained specifically just for corrosion features. Several uses of CNN models for feature extraction in images exist in other industries, including the medical industry: Xie et al. [45] used a CNN-based structured regression model to generate proximity patches as part of an effort for robust cell detection; and Ren et al. [46] also used a regression CNN to automatically assess the pediatric bone age from hand radiographs, with an accuracy comparable to human experts, as the average discrepancy was 5.2-5.3 months between clinical and automatic bone age evaluations.

Additionally, Oh et al. [47] used deep neural networks to predict the burst pressure of API 5L X-Grade dented pipelines. These dented pipelines were generated from an FEA based para-

metric study using a simulated indentation procedure, resulting in unconstrained, hemispherical, plain dents. The model was highly accurate when measured using mean absolute percentage error (MAPE), and was validated using 3 experimental burst pressures from prior studies. Pyo et al. [48] used a CNN model to extract features from hyperspectral images to estimate the concentrations of cyanobacteria. While this study did not involve features similar to oil and gas pipeline dents, it did experiment with using a variation of the AlexNet [49]. To reconstruct a digital hologram and understand its spatial location, Shimobaba et al. [50] proposed a regression CNN that can directly predict the depth position with millimeter precision.

Kantzos et al. [51] also used CNN to predict SCFs, but applied it to surface roughness to understand the underlying causes of fatigue crack initiation. These involved synthetically generated rough surfaces that were instantiated in a mechanical models. Their CNN model achieved a coefficient of determination, R^2 of 0.75 when using test images, and was specifically designed with simplicity in mind–i.e., bringing down the necessary learnable parameters. According to the study results, this allowed them to train the network on small datasets. On the flip side, another study used an analytical approach instead of a ML model in order to determine the SCF of machined surfaces [52].

Studies demonstrate that ML CNN models have the capability of predicting corrosion-like defect severity [27, 53, 54], identifying injurious defects on different ILI tools [55], and estimating a defect's SCF [51, 52], all while only using image data. All of these methods were examples of NDE, just as an ILI assessment, which are able to determine the integrity of a system without needing to destroy a sample. However, there was no literature discovered revolving the use of raw caliper ILI data to make SCF predictions as part of assessing a pipeline dent.

As a result of the literature review, to date and to the best of our knowledge, there are no studies on using CNN to predict the SCF of a dented pipeline using full circumferential, raw caliper ILI data. Additionally, more insight into ILI measurement variability influenced by data filtering and smoothing would be beneficial for future studies. Therefore, it is worth developing a CNN model that can be used for such an integral step of a pipeline's FFS assessment.

3. OVERALL METHODOLOGY

The remainder of this thesis will explain the methodology used to develop a machine learning (ML) model suitable for predicting the stress concentration factor (SCF) of a pipeline dent using raw in-line inspection (ILI) caliper data. The purpose of this chapter is to establish a high-level view of the research framework and clarify the assumptions made and existing constraints for the scope of this thesis. Figure 3.1 demonstrates this research framework using a flowchart method. The reader should note that this is an iterative process, where the output of this thesis is a software tool capable of predicting SCFs. Therefore, the closest resemblance of this iterative process is the **agile development methodology**: the requirements and solutions evolved an segments of the overall methodology were revisited throughout the entire span of the study.

The final ML model would satisfy the goals and objectives described in Section 1.5, which will be carefully reviewed in the selection process. The methodology framework is divided into the following steps: Data Collection, Data Pre-Processing, ML Model Development, Hyperparameter Tuning, and Final Model Selection. These steps are expanded into a high-level detail below.

First, the raw ILI caliper data was collected and inspected for use. The data was curated to select a sample that best represents a typical ILI run. Second, the data was pre-processed to prepare it for ML training, validation, and testing purposes. This involved data vectorization, normalization, smoothing, and conversion to an image format that best resembled a grayscale image. The data was then exported and organized into categories depending on the steps taken to prepare it, as well as dividing them into training, validation, and testing datasets. Third, a convolutional neural network (CNN) ML model was developed using a few different architectures, such as binary classification, multi-class classification, and finally, regression. Fourth, the hyperparameters of the ML model architecture were fine tuned to meet the desired performance of the model. The different ML models were tested by predicting SCFs and comparing them to the true SCFs, but by also observing the distribution of error. Lastly, the various ML models were compared and the one deemed most satisfactory was selected as the final ML model for the use in the software tool.

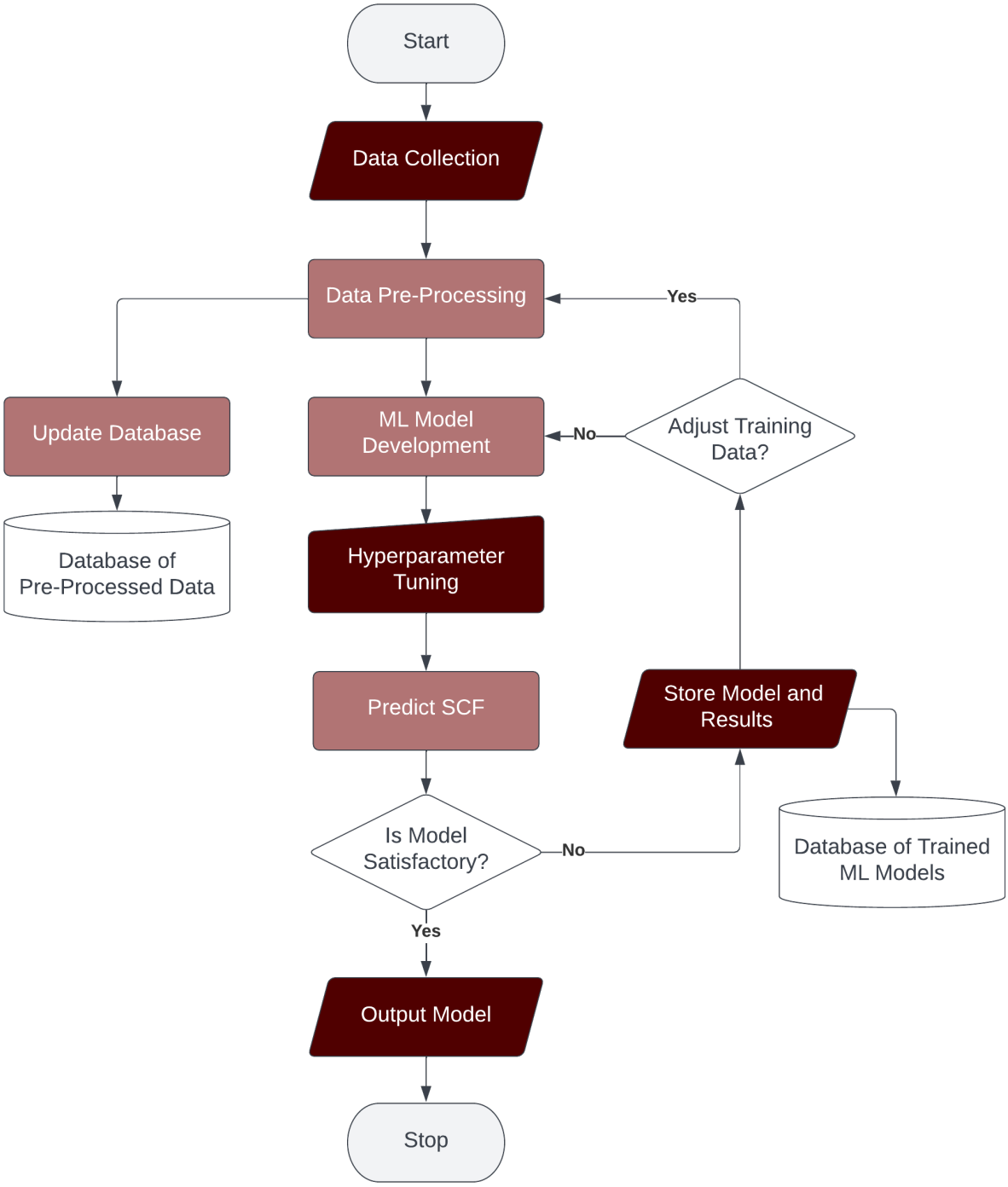


Figure 3.1: Flowchart of thesis methodology.

It is important to note that the final deliverable of this thesis will serve to make predictions within a justified margin of error in order to provide guidance in potential pipeline dent pre-assessment procedures. As it was explored in the Literature Review of Section 2, a proper ML model is **not** expected to make perfect predictions nor maintain a six sigma (99.999997%) rate of accuracy. A model that demonstrates good generalization will have an acceptable level of error in order to prevent overfitting, which will be explored in Section 3.4. For these reasons, an appropriate margin of error is established a result of this thesis and discussed in Section 5.

3.1 Assumptions and Constraints

Like any other real world study, there are assumptions made and inherent or desired constraints placed to limit the scope of the work. These add clarity to the value of the work done and demonstrate depth of understanding since no study can truly understand or control everything. This study consisted of multiple assumptions revolving around the validity of the data used and its application:

1. The raw ILI caliper data used in this study was assumed to be accurate and reliable depictions of internal pipe wall contours for oil and gas pipelines. While there is variability to consider due to the random and systematic error induced by each ILI tool, that is outside the scope of this study, and therefore the data collected is considered true as is.

2. The SCFs provided in conjunction with the raw ILI caliper data was considered as truth, regardless of the uncertainty in their calculations. Similar to the assumption made above, while there is variability to consider in the numerical assessment of SCFs and how well they agree with experimental results, that is outside the scope of this study.

3. The raw ILI caliper data measures the internal diameter (ID) of the pipeline, which is not an exact representation of the outside diameter (OD), due to pipe wall variation caused by material thickness, external/internal corrosion, and other external factors. However, consulting company ADV Integrity has determined that all full-scale testing has shown very close agreement with the finite element analysis (FEA) model results for thin wall pipes, so it is an acceptable assumption that the ID and OD shape of a pipe wall dent are matching.

4. Lastly, since all of the data used in this study was provided by a single pipeline operator and there is little to no access to additional ILI data from other oil and gas companies, then this study assumed that the raw ILI caliper data and SCFs were the best representation of global population statistics.

Similarly, the constraints of this study are described below:

1. **Only observing raw ILI caliper data.** This study will not review data from other ILI tools nor data that has already been processed into different formats. Simply put, the data should strictly be the direct output from an ILI run.

2. **Only observing dent defects.** This study will not consider other pipe wall defects and will be limited to dents only. Nevertheless, there are a broad range of dent shapes, such as adjacent, long, skewed, etc., but will all be considered only as dents.

3. **Only observing data containing a single dent.** There are ILI runs containing multiple dents in a single exported file, but as described in detail in Section 4.1.2, these files are omitted from the rest of the data.

4. **Only observing data from a single pipeline operator.** As mentioned in the assumptions, this study is limited to data provided by a single pipeline operator. This inherently creates a level of bias in the study, but was the only data available for the purposes of this study.

3.2 Metrics for Success

Before continuing with this thesis, it is important to understand the metrics by which to measure the results. As explained in Section 1.5, the final pre-assessment tool (i.e., the ML model) resulting from this work shall be able to predict the SCF of a pipeline dent based on the ILI data. Therefore, a *good* ML model should be able to consistently predict a SCF within 0.00 - 0.50 root mean square error (RMSE). A coefficient of determination R^2 greater than 0.90 would be ideal. Additionally, the SCF error distribution should follow a Gaussian (Normal) Distribution to indicate that there is

minimal or no bias in the model. Lastly, a lower error for smaller target SCFs is desirable. This is due to the disproportional population of small SCFs (i.e., SCF of 0.78 to 4.00, which makes up 70.99% of the entire test data) and therefore more likely to be observed in real world applications than larger SCFs, e.g., SCF of 10.69.

As demonstrated in Fig. 3.2, this thesis proposes a pre-assessment tool that can replace current steps taken to assess a dent. It is important to note, however, that the current procedure, denoted as (a) in Fig. 3.2, prepares the data through several pre-processing steps to run numerical methods and **generate**, not predict, a SCF value. Since the SCF is generated using numerical methods through software such as Abaqus, it is considered final and trustworthy for the fitness for service (FFS) assessment. However, the proposed pre-assessment tool using a trained ML model could only predict the SCF of the dent. But as shown in part (b) of Fig. 3.2, it reduces the need for various steps and most importantly, the need for FEA modeling. This is the most expensive step in the procedure, as described in Section 1.4. Therefore, this thesis proposes that any SCF predictions generated by the ML model should be validated as part of the post-assessment procedure, and the ML model should be updated accordingly – essentially, by adding the validation results as a new training data point and SCF value.

This implementation in real-world applications could look like using the SCF results from the pre-assessment tool to complete the FFS and calculate the remaining life of the pipeline, thus placing a level of prioritization on the multiple dents based on the scale of the SCF; i.e., the higher the SCF, the more higher level of priority. Then, once the pipeline operator has the opportunity to manually process the ILI runs and generate SCFs, those values can be compared with the predicted values to observe how well the pre-assessment tool can generalize, and add those results to the database of pre-processed data, which is the same database used in Fig. 3.1.

The following section will cover the steps taken and decisions made in the development of the final model, which involved a regression CNN. All ML model development and testing was written in **Python** and performed on **Google Colab**, which executes code on Google's cloud servers and incorporates integration with **TensorFlow 2.0** for CNN models.

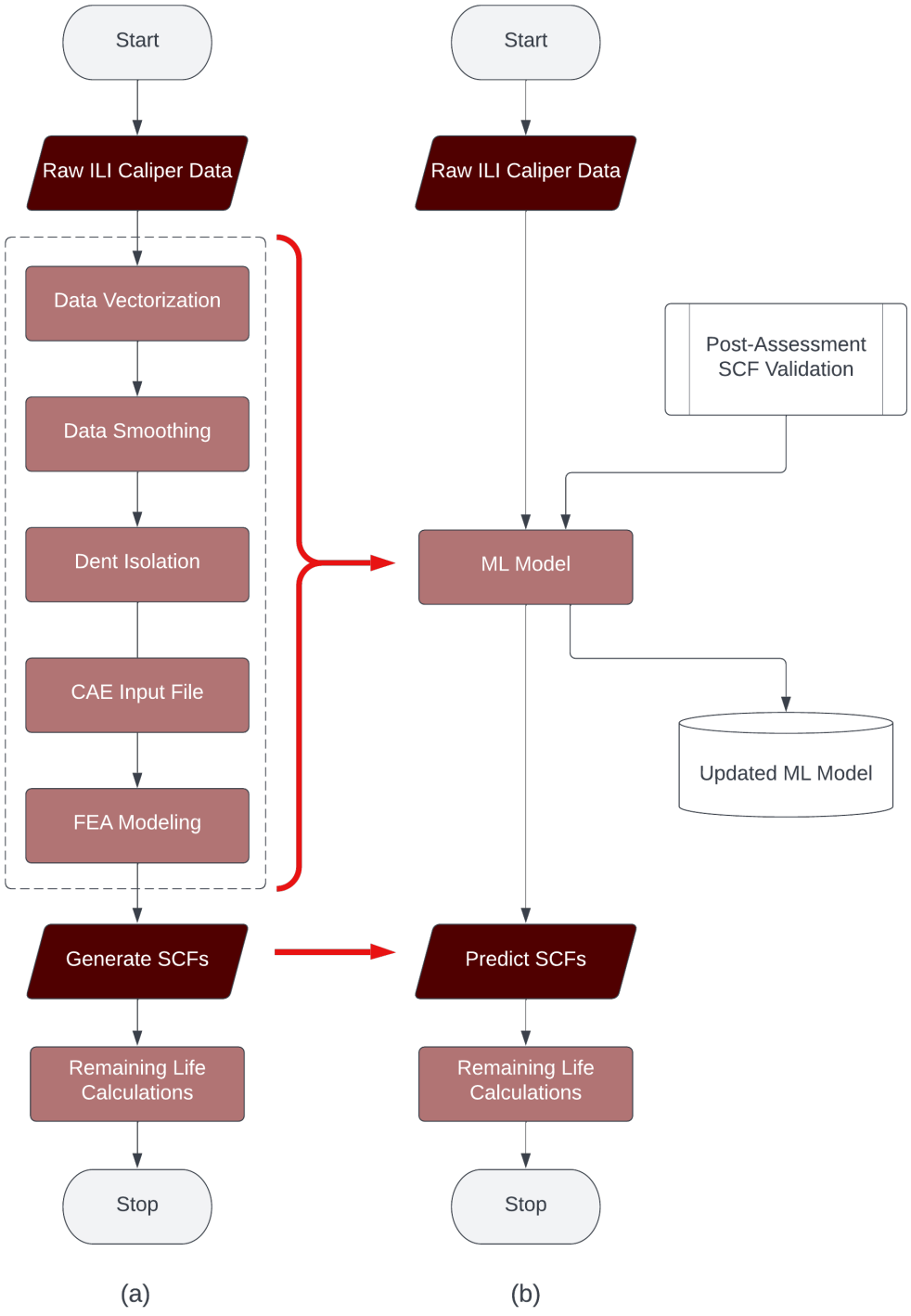


Figure 3.2: Comparison between (a) current procedure for generating SCFs vs (b) proposed procedure using ML to predict SCFs.

3.3 CNN Model Evaluation

The model evaluation involved testing various CNN models by adjusting the hyperparameters and observing the SCF prediction quality. This procedure involved using the **hold-out validation** method of evaluation, which consisted of setting apart a fraction of the data as the validation set. This way, after training the model on the training set, it can be evaluated strictly with the evaluation set. The purpose of this is to introduce new examples to the CNN model that it has not encountered before and not evaluate it with what has already been trained. The separation of this data is described further in Section 4.3, consisting of 70% of data for training and then 20% of data for validation. The remaining 10% of data was saved for the final model selection as a final test for generalization.

There were many ways to assess the development of the CNN model, but the items listed in Table 3.1 provides the final list of hyperparameters that were closely observed throughout this study. Most of the experimentation involved trial and error and random searching of the optimal configuration space, but typically observations were made by adjusting one hyperparameter at a time and gauging the impact on the model's prediction performance. The ability to evaluate the model's performance was made possible by a series of evaluation criteria as mentioned in Table 3.2

Table 3.1: Hyperparameter configurations for CNN model.

Hyperparameter Name	Value
Number of layers	2 - 4
Conv2D Filters	11 - 256
Conv2D Windows	(3×3), (5×5)
Conv2D Activations	ReLu, TanH
Dense Shapes	10 - 512
Dense Activations	ReLu
Compile Optimizer	Adam, RMSprop
Compile Loss	MAE, MSE, MSLE, MAPE
Compile Metrics	MAE, MSE

Table 3.2: Evaluation criteria for CNN model. Lower values are desirable for all except R^2.

Evaluation Criteria	Desired Range
Training Loss	< 0.20
Training Metric	< 0.30
Validation Loss	< 0.25
Validation Metric	< 0.35
RMSE	< 0.50
Max Error	< 3.00
Max Error Percentage	< 2.00
Training Time (s)	< 600
R^2	> 0.85

A major point of consideration when adjusting the hyperparameters was dealing with overfitting. According to [3], the "fundamental issue in machine learning is the tension between optimization and generalization". For ML model development, optimization refers to the process of minimizing the loss function, such as eq. 2.2, in order to get the best performance possible on the training data. On the flip side, generalization refers to the quality of performance on never before seen data, which in this case involves the validation data. Overfitting occurs when a ML model learns all relevant patterns in the training data, but after a number of epochs, begins to learn patterns that are specific to the training data but not representative of new data. An example of overfitting can be observed in Fig. 3.3 where the blue and orange curves are the metric and loss functions for training and validation, respectively. Towards the last epochs of that instance, the training values are significantly lower than the validation values by over a factor of 3, therefore overfitting to the training data. A minimal difference between the training and validation values is desirable with a slightly lower value for the training to demonstrate minimal overfitting.

3.4 Selection of Final Model

While there were many useful criteria to consider as described in Table 3.2, the selection of the final model was primarily driven by the minimization of the model's RMSE. When using MSE as the loss function for the hyperparameters, the RMSE was simply the root of the calculated validation loss over the final epoch. Consequently, several of the other evaluation criteria were also

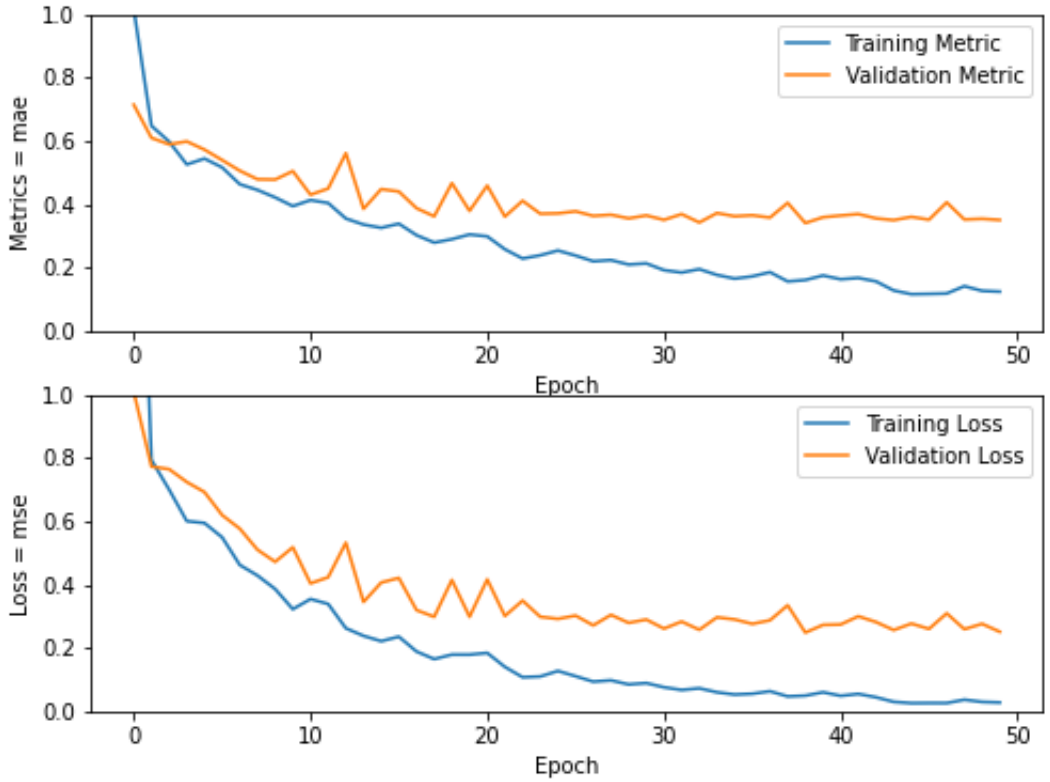


Figure 3.3: Example of a CNN model with significant overfitting during the training process.

minimized, but not the absolute lowest value of all the model iterations. A collection of best performing CNN models out of the 180 CNN model variations are described in the Appendices of this thesis. The graphical analysis of the training and validation procedure, prediction of SCF values, and distribution of residuals are displayed in Appendix A. The tabular display of the corresponding CNN model hyperparameters and their evaluation criteria are organized in Appendix B.

The following CNN model architecture displayed in Fig. 3.4 was selected as the optimal model. The same information is displayed in Table 3.3. The model used the following hyperparameters: 3 2D convolutional layers (Conv2D) with 64, 64, and 128 filters, and window size of 3×3 for all of them; 3 pooling layers (MaxPooling2D) of pool size 2×2; 3 fully connected layers (Dense) of 100, 10, and 1 nodes; activation funtion ReLu for all the Conv2D and Dense layers, except the last Dense layer; Adam optimizer with a learning rate of 0.001; loss function MSE; and metrics function MAE. The following section will review the performance of this model and observe how well it generalizes with the test data.

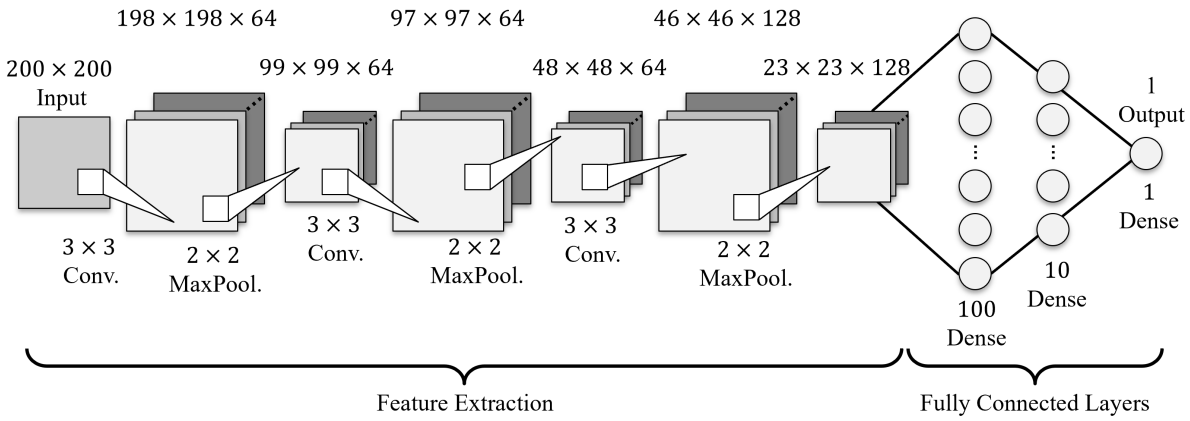


Figure 3.4: Architecture of CNN model used in this study.

Table 3.3: Architecture of CNN model used in this study (tabular format).

Layer (type)	Output Shape	Number of Parameters	Activation Function
Conv2D	(None, 198, 198, 64)	640	ReLu
MaxPooling2D	(None, 99, 99, 64)	0	-
Conv2D	(None, 97, 97, 64)	36,928	ReLu
MaxPooling2D	(None, 48, 48, 64)	0	-
Conv2D	(None, 46, 46, 128)	73,856	ReLu
MaxPooling2D	(None, 23, 23, 128)	0	-
Flatten	(None, 33856)	0	-
Dense	(None, 100)	6,771,300	ReLu
Dense	(None, 10)	1,010	ReLu
Dense	(None, 1)	11	-

4. DATA COLLECTION AND PRE-PROCESSING

Proper care is necessary when training and validating a machine learning (ML) model. After all, a model is only as good as the data used to train it – therefore, if bad data is used, then a ML model will make bad predictions. This section covers in detail the extensive work taken to prepare the data into a usable format before and during the development of the ML model as covered in Section 2.1. This work involves the collection and formatting of the data, as well as the pre-processing steps taken to vectorize, normalize, smooth, and convert the data into input datasets.

4.1 Data Collection

The data used in this study involved several in-line inspection (ILI) runs collected and analyzed in 2018 and 2019 for 8 distinct pipeline segments, given by an anonymous company[1]. The data contained raw ILI data collected from 3 distinct ILI vendors. To maintain the anonymity of the pipeline operator company, the ILI vendors will remain undisclosed as well. The number of dents identified is summarized in Table 4.1 below. Note that only the dent files containing a single defect were considered, therefore bringing down the total number of usable dents to 4,667.

Table 4.1: Dent data for all vendors.

Vendor #	Number of ILI Runs	Number of Dents Identified
Vendor 1	10	4,642
Vendor 2	4	1,723
Vendor 3	2	223
Total	**16**	**6,588**
Total Usable	**16**	**4,667**

As explained in Section 1.2, this study only considered caliper ILI tools which have mechanical arm components that are continuously measuring the internal radius of the pipe wall. Note that for

[1]The pipeline operator company gifted the data for the use of this study but explicitly requested to remain anonymous. All identification and proprietary information that could trace the study back to the company has been removed. The author plans to initiate conversation with the company to allow the publishing of redacted data to encourage repeatability and future studies.

the purposes of this study, the terms "calipers" and "mechanical arms" are used interchangeably. Every caliper displaces radially, therefore recording a unit of length in the radial direction as the ILI travels in the pipe's z-axis, or axial direction. The resultant data is in a cylindrical coordinate system as demonstrated in Fig. 4.1. The positive circumferential direction is in the clockwise direction, and the positive axial direction is moving (a) into the page from a front view perspective, and (b) towards the right from a side view perspective, as shown in the figure.

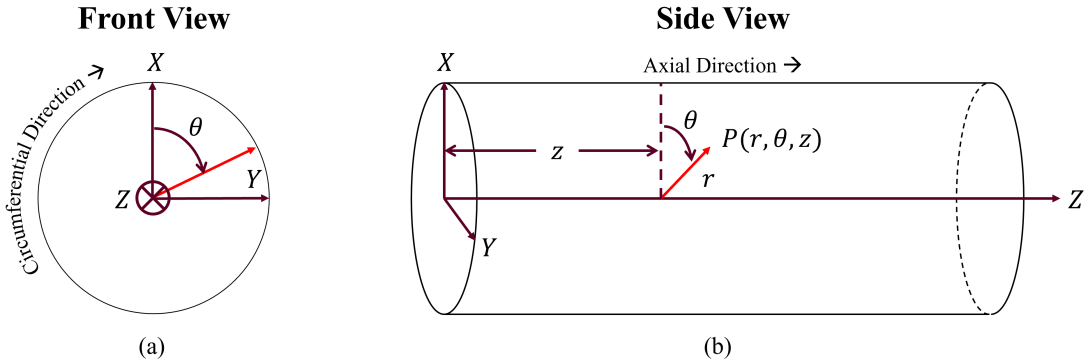


Figure 4.1: Cylindrical coordinate system for the ILI data, showing the (a) front and (b) side view.

The caliper arms measure the change in internal radius (IR) of the pipe wall. Figure 4.2 demonstrates the radial displacement in the radial direction r of a single caliper as it moves in the positive axial direction z. The caliper is in nominal position at time t_0, compressed at time t_1, and decompressed at time t_2. This example is only for visualization purposes and does not represent the actual shape of a dent in a pipe wall.

4.1.1 Data Resolution

Caliper ILI tools are physically constrained by the number of calipers they are configured with, resulting in a non-continuous angle θ. Instead, the angles increment in equal intervals encompassing the entire internal pipe wall circumference. Some of the tools that gathered data for this study are made with 80 calipers, therefore the angular displacement is fixed at 4.5° between each arm ($360°/80 = 4.5°$). The path of travel for the calipers can be visualized in Fig. 4.3 shown below.

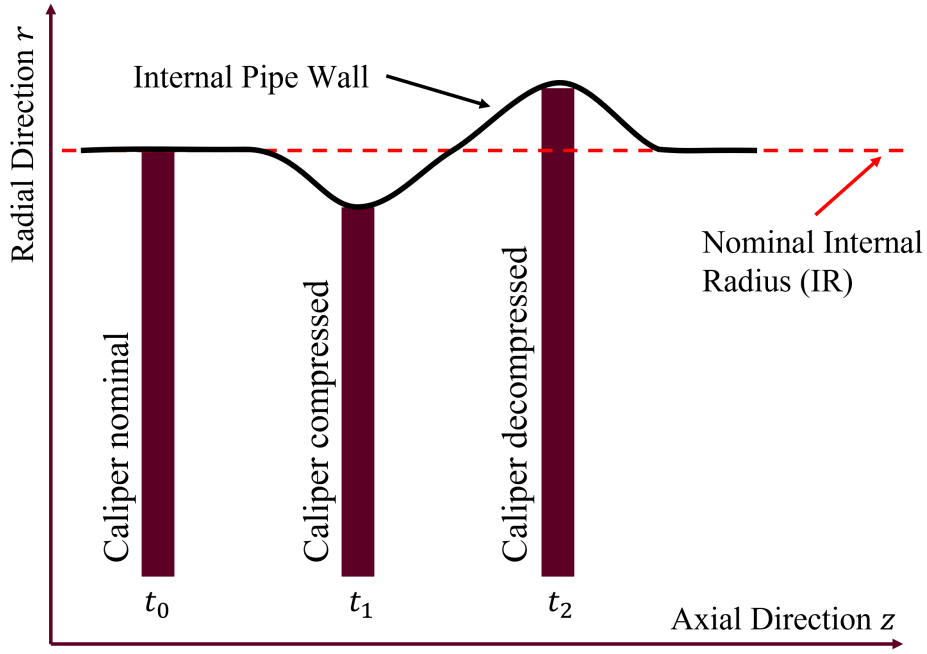


Figure 4.2: Displacement of ILI caliper arm in the radial direction r.

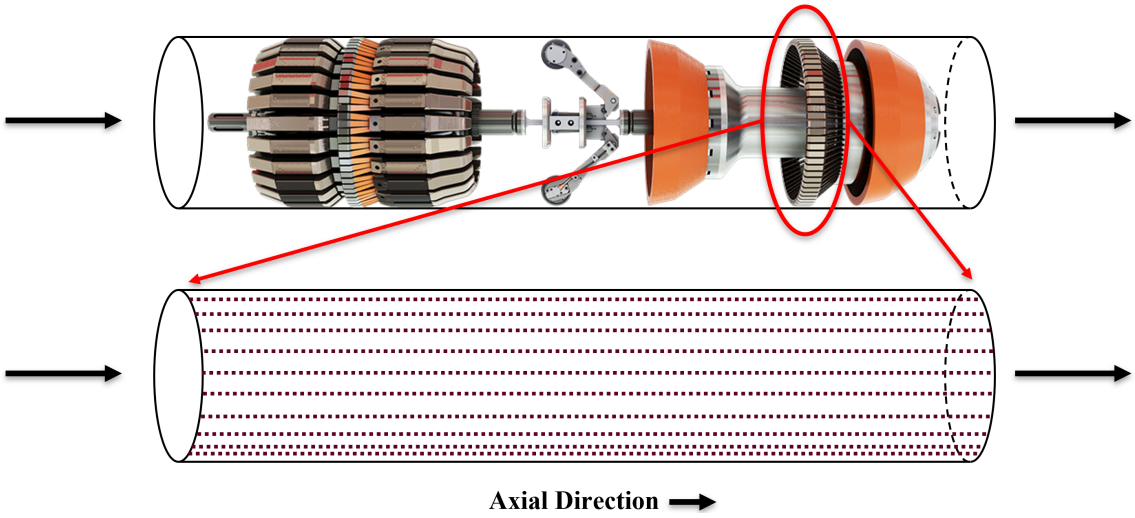


Figure 4.3: Collection of data from an ILI tool as it travels through the pipeline. Note: the displayed ILI tool [2] is only for example purposes, it is not the true tool used for the data gathered.

While the tool may be limited in the amount of data collected in the circumferential direction, data gathering in the axial direction is subject to the tool's sampling frequency. For example, if an ILI tool is gathering data points at a frequency of 320 Hz and flowing down a pipeline at a flow rate velocity of 6.67 feet per second (2.03 meters per second), then it will collect 1 data point every

0.25 inches (6.30 millimeters). The tool's flow rate can vary with pressure gradients, temperature, debris, and other external factors, causing the velocity to vary with time. This indicates that the gathering of data is time dependant and the density of the data in the axial direction varies as the tool travels down the pipe. Therefore, no two ILI runs are the same. The sampling rate in the axial direction was observed to vary anywhere from 1 data point per 0.10 inches to 0.25 inches. This was taken into consideration when pre-processing the sample data, as explained in Section 4.2, which resulted in interpolated data fixed at exactly 0.50 inch sample intervals for both the axial and circumferential direction.

4.1.2 Data Format

After a vendor performed an ILI run through a pipeline, the data was exported into comma-separated values (CSV) file format, which was converted into tabular format on Microsoft Excel. An example of raw ILI data from a specific vendor is displayed in Fig. 4.4. The Flow Direction indicates the positive temporal direction in which the ILI tool traveled, therefore showing the tool moving from left to right. The calipers (circumferential direction) were distributed across the table rows and the axial direction was distributed across the table columns. Critical identification information was redacted in the blackened regions to maintain the anonymity of the pipeline.

The body of the tabular data contains raw radial displacement values. These values are the residual of the caliper arm measurements r_{raw} and the nominal IR of the pipe wall. This simple difference calculation can be seen in Eqn. 4.1 shown below.

$$r_{disp} = r_{raw} - IR \tag{4.1}$$

It is important to note that not all vendors followed this same data format, as some vendors only exported the caliper arm measurements without taking the difference from the nominal IR. This was carefully noted and all data was converted to represent IR raw displacement measurements to maintain consistency.

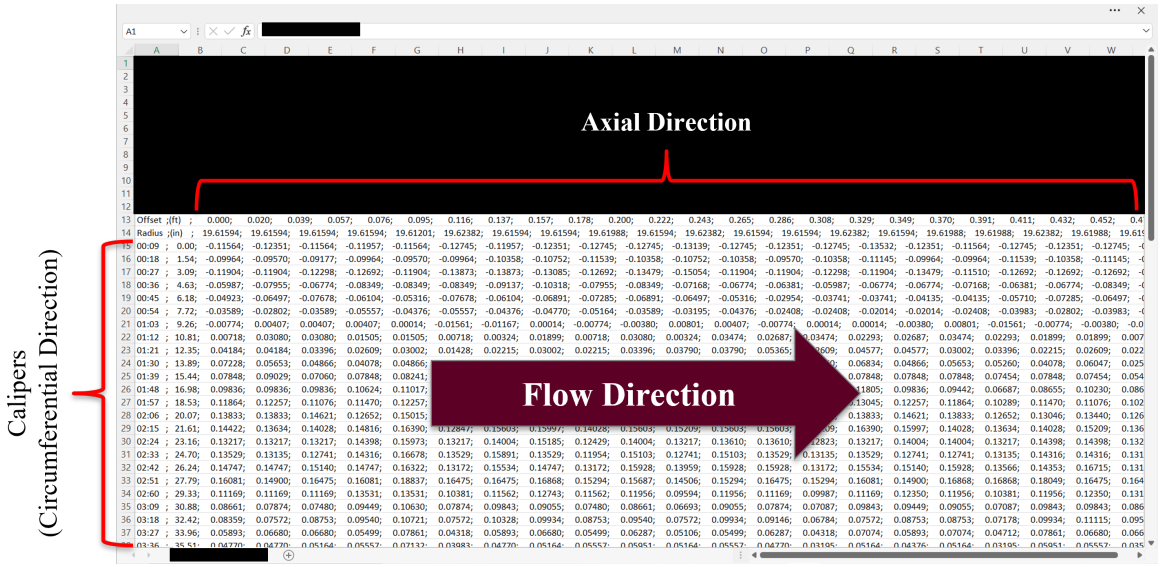


Figure 4.4: Example of the raw ILI data from Vendor 1 displayed on Excel Workbook.

4.1.3 Dent Registry

Coupled with this raw ILI data were various databases with information per ILI run containing mechanical information about the pipeline and the results from prior engineering assessments. Every ILI run had its own database, totalling in 16 databases. These databases were organized by the dents identified within the pipeline run. The number of dents stored per database varied; for example, a single database from Vendor 1 contained 276 dents, while a single database from Vendor 2 contained 484 dents. For the purposes of this study, the entire collection of dent databases will be referred to as the "dent registry", an example displayed in Table 4.2 below. The mechanical information observed in this study included the Outside Diameter (OD) and Wall Thickness (WT); likewise, the results from the prior engineering assessments considered in this study included the previously calculated SCFs and any written comments – engineers made specific comments per dent to provide guidance in the analysis and could identify unreliable or bad results. As these were entries entered manually by the engineer(s) performing the fitness-for-service (FFS) assessments, they held value to consider for future review. Such comments could indicate insight into the shape of the dent, interaction with other dents or features, or the possibility of underperforming ILI tool

calipers, to name a few. This information was taken into consideration as part of the pre-processing steps and are explained further in Section 4.2. There are additional columns of information in the dent registry that were not considered for this study, but they have potential use in future studies and are expanded at the end of this thesis in Section 6.1.

Table 4.2: Dent Registry data format example.

Dent Ref #	Outside Diameter, inches	Wall Thickness, inches	SCF	Comments
0001	40.00	0.500	4.44	N/A
0002	40.00	0.344	5.42	Adjacent Dent
0003	40.00	0.344	3.83	N/A
...	...	...	...	...
Nth Dent	Nth OD	Nth WT	Nth SCF	Nth Comments

Of the data collected from the dent registry, only the SCFs served as the targets for training the ML model. As covered in Section 1.5, the goal is to develop a ML model that can learn the relationship between training inputs, being the raw ILI data, and the training targets, being the SCF values. This is a method of *supervised learning* and will be expanded further in Section 2.1. For the purposes of this study, it was necessary to understand the distribution of SCFs for the training data used. Other studies have explored how imbalanced data impacts ML regression [56, 57] and demonstrated that it is a critical component to consider in order to maximize the generalization of the ML model. After all, a ML model is only as good as the data it is trained on; therefore, it will make better predictions with the data that matches the statistical properties of what it was trained on. The distribution of the SCFs of the raw ILI data can be observed in Fig. 4.5. As observed in the figure, majority of the data falls within SCF values of 1.04 to 4.00, making up 71.02% of all the data. By definition, this dataset would be slightly imbalanced. However, it is inevitable and expected for this field of application because it is rare to find oil and gas pipeline dents with an SCF as high as 10. In such cases, dents with SCFs that high are likely in an analysis condition that make the traditional methodology incapable of representing them [17].

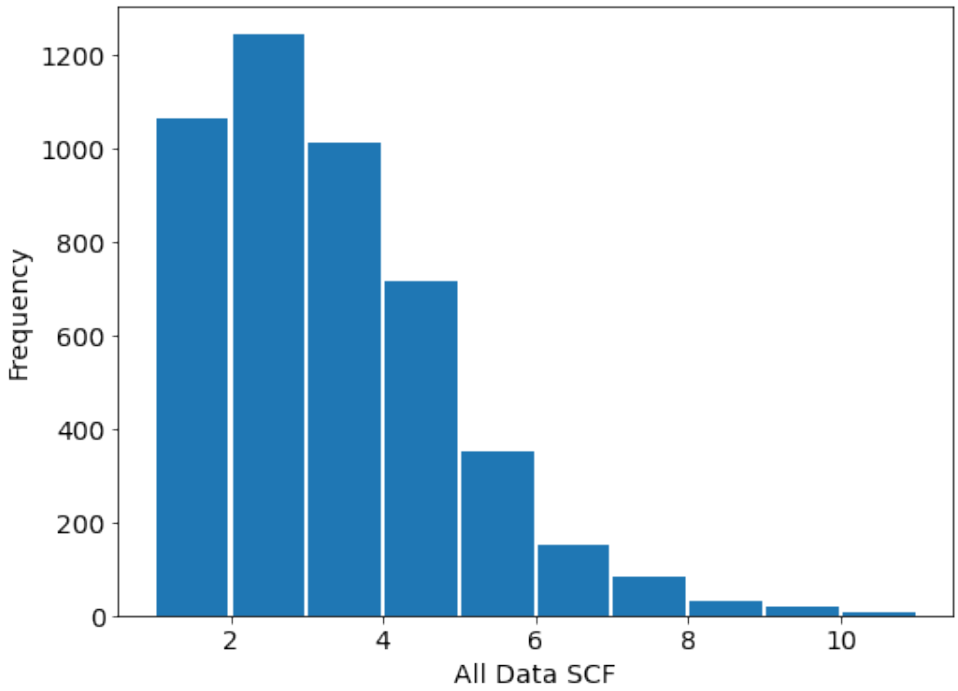


Figure 4.5: Distribution of SCF values.

4.2 Data Pre-Processing

This section covers in detail the steps taken to prepare the raw ILI data into a useable format to train, validate, and test a ML model. As covered in Section 1.5, this study aims to develop a tool that can predict the SCF values of a pipe dent based on raw ILI caliper data. The purpose of data pre-processing is to make the "raw data at hand more amenable to neural networks" [3], which includes vectorization, normalization, and other appropriate data manipulation techniques. There are additional steps taken in this study that may not be part of traditional CNN data pre-processing, such as data smoothing and data conversion to image, due to the unique problem setting at hand.

In the Fall of 2021, the author worked with ADV Integrity, Inc. (ADV) to develop a semi-automated numerical tool that outputs dent SCFs through finite element analysis (FEA) of ILI data. The tool would help in the completion of fatigue life assessments to support the dent FFS assessments. The resultant tool was the Pipeline Defect Modeling and Analysis Calculator

(PDMAC)[2], which uses a Python script and FEA to complete the pre-processing, processing, and post-processing steps using ILI data for multiple dent features. This tool is an example of a streamlined procedure as described in Section 1.3. The development of the PDMAC inspired a lot of the work in this study, such as the data smoothing steps covered in Section 4.2.3. All of the pre-processing was executed using **Python 3.9** and the **Spyder IDE**. The **SciPy** open-source Python library was used with specific packages **scipy.interpolate** and **scipy.signal** for the Data Smoothing in Section 4.2.3.

4.2.1 Vectorization

All of the inputs in a neural network must be tensors of floating-point data. A tensor is a "generalization of matrices to an arbitrary number of dimensions" [3] which, at its core, is just a container for data. A 2-dimensional (2D) tensor is a matrix as can be displayed in Eqn. 4.2 where $\mathbf{R}_{disp}$ is a matrix of raw radial displacement values. In the context of tensors, dimension is often called an axis; so for a 2D tensor, those 2 axes are simply the *rows* and *columns* of the tensor. As a result, a matrix takes on the shape of $N_{rows} \times N_{columns}$. For the purposes of this study, this is equivalent to $N_{circ} \times N_{axial}$ (i.e., size in circumferential direction by size in axial direction).

$$\mathbf{R}_{disp} = \begin{bmatrix} r_{11} & r_{12} & \dots \\ \vdots & \ddots & \\ r_{N1} & & r_{NN} \end{bmatrix} \tag{4.2}$$

As reviewed in Section 4.1.2, the raw ILI caliper data is in tabular format. This is equivalent to a matrix if the axial header labels and circumferential labels are not considered, i.e., by "dropping" anything besides the radial displacement values. As a result, the raw ILI caliper data becomes 2D tensors of radial displacement measurements. Following the convention of rows and columns, the rows and columns of $\mathbf{R}_{disp}$ are the circumferential direction and axial direction of the radial displacement values, respectively.

[2]PDMAC is a registered tool for ADV Integrity, Inc. For more information regarding the development or application of the PDMAC, please contact Emmanuel.Valencia@advintegrity.com

4.2.2 Value Normalization

Common best practices for ML neural networks involve using data that does not contain relatively large values or that is heterogeneous, i.e., data where one segment has values in a small range, like 0-1, and another segment has values in a much higher range, like 100-200 [3]. Therefore, it is recommended that most values be relatively small and that they be homogeneous (i.e., values should be in the same range). If this practice is not followed, then the data can "trigger large gradient updates that will prevent the network from converging" [3]. In application, this means adjusting the raw radial displacement values, or $\mathbf{R}_{disp}$, to a similar range of values kept consistent through the entire set of data. To do this, the nominal IR of a pipe wall is used to scale all of the data to a similar range of values, as shown in Eqn. 4.3.

$$\mathbf{R}_{norm} = \frac{\mathbf{R}_{disp}}{IR} = \frac{\mathbf{R}_{raw} - IR}{IR} \tag{4.3}$$

This is similar to the pipe wall metal loss ratio, which is commonly used in the oil and gas industry but scaled by the nominal outside diameter (OD) and represented as the percent of diameter (e.g., 6% of diameter). This study assumed that the differences in scaling would be negligible as long as the application was consistent, since common oil and gas industry practices label pipelines with a standard wall thickness.

4.2.3 Data Smoothing

The data smoothing pre-processing covered in this section was influenced by the development of the PDMAC for ADV. The ILI tools are prone to random and inherent error affecting the quality of the ILI data. Random events can range from vibration of the ILI tool as it speeds through the pipe, or bumping on an obtrusion (e.g., dent defect, pipe wrinkle, debris, etc.) that results in brief oscillations of the system, among many more examples. But other errors are caused by the design or failure of the ILI tool itself. For example, if an individual caliper on an ILI tool is malfunctioning during operation, then that channel of data will produce outliers when compared to the surrounding calipers and skew the rest of the data. This variability is worth considering for further study, but

as mention in Section 3.1, this study assumes that the raw ILI caliper data used to be accurate and reliable depictions of internal pipe wall contours for oil and gas pipelines.

Nevertheless, the original consideration for data smoothing was required in order to perform an FEA of a dent feature. The reader should note that while the final result of this thesis **does not** involve FEA modeling (in fact, the desired result is to delay or replace the need for FEA modeling as explained in Section 3.2 and shown in Fig. 3.2), several dent features in this study were processed by the PDMAC which involved FEA. Performing a FEA requires a fine mesh model, which is a three-dimensional grid dividing complex geometries into elements in order to discretize a domain [58]. Therefore, a finer mesh allows higher fidelity simulations. In this case, any type of noise in the mesh can result in incorrect simulations. Noise in the ILI data creates jagged shapes resulting in zig-zag patterns across the circumferential direction, as demonstrated by the image (a) in Fig. 4.6; the result of proper smoothing techniques is shown in image (b).

Industry standards such as the ASME B31.8 Appendix R Estimating Strain in Dents establishes a procedure for minimizing this noise, as it states: "The user is cautioned that analysis of surface curvatures for the purpose of determining local strains of deformation can be significantly affected by random error inherent with all geometric measurement techniques. *Suitable data smoothing techniques* should be employed to minimize the effect of such errors" [20]. The following steps explain the smoothing techniques that this study used for this section.

Smoothing Windows

The raw ILI caliper data was smoothed using a Savitzky-Golay filter (Python function is **scipy.signal.savgol_filter**). This digital filter was selected due to its ability of increasing the precision of the data without distorting the signal tendency, effectively reducing the signal-to-noise ratio [59]. As part of the application, one of the function parameters requested the length of the filter window to determine the number of data points to consider. This can be demonstrated in Fig. 4.7. Due to the disproportionate resolution between the axial and circumferential direction, a greater window size was considered for the axial direction. The filtered data points were then used to determine B-spline cubic representations in both axial and circumferential directions.

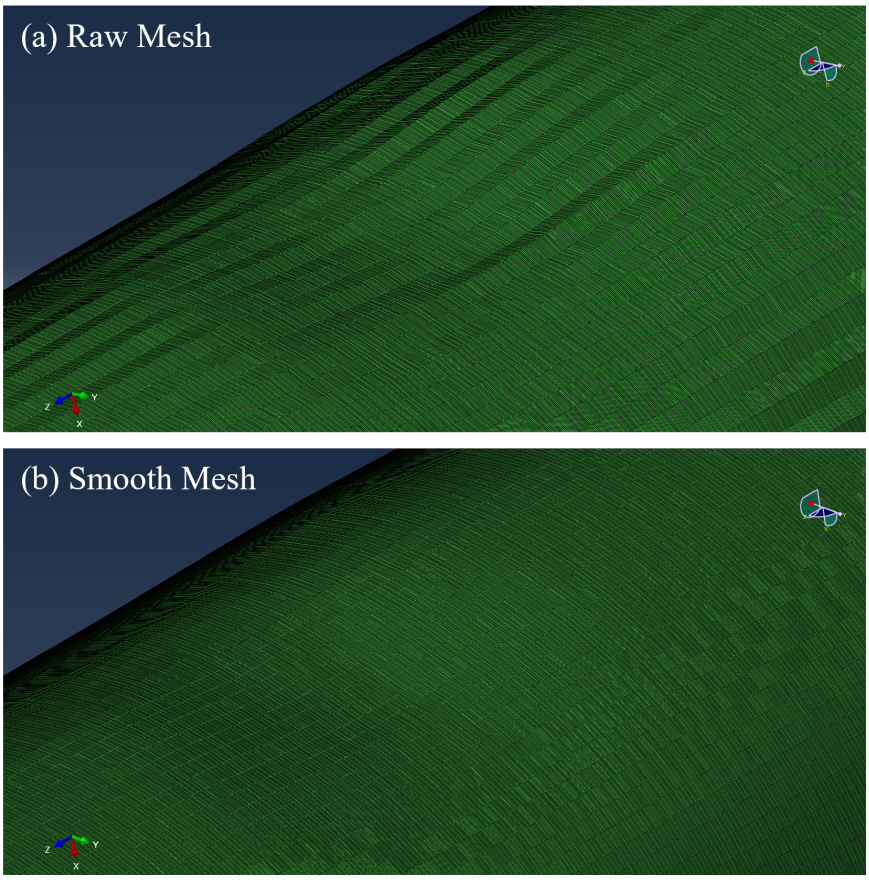
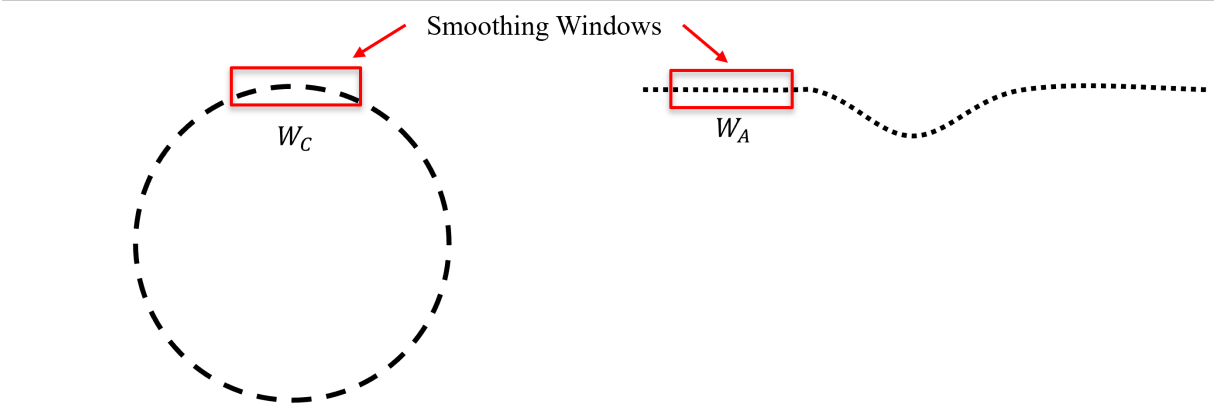


Figure 4.6: Mesh models of (a) raw ILI data and (b) smooth ILI data. Jagged lines on top mesh are a result of noise in the raw ILI data.

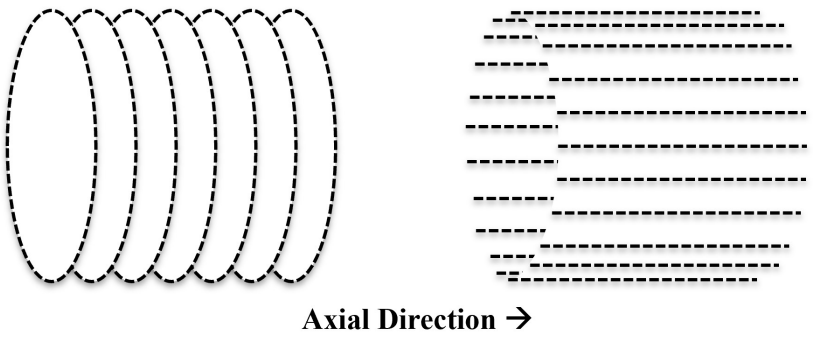


(a) Circumferential Smoothing Window (b) Axial Smoothing Window

Figure 4.7: Selection of data points with W_C and W_A being the circumferential and axial smoothing window parameters, respectively.

B-Spline Representation

A method of smoothing 3D space data involves surface reconstruction with B-splines. A B-spline is a piecewise polynomial function that can be used to model the solution space for object representation. This technique has been studied in [60, 61] and demonstrates that scattered 3D data can be reconstructed into smooth representations. For this study, the 3D data is the radial displacement data in the cylindrical coordinate system as shown previously in Fig. 4.1. The objective was to find the cubic B-spline representation of one-dimensional (1D) curves by iterating through all of the circumferential profiles at every axial interval. This procedure was repeated by switching the profile direction, this time by iterating through all of the axial profiles at every circumferential interval. In effect, this generated two sets of 1D curves (i.e., a surface) giving a representation of the pipe wall surface as demonstrated in Fig. 4.8: (a) one from circumferential profiles, and (b) the other from axial profiles.



(a) Circumferential Profiles (b) Axial Profiles

Figure 4.8: Generated 1D curves from (a) circumferential profiles or (b) axial profiles.

These two surfaces are describing the same true surface but were generated by fixing one variable at a time (i.e., the circumferential or axial position), thus resulting in different variation of error. The variation for each set of curves from the raw data was calculated and used to determine the weighted average positions based on the magnitude of variation, i.e., if the surface from one profile representation resulted in greater variation then it would be weighted heavier. As a result,

new surfaces were generated by combining multiple cubic spline functions in both the circumferential and axial directions.

Sub-Sampling

The final step in the data smoothing process was to evaluate the splines on the desired axial and circumferential intervals and export the profiles as a single surface. Since the axial direction had a greater resolution due to the higher frequency of sampling as explained in Section 4.1.1, the aspect ratio was skewed in the axial direction. To visualize this, if the raw ILI caliper data were represented as an image where every pixel was a data point, then it would resemble a picture that was significantly stretched in one direction. Sub-sampling can help eliminate this distortion by evaluating the splines at the interpolated intervals that create a 1:1 ratio for circumferential to axial direction. Effectively, this converts long rectangle elements into squares elements, as demonstrated in Fig. 4.9. For the purposes of this study, the desired interval lengths were 0.50 inches in both the axial and circumferential directions.

To summarize, the raw ILI caliper data is smoothed using a Savitzky-Golay filter; then used to create a weighted average, B-spline representation considering both the axial and circumferential directions, effectively creating a new surface; and the surface is evaluated at desired intervals of 0.50-inch by 0.50-inch for the length between each point in space for both the circumferential and axial directions. In other words, the surface data now has a square resolution. Finally, these new data points replaced the raw data points. An example of this procedure when applied to a 1D curve is shown in Fig. 4.10 and then to a surface in Fig. 4.11. As observed in the example figure, the new data points follow the curve that best represents the true data while applying a level of "smoothness", i.e., no jagged lines or instantaneous jumps in the data.

4.2.4 Data Conversion to Image

Since the most common use for a ML convolutional neural network (CNN) takes in visual imagery as input, this study had to treat the ILI data as pseudo-images for the CNN model development. This is not a new practice, as Kwak et al. [62] explored this methodology by converting

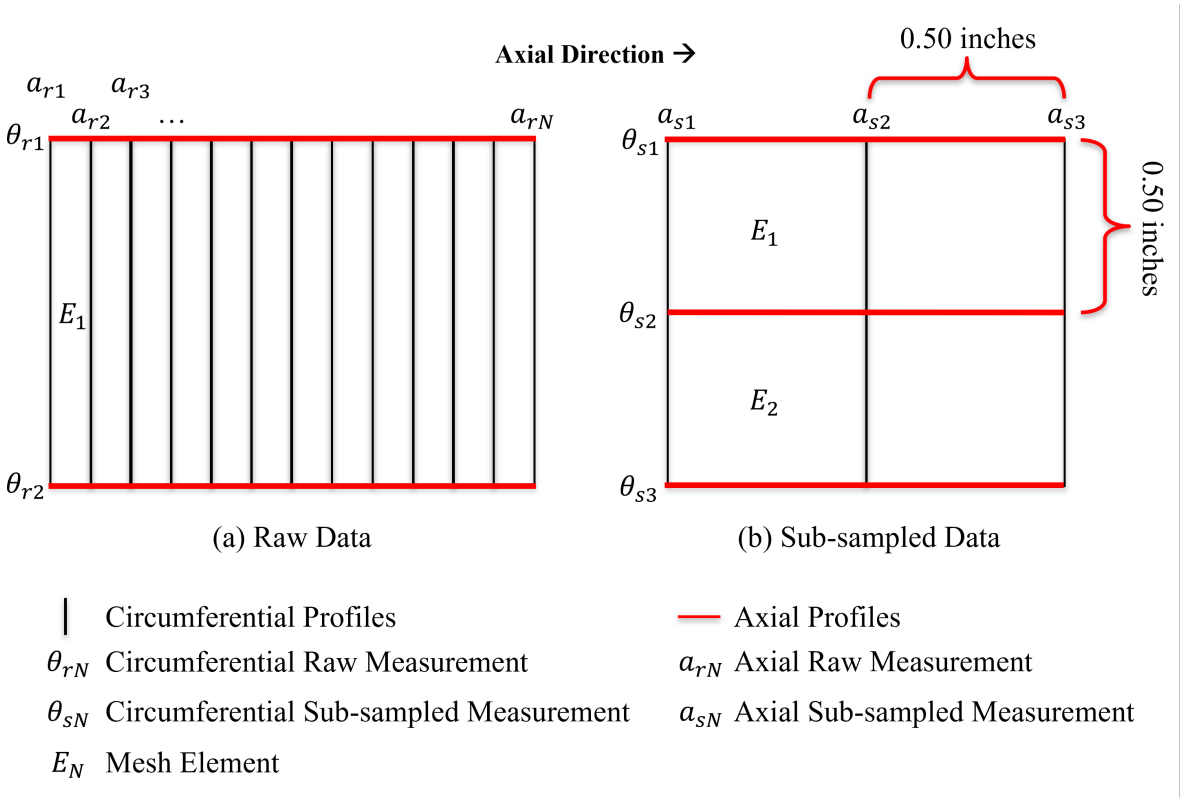


Figure 4.9: The change of sampling density in the axial direction as the raw data was sub-sampled. Variables a_r and a_s denote the axial measurement of raw data and sub-sampled data, respectively. Variables E denote the elements composing the mesh.

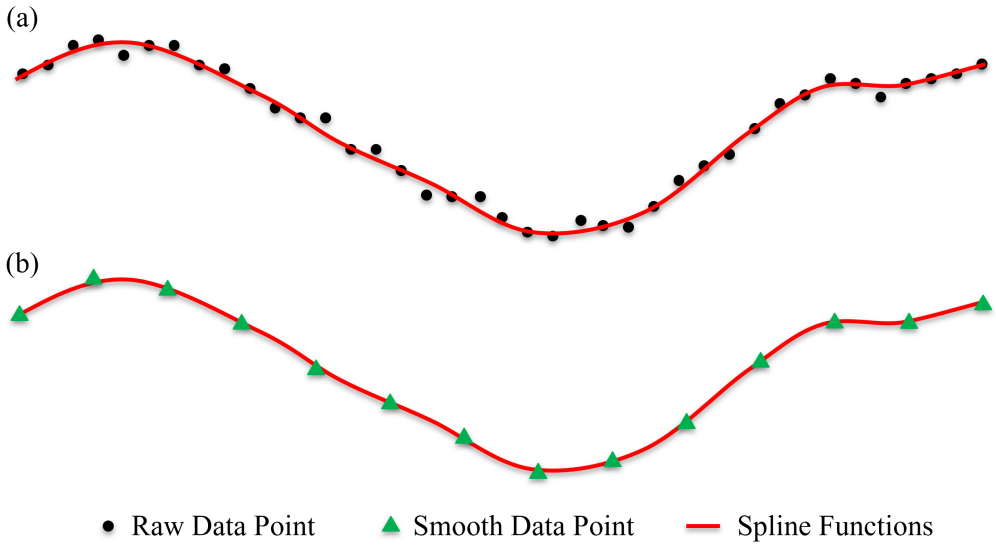


Figure 4.10: Example of resultant axial direction curve after applying B-spline functions and the output smooth data points.

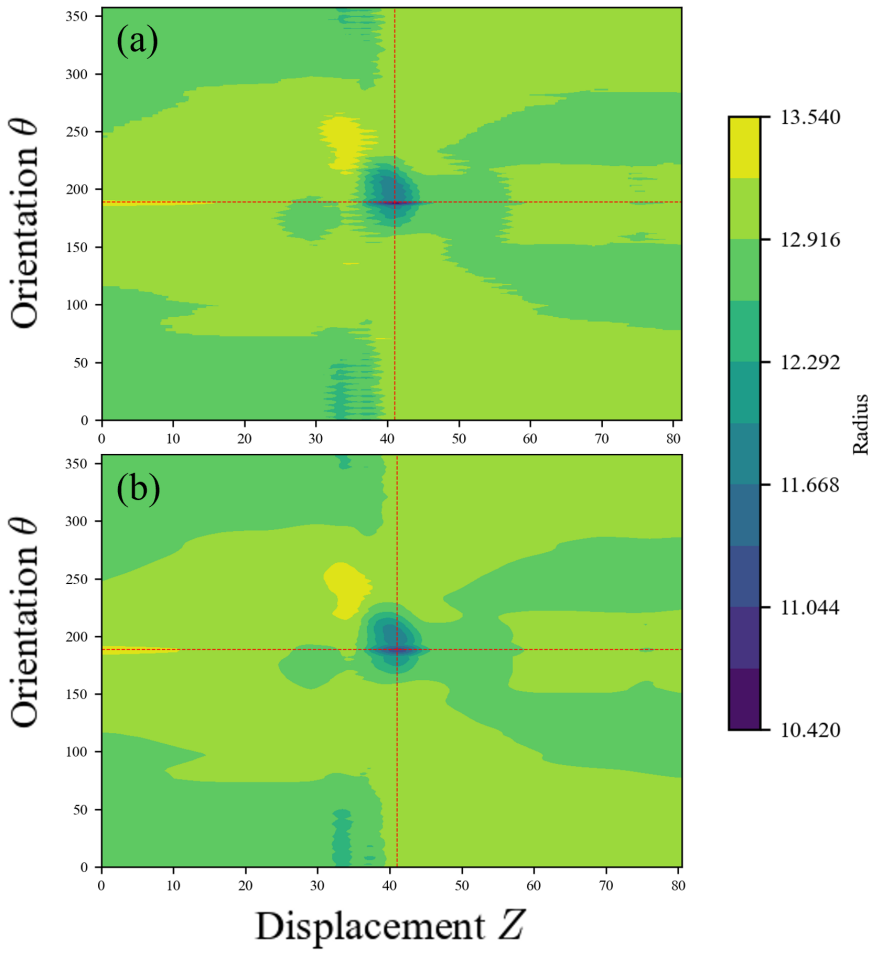


Figure 4.11: Contour map of a pipeline's IR showing (a) raw data and (b) smooth data. The deepest point (smallest IR value) of the dent is the center of the crosshair.

audio signals into pseudo-images to train a CNN. Similarly, Kantzos et al. [51] synthetically generated rough surfaces by combining sine waves and representing them as images to use with a CNN. As covered in the previous section, the exported data from the data smoothing procedure is now a smooth representation of the pipeline's internal wall with an adjusted resolution in both the axial and circumferential directions. Every data point in the exported matrix is equivalent to a pixel in an image, as visualized in Fig. 4.12. Moving forward with this thesis, the concept of matrix data displayed as an image will be simply referred to as images. To clarify further analysis, the term image size is used to describe the $pixel \times pixel$ size of an image, which for these purposes, is equivalent to the $rows \times columns$ shape of a matrix.

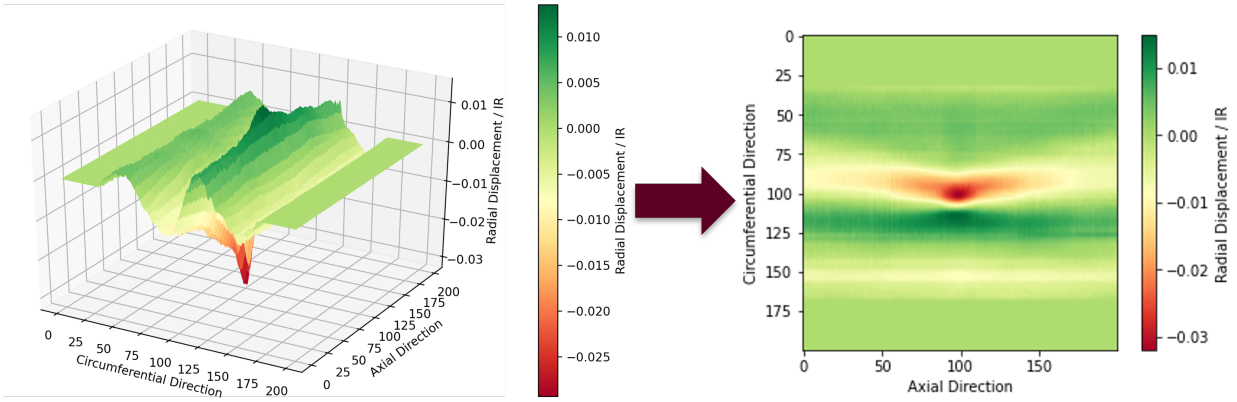


Figure 4.12: Representation of dent surface as an image.

There were two important things to consider before naively inserting these images into a CNN model: (1) choosing an optimal image size $N_{circ} \times N_{axial}$, and (2) reformatting the images so that the key information (i.e., the dent feature) was always in the same location. For (1), the resultant images still had relatively large and nonuniform shapes after the data smoothing process; an example raw data matrix of shape 80×816 become shape 252×403 after the data smoothing process, demonstrating a trade-off of resolution by decreasing the size of the columns (axial direction) and increasing the size of the rows (circumferential direction). Larger images are slower to train and will require more computational power but also provide more pixels (or data points) for the CNN model to learn important features. Also, a fixed shape, representing width and height of the image, is desired to standardize the procedure and have uniformity. After careful review of the typical smooth data matrix shape, an image size of 200×200 was determined. In response to point (2), the images were reformatted so that the deepest point of the dent was at the center of the image. The purpose of this centering was to place the CNN model's "focus of attention" in a consistent region to facilitate the feature engineering of the images. This alleviated the learning of those features as the CNN model was able to learn the relationship more efficiently between the images, seeing that they were relatively easier to locate [63].

Therefore, the following steps were taken to create the final output image data:

1. Find the index of the deepest point in the dent. Following the convention from the value normalization discussed in Section 4.2.2, the deepest point will be the largest **negative** value. This is now the center of the dent.

2. Center the dent in the circumferential direction by rearranging the matrix values. This process involves shifting the top vector array to the bottom in order to "shift up" the dent profile.

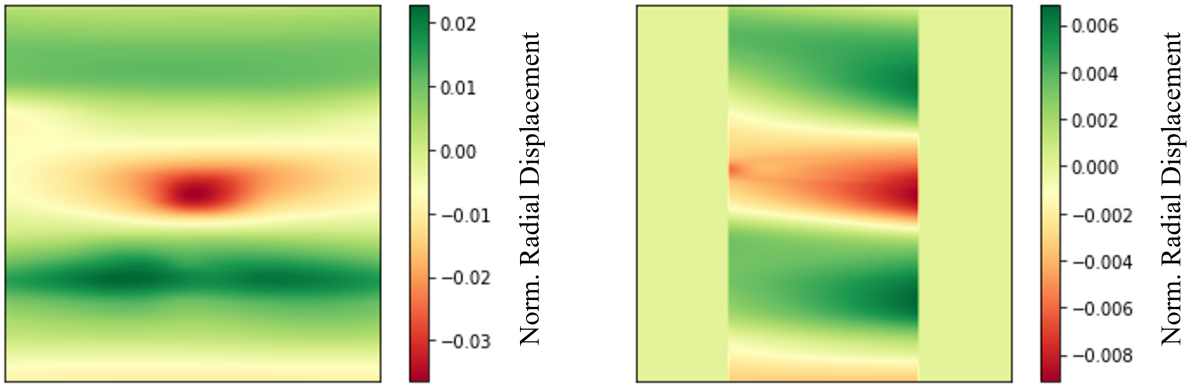
3. Starting from the center of the dent, create a perimeter expanding out N_{axial} and N_{circ} number of points in the axial and circumferential directions, respectively.

4. Once the square perimeter is established, export everything contained within the perimeter as the final matrix of the dent profile. This should result in a $N_{circ} \times N_{axial}$ image shape.

5. In the case that the desired perimeter exceeds the number of data points available, generate a border matching the nominal value (i.e., value of 0) until the desired dimension is reached. This is common for ILI data where the center of the dent is close to the start or end of the ILI run segment, leading to a "cut-off" effect.

In addition to Fig. 4.12, the set of images in Fig. 4.13 contain examples of this result. In such cases, the image sizes are 200×200 pixels, which is equivalent to a physical size of 100-inch width by 100-inch height (every pixel represents a spatial distance of 0.50 inches, therefore $200 * 0.50 = 100$). The image (a) in Fig. 4.13 has a circumference of about 125.66 inches, therefore this image size covers almost the entire circumference. This demonstrates that the choice of image size 200×200 is suitable.

By convention, digital image data has 3 dimensions: height, width, and color depth. For an image with an RGB color space, the color depth will be 3, but grayscale images only have a single color channel, and therefore a color depth of 1 [64]. Therefore, image tensors are always 3D. In order to make the ILI image data compatile, the 2D tensors (or matrices, as explained in



(a) Dent surface (b) Dent surface with border

Figure 4.13: Data displayed as an image using Red Yellow Green as the colormap. Image (a) is a normal dent surface, while image (b) is a dent surface with the borders added on the axial direction.

Section 4.2.1) had a dimension expanded to become 3D tensors. The resultant tensor has the shape of $N_{circ} \times N_{axial} \times 1$. Only the structure of the data has changed in this step.

A single 3D tensor represents one ILI image data. After all of the raw ILI caliper data was pre-processed, a batch of images could be stored as a 4D tensor (a packing of 3D tensors in an array) of shape $N_{samples} \times N_{circ} \times N_{axial} \times 1$, otherwise described as a tensor of shape (samples, height, width, color depth). A visual representation of this data structure is displayed in Fig. 4.14.

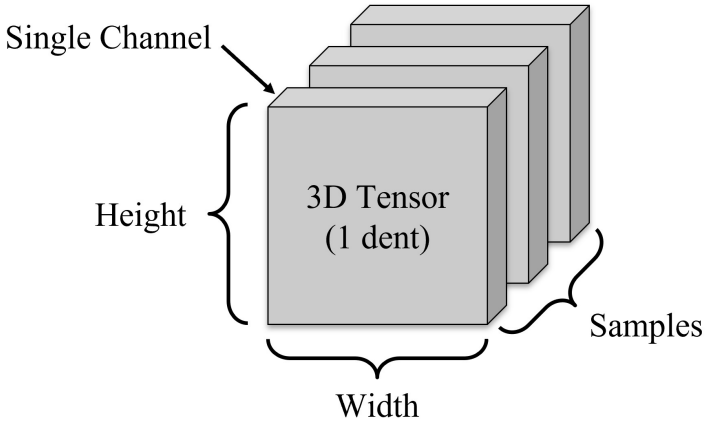


Figure 4.14: The packing of 3D tensors in an array becomes a 4D tensor.

4.3 Resultant Datasets

The series of steps explained in Section 4.2 were performed with slight variations to develop a series of image datasets for training and validating a CNN model. There were two parameters observed: (1) raw vs smooth data, and (2) all vs pristine data. For parameter (1), the need for data smoothing was primarily for running raw ILI data through the PDMAC system in order to generate SCFs as part of the standard FEA procedure. However, the capabilities of data smoothing remained in order to essentially perform cubic B-spline interpolation and achieve desired intervals of 0.50-inch by 0.50-inch for the length between each point in space for both the circumferential and axial directions. This served as a form of standardization of the data. Without data smoothing, then the raw ILI data was used as is, and with varying displacements in either direction. But the raw ILI data is a better representation of the actual observations made in the field for an ILI run without any additional pre-processing. For parameter (2), this implied the manual data curation and removal of outliers in the data. Such outliers resulted from unreliable ILI runs and were determined from the comments made in the dent registry as explained in Section 4.1.3 and from manually reviewing all 4,667 images. Therefore, a pristine dataset was composed of only the data points that had **zero** comments in the dent registry, implying that their analysis was smooth and without need for concern. The resultant 4 datasets were: (1) raw and all data; (2) raw and pristine data; (3) smooth and all data; and (4) smooth and pristine data. These datasets are summarized in Table 4.3.

The datasets were separated into three bins: training, validation, and testing. The purpose of the training data was to train the CNN model and determine the model weights, while the validation data was to evaluate the model and tune the hyperparameters. However, the testing data was set apart until the final CNN model was selected, at which point the testing data was used to check for generalization error [3]. This study followed a 70/20/10 split of the data; 70% for training, 20% for validation, and 10% for testing. The distribution of this data is displayed in Fig. 4.15. The testing data for all 4 datasets was set apart and used only once with the final CNN model after several comparisons were made between the datasets. The following section describes the development of the ML CNN model.

Table 4.3: Dent data for all vendors.

Dataset Name	Number of Dent Images	Training Size	Validation Size
Raw & All	4,683	3,455	759
Raw & Pristine	2,541	1,723	412
Smooth & All	4,683	3,455	759
Smooth & Pristine	2,541	1,723	412

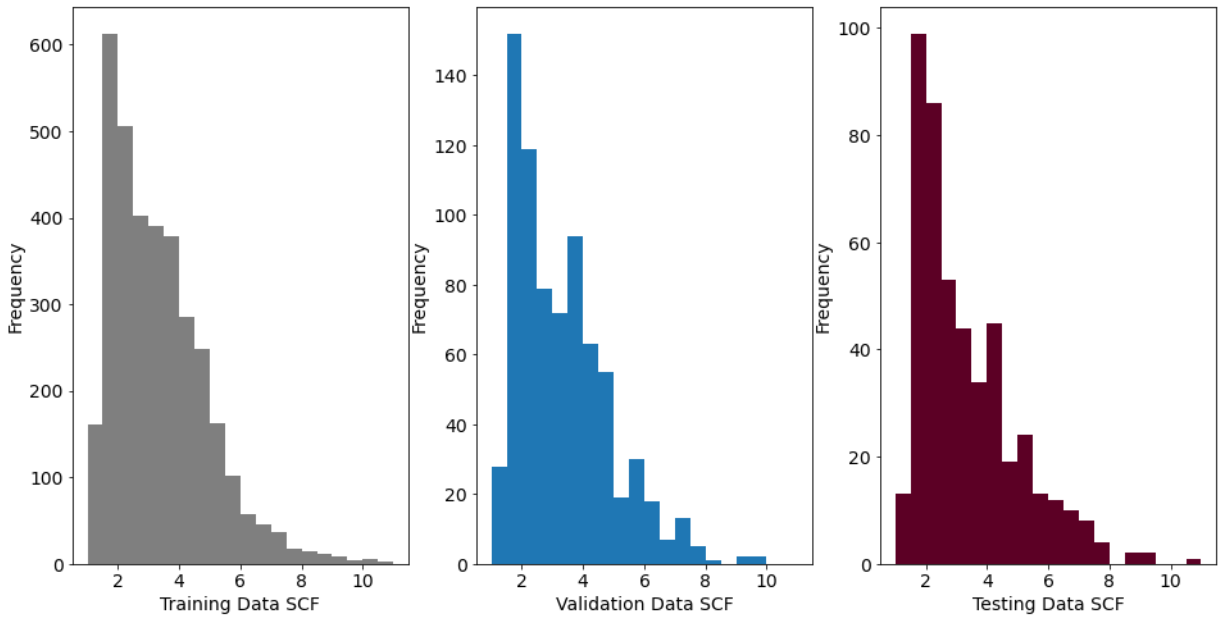


Figure 4.15: Distributions of 70% training, 20% validation, and 10% testing datasets

5. RESULTS AND DISCUSSION

This section examines the performance of the final model selected in Section 3.4 out of 180 CNN model iterations, and provides commentary for its generalization. The final model was trained on the **Raw & Pristine** dataset with 1,777 training points (70%) and 509 evaluation points (20%) out of the 2,541 total data points. Its distribution of SCFs is shown in Fig. 5.1. This dataset was selected instead of the other datasets described in Table 4.3 for the following reasons:

- Training on raw ILI data gives the best representation of exported data mid operation. This eliminates the need to run the raw ILI data through the smoothing portion of the pre-processing described in Section 4.2.3. Instead, the software could simply run the raw ILI data as is through the CNN model in order to make a prediction.
- Removing the need for data smoothing as part of the pre-processing can save computational power, thus resulting in faster SCF predictions.
- Performing data smoothing has the risk of resulting in dent shapes that either undermine or over emphasize the severity of the true dent shape. This would place less risk on operation of the CNN model, as there are no additional steps necessary to manipulate the data.
- When comparing the performance of CNN models trained on the smooth datasets (i.e., Smooth & All, Smooth & Pristine), the improvements on SCF prediction quality do not exceed that of using the raw datasets.
- Lastly, using a Pristine dataset is acceptable for the purposes of this study. This is reasoned by the fact that some dents will require human intervention despite the best efforts of a CNN model. It would be better to develop a pre-assessment tool capable of giving trustworthy predictions for a subset of ideal dents, than to use a tool lacking the same level of trust but useful for many more dents – which would still a small subset of the global population dent statistics due to the limited data available for this study.

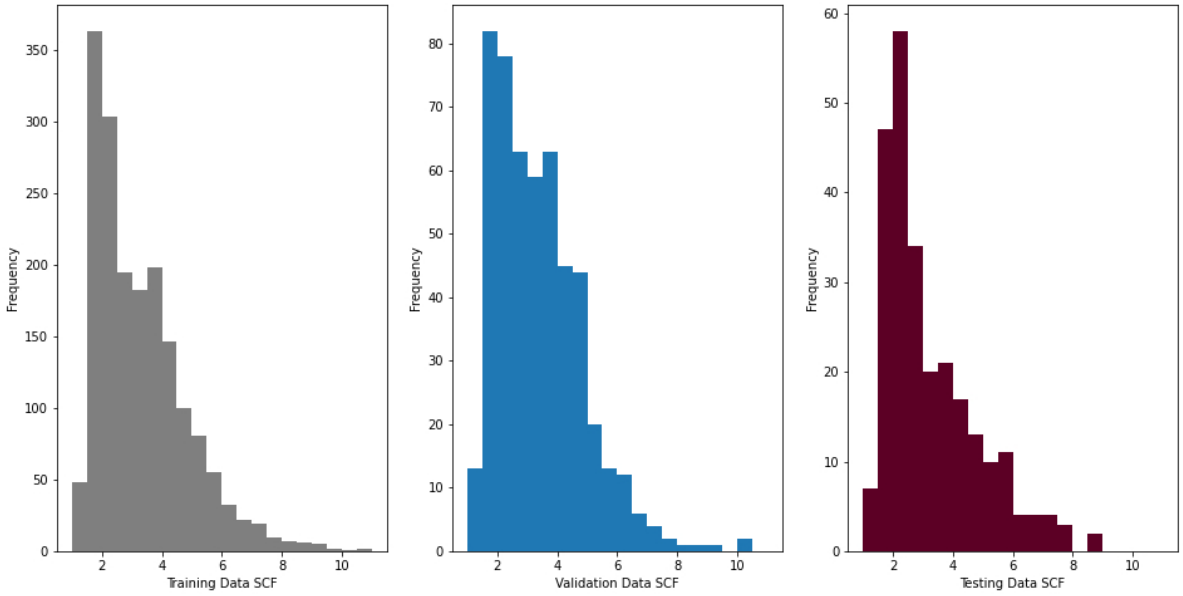


Figure 5.1: Distribution of SCFs among Training (70%), Validation (20%), and Test (10%) sets.

The CNN model resulted with the evaluation criteria values as shown in Table 5.1. The coefficient of determination R^2 alone is not an excellent demonstration of good prediction quality, but coupled with the other evaluation criteria, it becomes stronger evidence for this model. Both the validation loss and metric are slightly above the training loss and metric values, respectively, which is desirable as this serves as a good check to ensure that the model did not overfit. It is worth noting that there is a significant Max Error of 1.216. While this appears concerning, the Max Error Percentage was only 44.16%, therefore this particular sample point had a target SCF of 2.754 and a predicted SCF of 3.970. Nevertheless, these values were the lowest of all 180 model iterations, as shown in Fig. 5.2.

The training history of the model is demonstrated in Fig. 5.3. The validation metric and loss appear to perform better than the training metric and loss, respectively, at the beginning of the training epochs but appear to reach a similar value towards the end. This may be explained by regularization, which is method to prevent overfitting by controlling the model complexity varying by epoch. As displayed, the number of 50 epochs was considered an optimal amount of training iterations, but for the sake of comparison, one model was trained up to 150 epochs to observe overfitting and is displayed in Fig. A.1.

Table 5.1: Results of evaluation criteria for final CNN model using validation data. Lower values are desirable for all except R^2.

Evaluation Criteria	Final Value
Training Loss	0.121
Training Metric	0.258
Validation Loss	0.140
Validation Metric	0.290
RMSE	0.374
Max Absolute Error	1.216
Max Absolute Error Percentage	44.16%
Training Time (s)	155
R^2	0.935

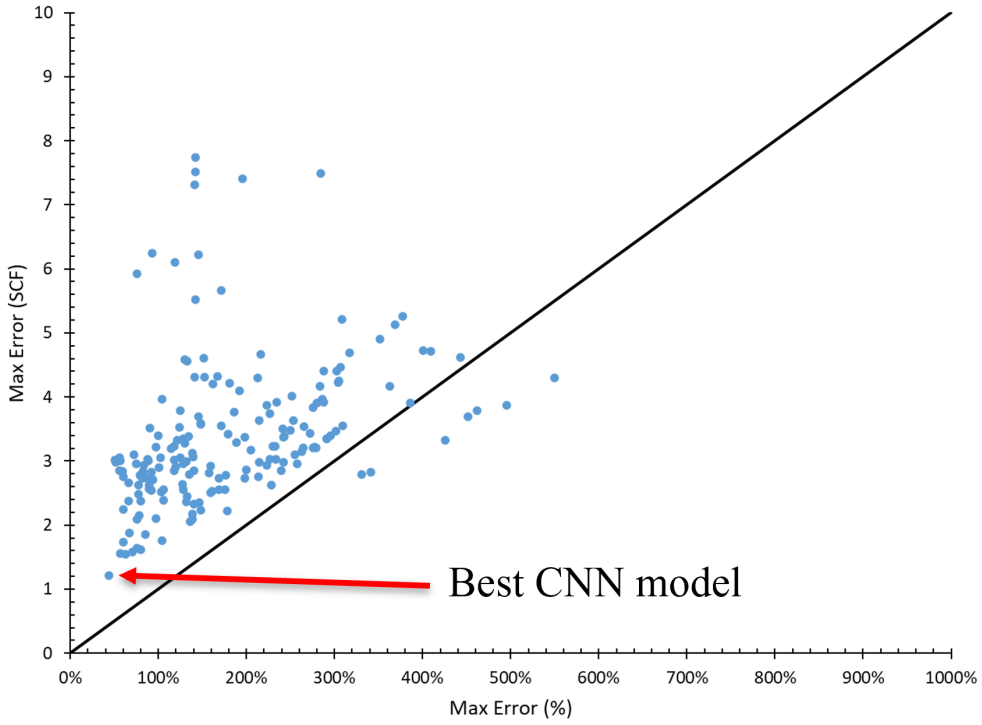


Figure 5.2: Comparison of Max Error and Max Error Percentage between all 180 models.

In order to observe how well the model performed on the validation data, a unity plot such as Fig. 5.5 was made demonstrating the predicted SCF values vs the truth data (or target values). This graph was accompanied by the distribution of predicted SCFs to observe how closely it matched distributions as those in Fig. 5.1. The unity plot showed excellent agreement between the 509

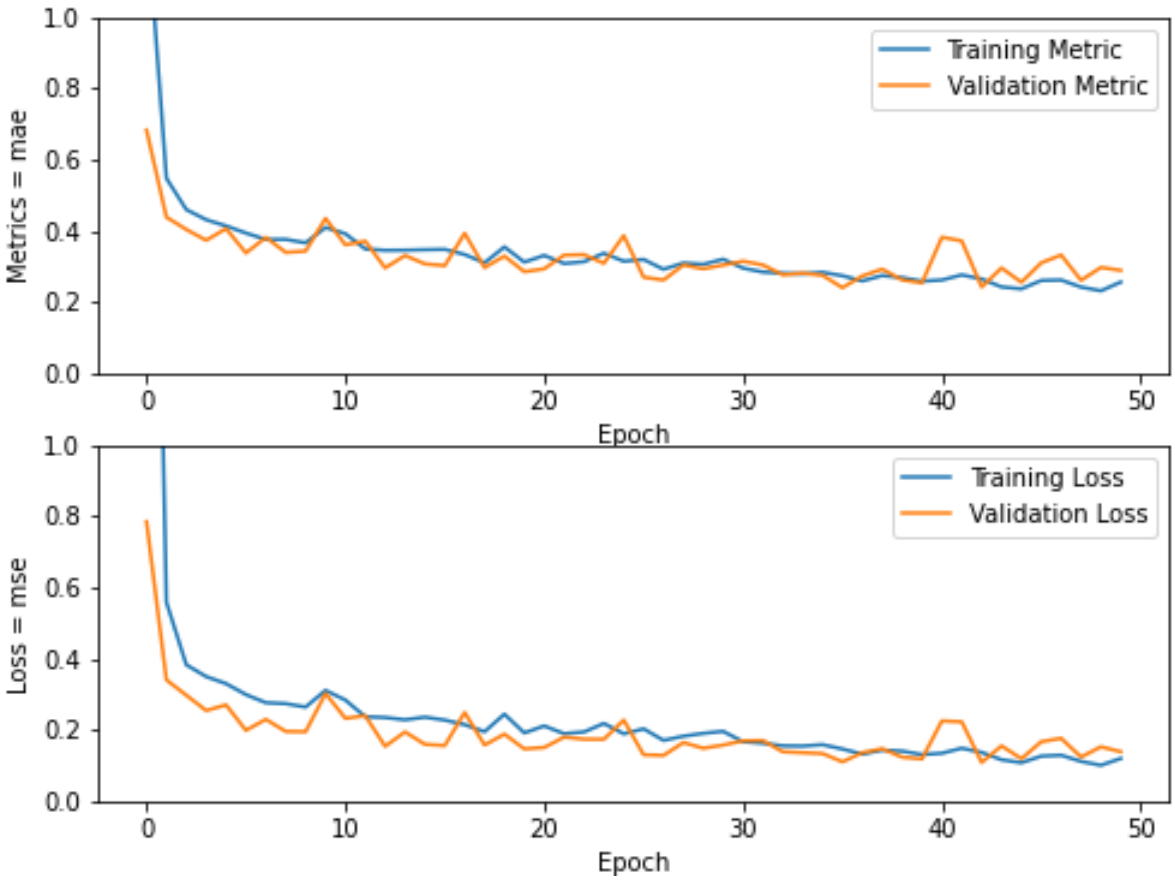


Figure 5.3: Training and validation of the model over 50 epochs.

predicted SCFs and their corresponding target value. The difference error between predicted values and target values, otherwise the residual error ($Residual = Prediction - Target$), is shown in Fig. 5.6 along with the distribution of those residuals. A desired distribution of residuals is as close to zero as possible with the average value at zero. In this case, a careful examination of these residuals show that while they are very close to zero, the average is slightly shifted towards the negative range. The significance of these residual errors are made easier to see by plotting the percent error ($\%Error = 100 * (Prediction - Target)/Target$) in Fig. 5.7, which scales the error by the relative severity of the target SCF value. As expected, there is a greater range of percent error with smaller target SCFs due to the fairly consistent distribution of residuals. This demonstrates that the model performs comparably with smaller SCFs (1 to 4) as it does with larger SCFs (4 to 7), and even maintains acceptable performance with very high SCFs (7 or more).

Figure 5.4 contains a series of 9 different validation dent samples and the SCF prediction from the CNN model and the target SCF values. There are several different dent shapes in these few examples alone out of the 509 validation samples. As a reminder, the contour map color scale uses red-yellow-green (RYG), where red is the most negative value and green is the most positive value of the normalized residual ILI data as covered in Section 4.2.2.

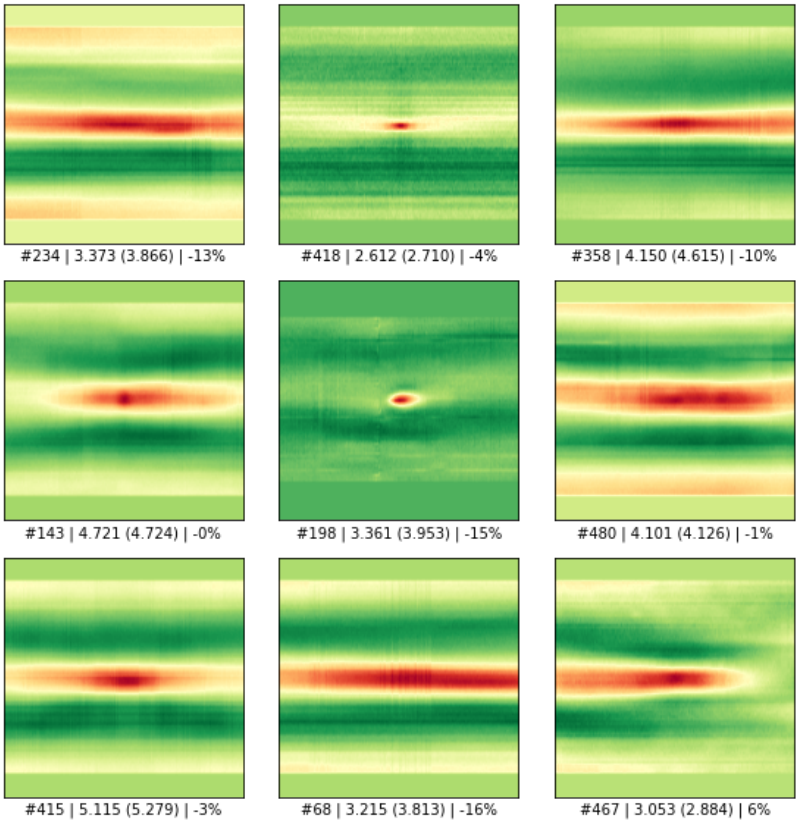


Figure 5.4: Prediction of SCF values from 9 random validation data points. Image label contains: # Dent Number | Predicted SCF (True SCF) | Percent Error.

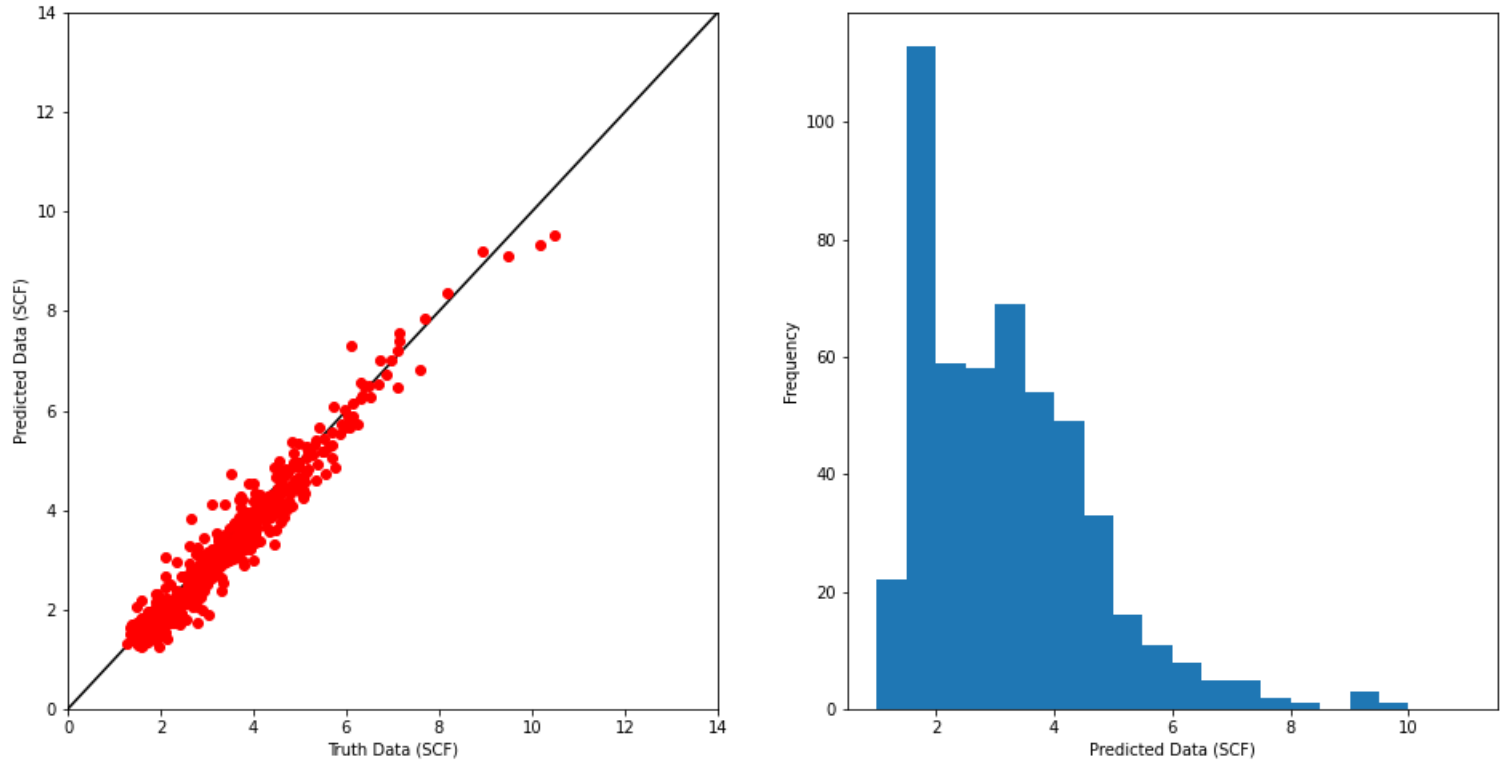


Figure 5.5: Unity plot and distribution of SCF predictions for validation data. The black diagonal line represents a perfect 1:1 agreement between predicted and truth (target) values.

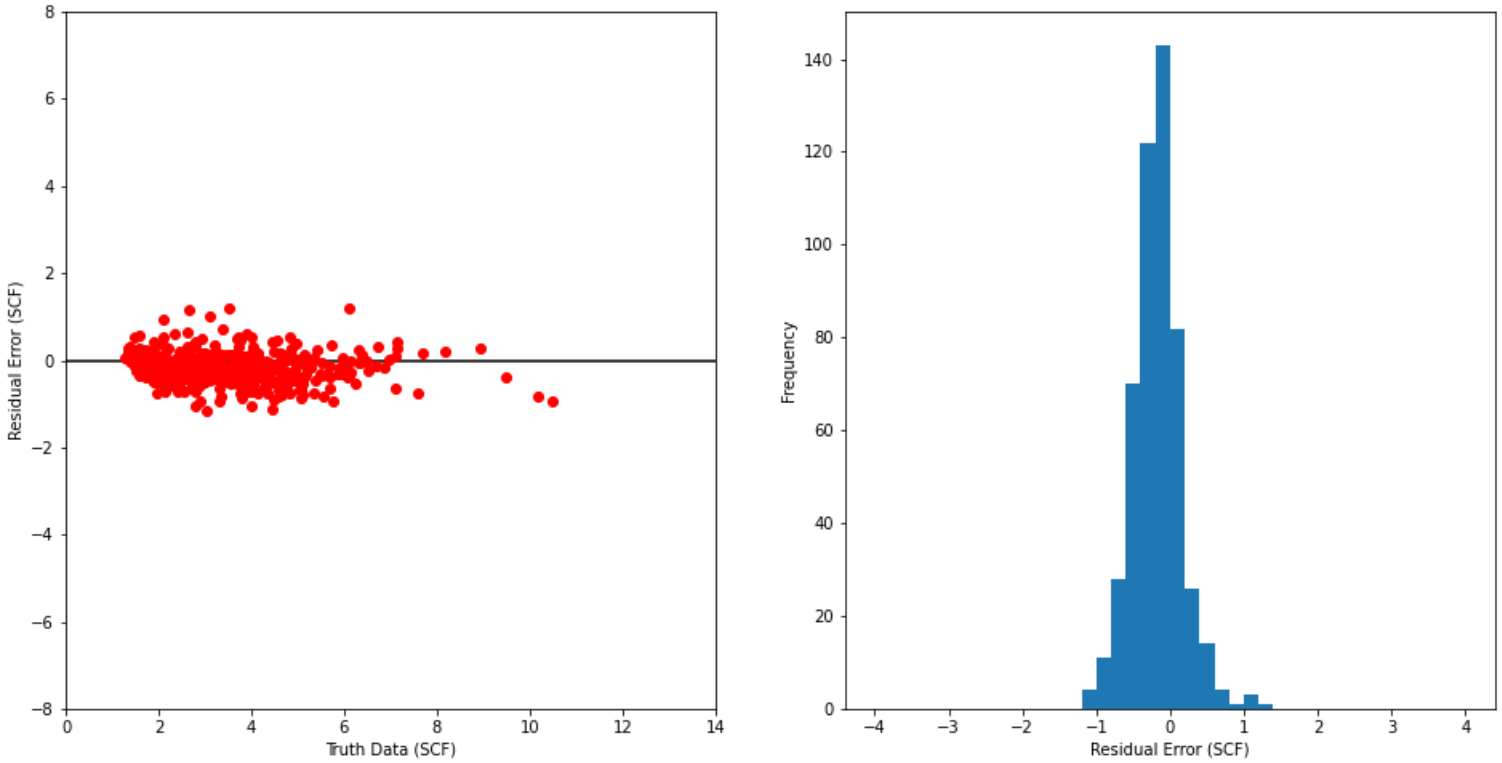


Figure 5.6: Residual plot and distribution of SCF predictions for validation data. The black horizontal line represents zero residual error.

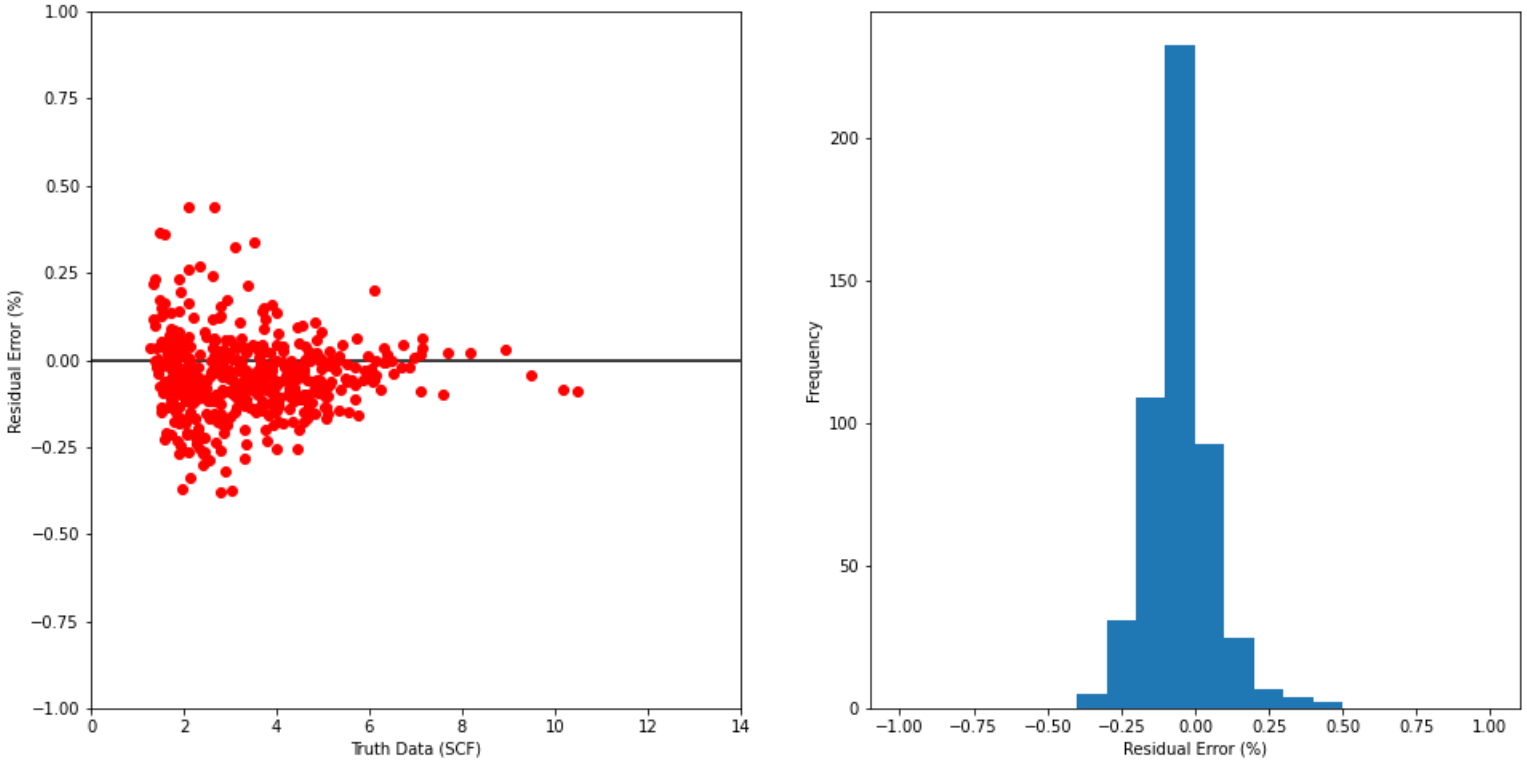


Figure 5.7: Percent residual plot and distribution of SCF predictions for validation data. The black horizontal line represents zero percent residual error.

In general, it is difficult to understand what is truly happening inside any ML model as it is essentially a black box that can be tested by intelligent methods of trial and error. But with the intent of gaining additional insight into what the CNN model has learned, the convolutional layers and all their filters are visualized in Fig. 5.9 and 5.10 when taking the example dent in Fig. 5.8 as the data input. Several of these filters appear to be activated by the deepest region of the dent image, while others are activated by the outline of the center, and even yet others are activated by the outside region of a dent. Effectly, the CNN model has *learned* these features and responded to them accordingly. As the layer depths increment, the filters become more and more abstract and the sparsity of activations increases. Therefore, the higher layers appear to carry less information about the original input into the CNN model.

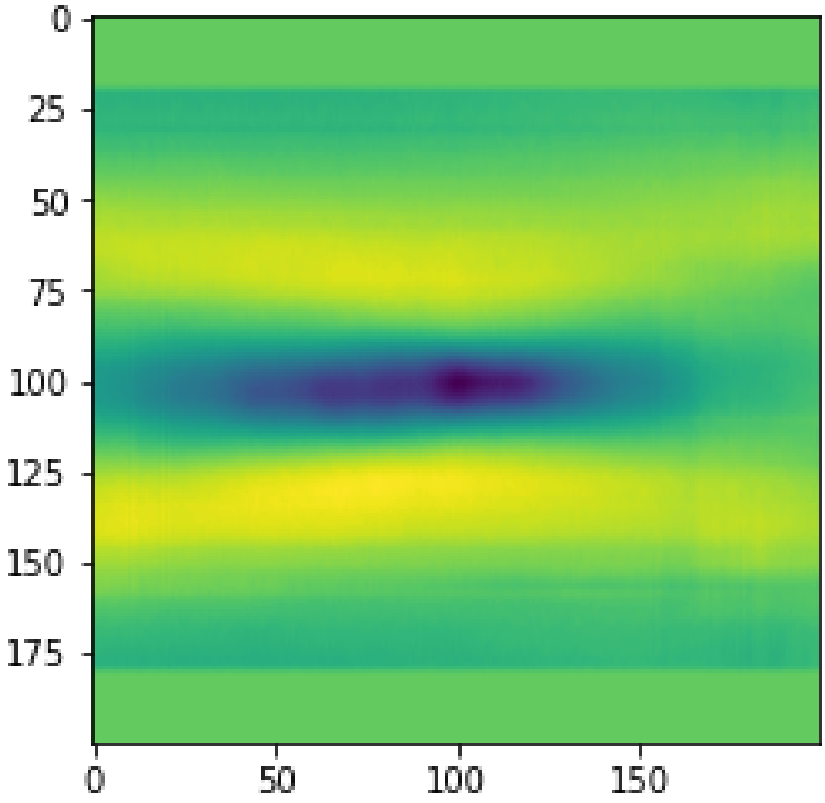


Figure 5.8: Example dent used for the visualization of all filter activations in Fig. 5.9 and 5.10.

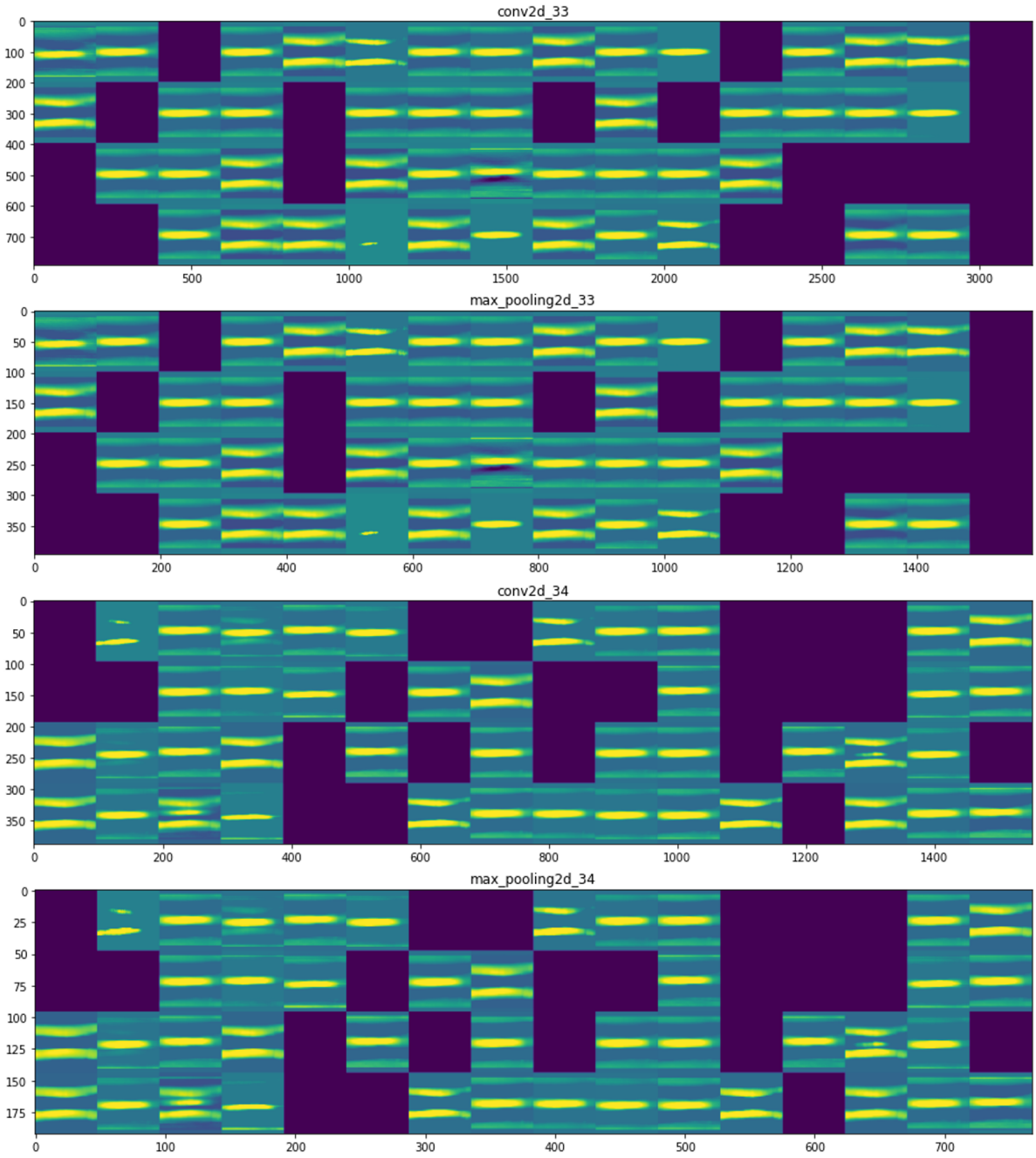


Figure 5.9: Visualization of all the filter activations in the CNN (Part 1 of 2).

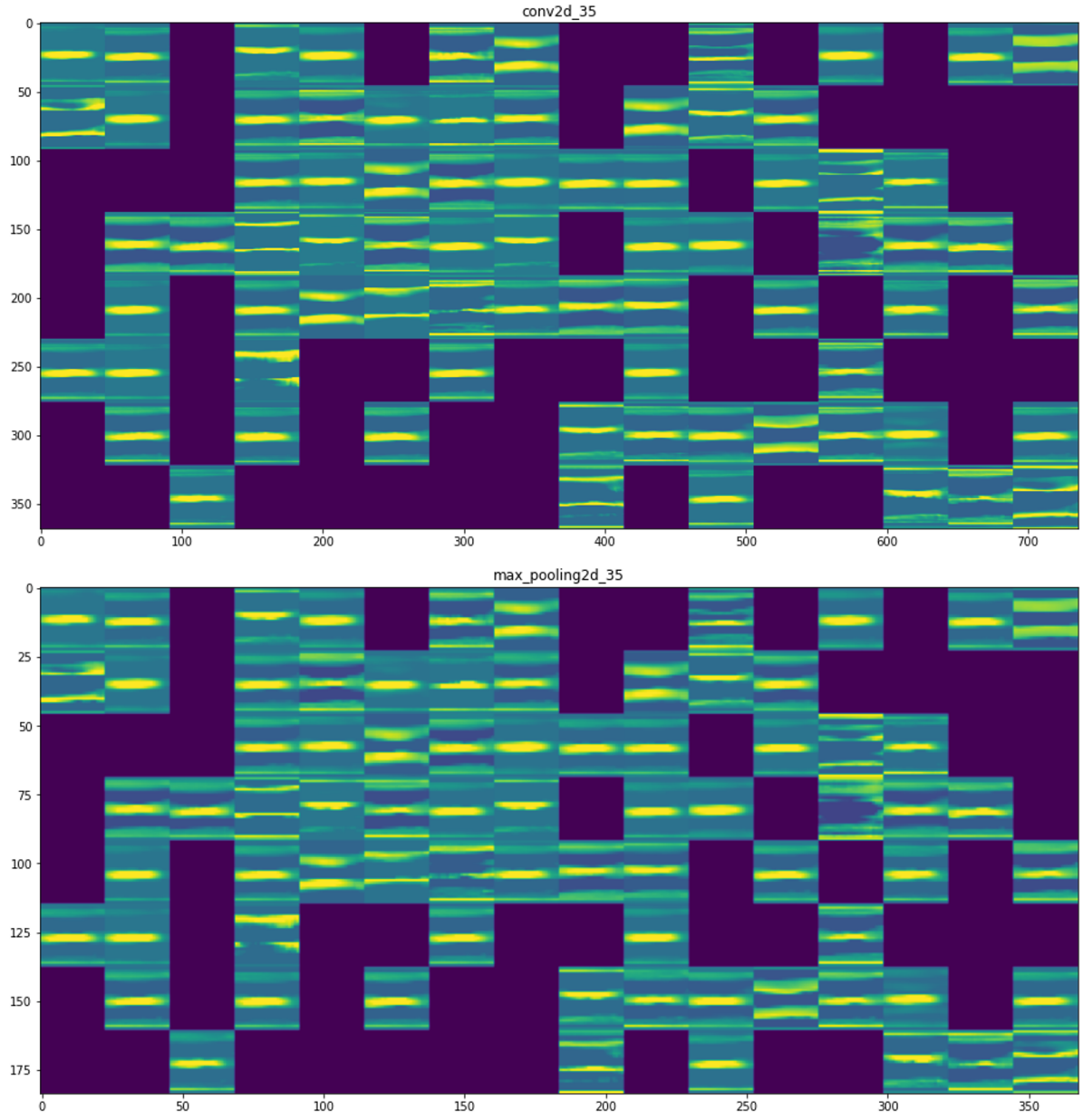


Figure 5.10: Visualization of all the filter activations in the CNN (Part 2 of 2).

5.1 Test for Generalization

The final evaluation of the CNN model happens by making predictions on the test data to check for generalization. This test data was 10% of the entire data with only 255 data points out of the 2,541 total data points. Unlike the validation dataset, the CNN model hyperparameters were not tuned by observing performance on this testing dataset. What is unique about this dataset is that it was left untouched for the entire duration of the study – therefore, neither the author nor the CNN model has seen this data before. The only characteristics known about it is that it follows a similar distribution of SCFs, as shown in Fig. 5.1. This is significant because it permits an appropriate evaluation of the range of SCF values that the model was trained and tuned on. Therefore, it checks for how well the CNN model can generalize on new data.

As observed in Figures 5.11, 5.12, 5.13, and 5.14, the selected CNN model appears to perform well and follows a similar distribution of residuals as with the validation data. There are a few data points with greater than usual residual error; the Max Error Percentage was 67.44% for a sample point with a target SCF of 2.307 and a predicted SCF of 4.330. Despite these few outliers, the CNN model predictions on the test data appears to perform very comparably to the validation data. In addition, the evaluation criteria were calculated using the test data results and are displayed in Table 5.2. While the Max Error and Max Error Percentage have increased, the RMSE is still below 0.5 and R^2 above 0.90, thus meeting the ideal metrics described in Section 3.2.

After this careful review, the final CNN model was selected as the architecture that can be used for developing a pre-assessment tool to predict SCF values based on pipeline ILI data.

Table 5.2: Results of evaluation criteria for final CNN model using test data. Lower values are desirable for all except R^2.

Evaluation Criteria	Final Value
RMSE	0.418
Max Absolute Error	2.143
Max Absolute Error Percentage	87.69%
R^2	0.929

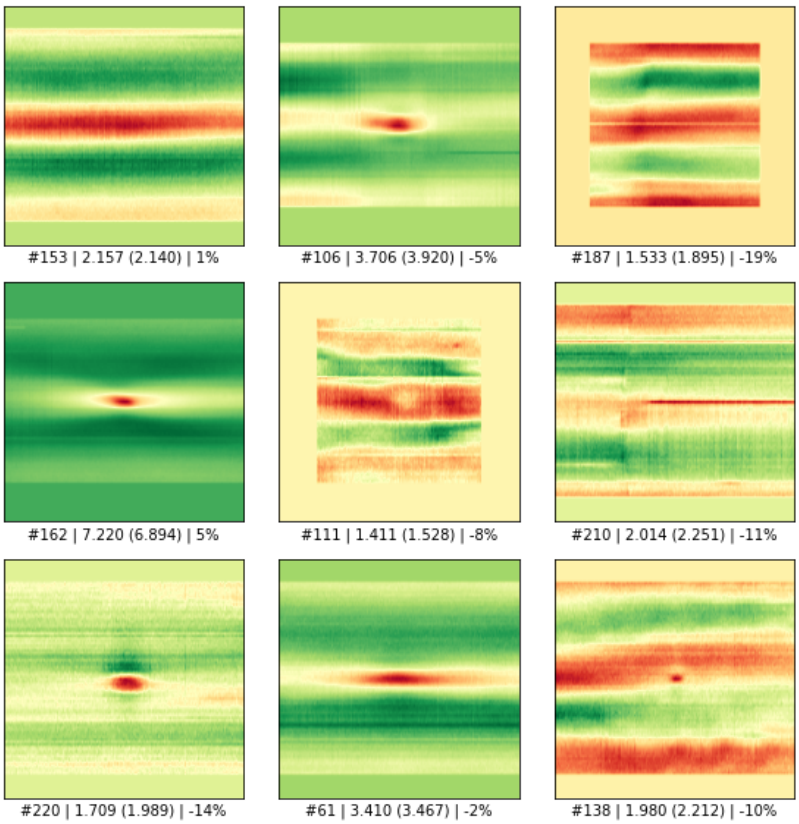


Figure 5.11: Prediction of SCF values from 9 random test data points. Image label contains: # Dent Number | Predicted SCF (True SCF) | Percent Error.

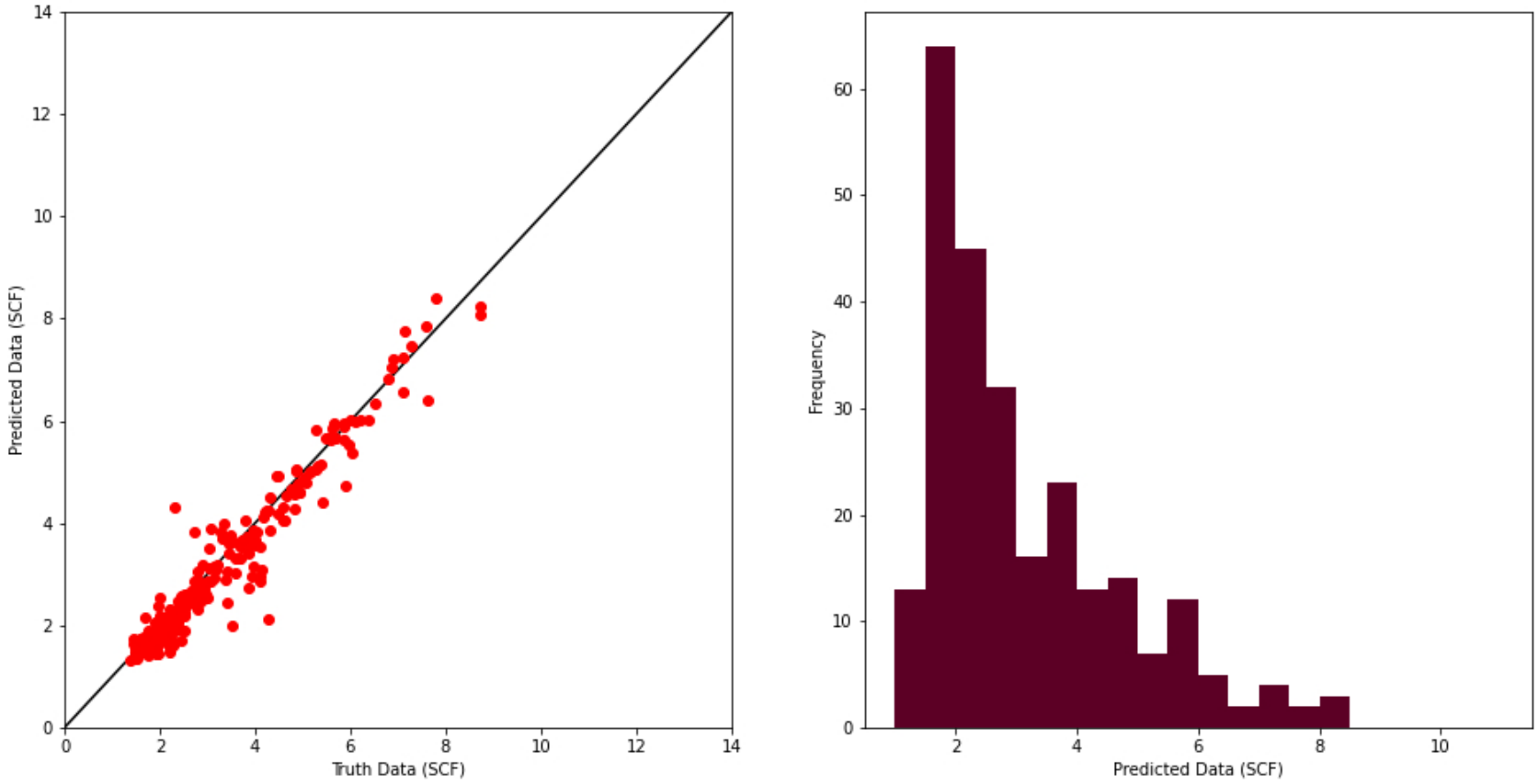


Figure 5.12: Unity plot and distribution of SCF predictions for test data. The black diagonal line represents a perfect 1:1 agreement between predicted and truth (target) values.

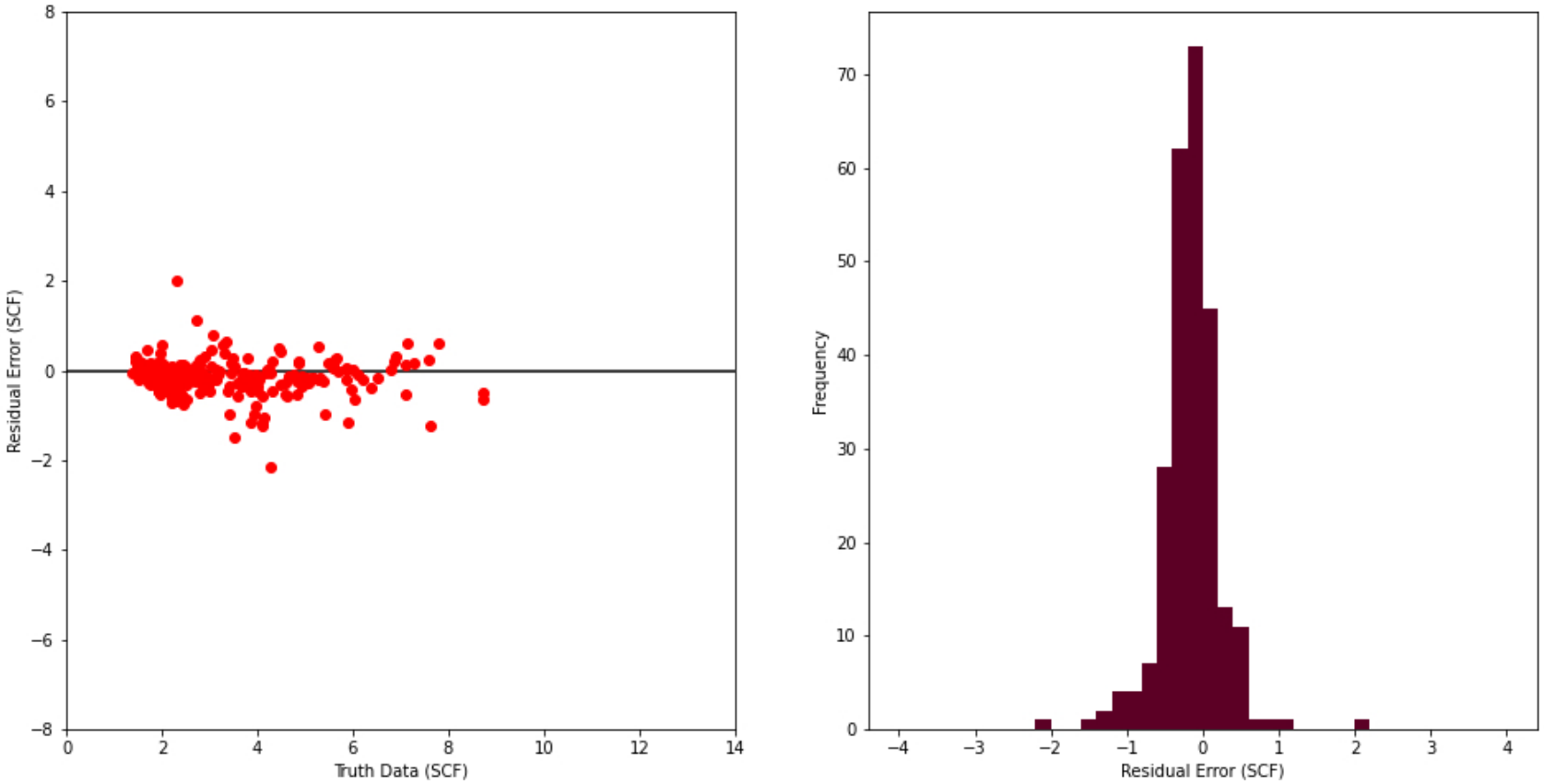


Figure 5.13: Residual plot and distribution of SCF predictions for test data. The black horizontal line represents zero residual error.

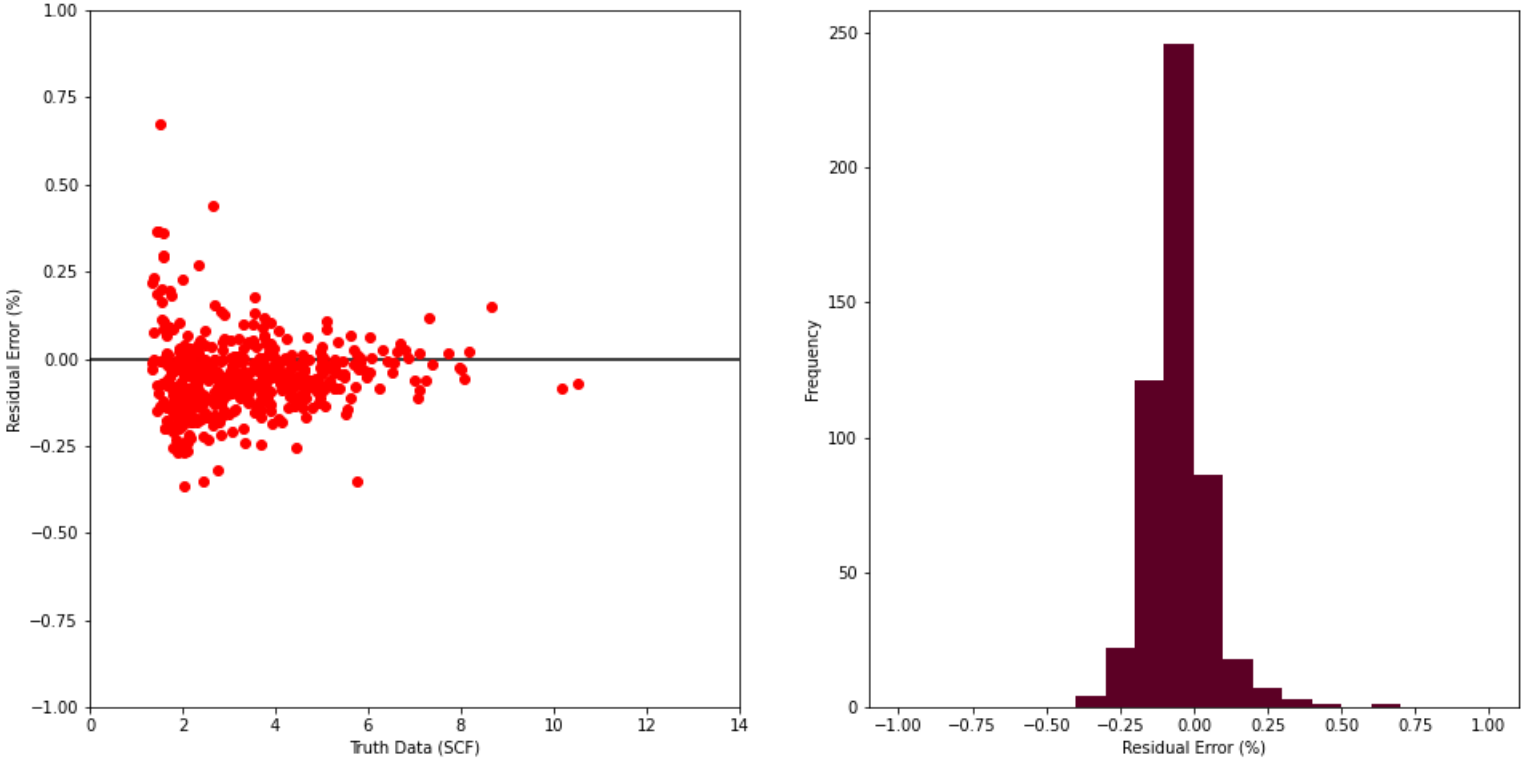


Figure 5.14: Percent residual plot and distribution of SCF predictions for test data. The black horizontal line represents zero percent residual error.

6. CONCLUSION AND RECOMMENDATIONS FOR FUTURE STUDY

Oil and gas pipelines make up a significant part of the energy infrastructure in the United States such that over time and use, they naturally decay and are subject to dent defects that decrease the integrity of the system. And with majority of pipelines being over 50 years old, this is becoming a growing concern to maintain energy stability. Therefore, existing and rising in-line inspection (ILI) tools should make use of the technology that is available, and make use of powerful tools such as Artificial Intelligence and Machine Learning.

This thesis reviewed the significance of quick and reliable pre-assessment tools for determining the stress concentration factor (SCF) of a pipeline dent defect. This is in partial fulfillment of the fitness-for-service (FFS) that every pipeline must pass. To predict these SCF values, a Convolutional Neural Network (CNN) model was developed using 2,541 data points that achieved evaluation criteria Root Mean Squared Error (RMSE) of 0.374, R^2 of 0.935, and Max Error Percentage of 44.16% as part of the validation process. Not only that, but the CNN model demonstrated excellent generalization by reaching comparable evaluation criteria with the test dataset, which was never before seen data only used once for the final model selection. These results were RMSE of 0.418, R^2 of 0.929, and Max Error Percentage of 87.69%.

This study concludes that the final CNN model is capable of correctly assessing the SCF of raw ILI dent data, but is targeted for pristine samples that have minimal complexities. Therefore, if it is a dent that would not require additional human interaction for assessment, then this CNN model may satisfy the requirements for a quick and reliable SCF pre-assessment tool. If used properly, the tool may provide ILI vendors and pipeline operators with the immediate guidance over how to prioritize dented pipeline segments based on SCF values.

6.1 Future Study

With the onset of changes to the energy infrastructure, there is plenty of potential for future study in the area of oil and gas pipeline predictive maintenance and SCF assessments. This study was limited to the assumptions and constraints made in Section 3.1, therefore there is plenty of room to expand and grow for future development. The following are some recommendations for encouraging future studies and providing initial guidance:

- There is plenty of variability in ILI data due to the random and systematic error for each ILI tool. Future studies may consider the impact of this variability and perform a sensitivity analysis with Monte Carlo simulation to observe the worst possible scenarios for SCF predictions when considering ILI data variability.

- The ILI runs collect additional information including rated pipe strength, material properties, and other variables that may be used as additional inputs to a machine learning (ML) model. These parameters may provide a greater understanding of the pipeline system in order to assess the dent besides only the dent shape.

- There are other Deep Learning (DL) architectures that may prove useful for studies with limited data as this one. A future study may consider using a Generative Adversarial Network (GAN) and validate the type of dent shapes generated through coupled finite element analysis (FEA). It is always a good idea to validate anything representing or simulating a real-world system.

- Considering that the distribution of prediction error follows a zero-mean normal distribution, it is possible to develop a prediction and/or confidence interval in order to give insight into the uncertainty of a SCF prediction. Future studies are encouraged to expand into this statistical analysis.

REFERENCES

[1] U.S. GAO, "Critical Infrastructure Protection: Key Pipeline Security Documents Need to Reflect Current Operating Environment," June 2019. Available: https://www.gao.gov/products/gao-20-299.

[2] ENTEGRA, "Ultra-High Resolution In-Line Inspection Services," 2022. Available: https://www.entegrasolutions.com/entegra-services/.

[3] F. Chollet, *Deep Learning with Python*. Shelter Island, NY: Manning Publications Co., 2018.

[4] U.S. Department of Transportation Pipeline and Hazardous Materials Safety Administration (PHMSA), "General Pipeline FAQs," November 2018. Available: https://www.phmsa.dot.gov/faqs/general-pipeline-faqs.

[5] J. Conca, "Which is safer for transporting crude oil: Rail, truck, pipeline or boat?." Forbes Magazine, October 2018. Available: https://www.forbes.com/sites/jamesconca/2018/10/11/which-is-safer-for-transporting-crude-oil-rail-truck-pipeline-or-boat/?sh=275089437b23.

[6] Library of Congress, "Oil and gas industry: A research guide. modes of transportation." Available: https://guides.loc.gov/oil-and-gas-industry/midstream/modes.

[7] Statista, "Average Brent Price 1976-2022," March 2022. Available: https://www.statista.com/statistics/262860/uk-brent-crude-oil-price-changes-since-1976/.

[8] International Energy Agency, "Key World Energy Statistics," 2016.

[9] American Gas Association (AGA), "How Does the Natural Gas Delivery System Work?," 2022. Available: https://www.aga.org/natural-gas/delivery/how-does-the-natural-gas-delivery-system-work-/.

[10] U.S. Department of Transportation Pipeline and Hazardous Materials Safety Administration (PHMSA), "Pipeline Failure Causes," 2018. Available: https://www.phmsa.dot.gov/incident-reporting/accident-investigation-division/pipeline-failure-causes.

[11] J. Pless, "Making State Gas Pipelines Safe and Reliable: An Assessment of State Policy." National Conference of State Legislatures (NCSL), March 2011. Available: https://www.ncsl.org/research/energy/state-gas-pipelines-federal-and-state-responsibili.aspx.

[12] ROSEN, "Fitness-For-Service Assessment (FFS)," 2022. Available: https://www.rosen-group.com/global/solutions/services/service/Fitness-For-Service.html.

[13] California State Assembly: Committee on Natural Resources, "Oil Pipeline Testing Methods," *Joint Informational Hearing: Senate Select Committee on Refugio Oil Spill, Assembly Natural Resources Committee, and Assembly Governmental Organization Committee*, October 2015. Available: https://antr.assembly.ca.gov/2015hearings.

[14] H. Liu, Z. Liu, B. Taylor, and H. Dong, "Matching pipeline in-line inspection data for corrosion characterization," *NDT E International*, vol. 101, pp. 44–52, 2019.

[15] V. Semiga, "Fatigue Considerations for Natural Gas Transmission Pipelines." BMT Fleet Technology Limited, June 2016.

[16] Corrosionpedia, "Stress Concentration Factor (K_t)," 2020. Available: https://www.corrosionpedia.com/definition/1035/stress-concentration-factor-kt.

[17] R. Dotson. Pipeline Integrity Engineer at ADV Integrity, Inc. Subject matter expert on pipeline in-line inspection tools.

[18] Stress Engineering Services, Inc, "Defect Characterization and Assessment," 2021. Available: https://www.stress.com/services/energy/midstream/defect-characterization-and-assessment/.

[19] U.S. Department of Transportation Pipeline and Hazardous Materials Safety Administration (PHMSA), "Source Data," 2021. Available: https://www.phmsa.dot.gov/data-and-statistics/pipeline/source-data.

[20] ASME, "B31.8 Gas Transmission and Distribution Piping Systems: Appendix R, Estimating Strain in Dents," 2018. ASME Std., pp. 185-186.

[21] V-Soft Consulting, "6 Ways AI is Transforming the Oil and Gas Industry," 2021. Available: https://blog.vsoftconsulting.com/blog/ais-role-in-oil-and-gas-industry.

[22] M. Dabiri, M. Ghafouri, H. R. Rohani Raftar, and T. Björk, "Neural network-based assessment of the stress concentration factor in a T-welded joint," *Journal of Constructional Steel Research*, vol. 128, pp. 567–578, 2017.

[23] Medium, "Deep Learning Fundamental Terms," 2022. Available: https://medium.com/@emrahyurtlu/deep-learning-fundamental-terms-9f12a17a9aea.

[24] IBM, "AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the difference?," 2020. Available: https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks.

[25] Towards Data Science, "What is Feature Engineering — Importance, Tools and Techniques for Machine Learning," 2021. Available: https://towardsdatascience.com/what-is-feature-engineering-importance-tools-and-techniques-for-machine-learning-2080b0269f10.

[26] Simplilearn, "Top 10 deep learning algorithms you should know in 2022," 2022. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm.

[27] B. T. Bastian, J. N, S. K. Ranjith, and C. V. Jiji, "Visual inspection and characterization of external corrosion in pipelines using deep neural network," *NDT E International*, vol. 107, p. 102134, 2019.

[28] R. Dotson, C. Holliday, L. Torres, and D. Hagan, "An Authoritative Comparison of Remaining Life Assessments for Pipeline Dents," vol. 1, American Society of Mechanical Engineers (ASME), 2018.

[29] R. Dotson, R. Sager, F. Curiel, and M. Le Roy, "Judge Me by My Size, Do You? How Reliable Are Dent Assessments Based on ILI Data?," vol. 1, American Society of Mechanical Engineers (ASME), 2020.

[30] R. L. Dotson, M. Ginten, C. Alexander, J. J. Bedoya, and K. Schroeer, "Combining high-resolution in-line geometry tools and finite element analysis to improve dent assessments," (Houston, USA), Pipeline Pigging and Integrity Management Conference, February 2014.

[31] J. M. Race, J. V. Haswell, R. Owen, and B. Dalus, "UKOPA Dent Assessment Algorithms: A Strategy for Prioritising Pipeline Dents," vol. 1, pp. 923–933, IPC, 2010.

[32] B. Pinheiro, C. G. Soares, and I. Pasqualino, "Generalized expressions for stress concentration factors of pipeline plain dents under cyclic internal pressure," *International Journal of Pressure Vessels and Piping*, vol. 170, pp. 82–91, 2019.

[33] B. Pinheiro, I. P. Pasqualino, and S. Barros da Cunha, "Stress Concentration Factors of Dented Pipelines," vol. 1, IPC, 2006.

[34] M. Beller, C. Mattheck, and J. Zimmermann, "Stress Concentrations In Pipelines Due To The Presence Of Dents," ISOPE, 1991.

[35] R. L. Dotson, M. Ginten, C. Alexander, J. J. Bedoya, and K. Schroeer, "Combining high-resolution in-line geometry tools and finite-element analysis to improve dent assessments," vol. 13, pp. 113–123, Scientific Surveys Ltd, 2014.

[36] B. Wang, W. Zhao, Y. Du, G. Zhang, and Y. Yang, "Prediction of fatigue stress concentration factor using extreme learning machine," *Computational Materials Science*, vol. 125, pp. 136–145, 2016.

[37] A. Raja, S. T. Chukka, and R. Jayaganthan, "Prediction of Fatigue Crack Growth Behaviour in Ultrafine Grained Al 2014 Alloy Using Machine Learning," *Metals*, vol. 10, p. 1349, 2020.

[38] J. Ji, C. Zhang, J. Kodikara, and S. Yang, "Prediction of stress concentration factor of corrosion pits on buried pipes by least squares support vector machine," *Engineering Failure Analysis*, vol. 55, pp. 131–138, 2015.

[39] T. Yamaguchi, and H. Okuda, "Prediction of stress concentration at fillets using a neural network for efficient finite element analysis," *Mechanical Engineering Letters*, vol. 6, pp. 20–00318–20–003, 2020.

[40] G. C. Silva, V. C. Beber, and D. B. Pitz, "Machine learning and finite element analysis: An integrated approach for fatigue lifetime prediction of adhesively bonded joints," *Fatigue & Fracture of Engineering Materials & Structures*, vol. 44, pp. 3334–3348, 2021.

[41] H. C. Phan, and A. S. Dhar, "Predicting pipeline burst pressures with machine learning models," *International Journal of Pressure Vessels and Piping*, vol. 191, p. 104384, 2021.

[42] H. C. Phan, and H. T. Duong, "Predicting burst pressure of defected pipeline with Principal Component Analysis and adaptive Neuro Fuzzy Inference System," *International Journal of Pressure Vessels and Piping*, vol. 189, p. 104274, 2021.

[43] S. Chatterjee, and A. Keprate, "Predicting remaining fatigue life of topside piping using deep learning," vol. 1, Institute of Electrical and Electronics Engineers (IEEE), 2021.

[44] M. R. Kaloop, P. Samui, J. Kim, J. W. Hu, and A. Ramzy, "Stress Intensity Factor Prediction on Offshore Pipelines using Surrogate Modeling Techniques," *Case Studies in Construction Materials*, vol. 16, p. e01045, 2022.

[45] Y. Xie, F. Xing, X. Kong, H. Su, and L. Yang, "Beyond Classification: Structured Regression for Robust Cell Detection Using Convolutional Neural Network," pp. 358–365, 2015.

[46] X. Ren et al., "Regression Convolutional Neural Network for Automated Pediatric Bone Age Assessment From Hand Radiograph," *IEEE Journal of Biomedical and Health Informatics*, vol. 23, pp. 2030–2038, 2019.

[47] D. Oh, J. Race, S. Oterkus, and B. Koo, "Burst Pressure Prediction of API 5L X-Grade Dented Pipelines Using Deep Neural Network," *Journal of Marine Science and Engineering*, vol. 8, p. 766, 2020.

[48] J. Pyo et al., "A convolutional neural network regression for quantifying cyanobacteria using hyperspectral imagery," *Remote Sensing of Environment*, vol. 233, p. 111350, 2019.

[49] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems* (F. Pereira and C.J. Burges and L. Bottou and K.Q. Weinberger, ed.), vol. 25, Curran Associates, Inc., 2012.

[50] T. Shimobaba, T. Kakue, and T. Ito, "Convolutional Neural Network-Based Regression for Depth Prediction in Digital Holography," IEEE, 2018.

[51] C. Kantzos, J. Lao, and A. Rollett, "Design of an interpretable Convolutional Neural Network for stress concentration prediction in rough surfaces," *Materials Characterization*, vol. 158, p. 109961, 2019.

[52] I. Perez, A. Madariaga, P. J. Arrazola, M. Cuesta, and D. Soriano, "An analytical approach to calculate stress concentration factors of machined surfaces," *International Journal of Mechanical Sciences*, vol. 190, p. 106040, 2021.

[53] G. Rezende Bessa Ferreira, P. Aida Sesini, L. Paulo Brasil de Souza, A. Conci Kubrusly, and H. Vicente Hultmann Ayala, "Corrosion-like defect severity estimation in pipelines using convolutional neural networks," pp. 01–07, Institute of Electrical and Electronics Engineers (IEEE), 2021.

[54] D. Yang, M. Xiong, T. Wang, and G. Lu, "Percussion-Based Pipeline Ponding Detection Using a Convolutional Neural Network," *Applied Sciences*, vol. 12, p. 2127, 2022.

[55] J. Feng, F. Li, S. Lu, J. Liu, and D. Ma, "Injurious or Noninjurious Defect Identification From MFL Images in Pipeline Inspection Using Convolutional Neural Network," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, pp. 1883–1892, 2017.

[56] H. Kaur, H. S. Pannu, and A. K. Malhi, "A Systematic Review on Imbalanced Data Challenges in Machine Learning," *ACM Computing Surveys*, vol. 52, pp. 1–36, 2020.

[57] Y. Yang, K. Zha, Y. Chen, H. Wang, and D. Katabi, "Delving into Deep Imbalanced Regression," vol. 139, pp. 11842–11851, PMLR, 2022.

[58] SimuTech Group, "Why is Ansys meshing important for structural FEA and fluid CFD simulations?," 2021. Available: https://simutechgroup.com/why-is-meshing-important-for-fea-fluid-simulations/.

[59] V. C. Kagawade, and S. A. Angadi, "Savitzky-Golay filter energy features-based approach to face recognition using symbolic modeling," *Pattern analysis and applications : PAA*, vol. 24, pp. 1451–1473, 2021.

[60] B. F. Gregorsky, B. Hamann, and K. I. Joy, "Reconstruction of B-spline surfaces from scattered data points," IEEE Comput. Soc, 2000.

[61] M. Rouhani, A. D. Sappa, and E. Boyer, "Implicit B-Spline Surface Reconstruction," *IEEE Transactions on Image Processing*, vol. 24, pp. 22–32, 2015.

[62] Y. Kwak, D. Kim, H. Ham, and J. Park, "Convolutional neural network trained with synthetic pseudo-images for detecting an acoustic source," *Applied Acoustics*, vol. 179, p. 108068, 2021.

[63] K. M. Koo, and E. Y. Cha, "Image recognition performance enhancements using image normalization," *Human-centric Computing and Information Sciences*, vol. 7, 2017.

[64] TensorFlow, "Color Space Conversions," 2022. Available: https://www.tensorflow.org/io/tutorials/colorspace.

APPENDIX A

MODEL EVALUATION IMAGES

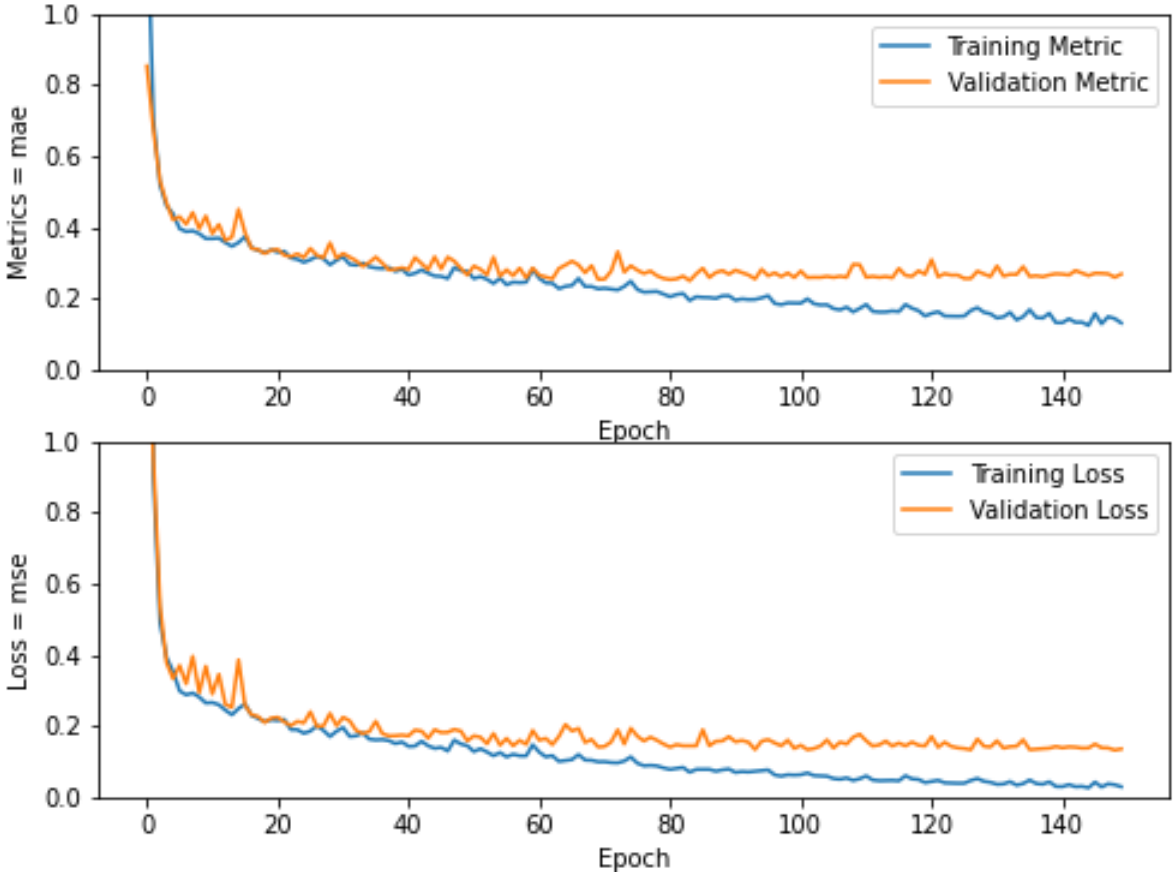


Figure A.1: Example CNN model trained to 150 epochs to demonstrate the impact of epochs on overfitting.

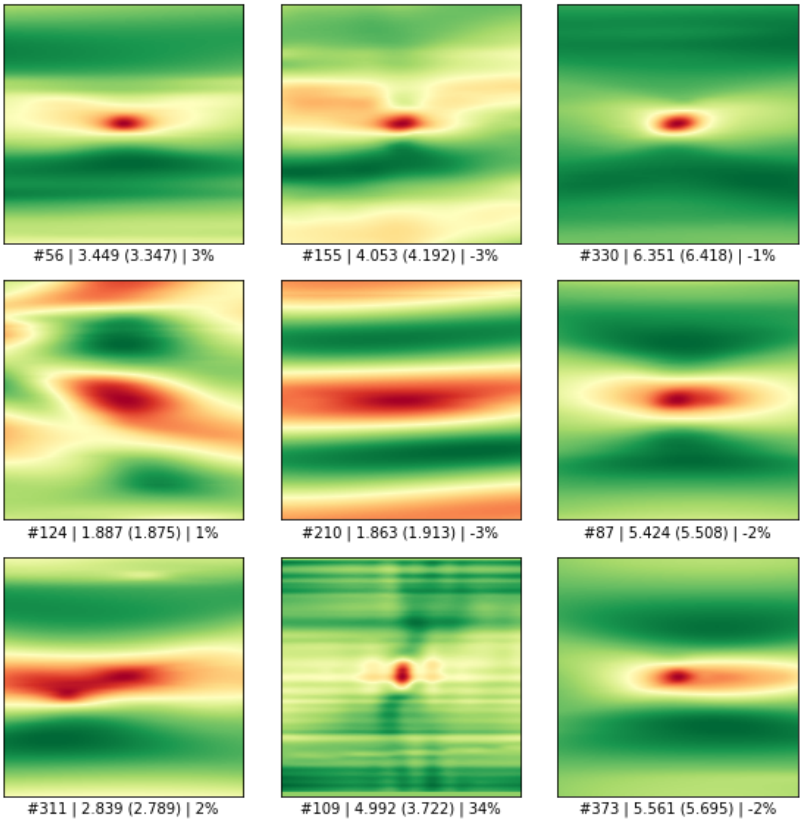


Figure A.2: Model 106: Prediction of SCF values from 9 random validation data points.

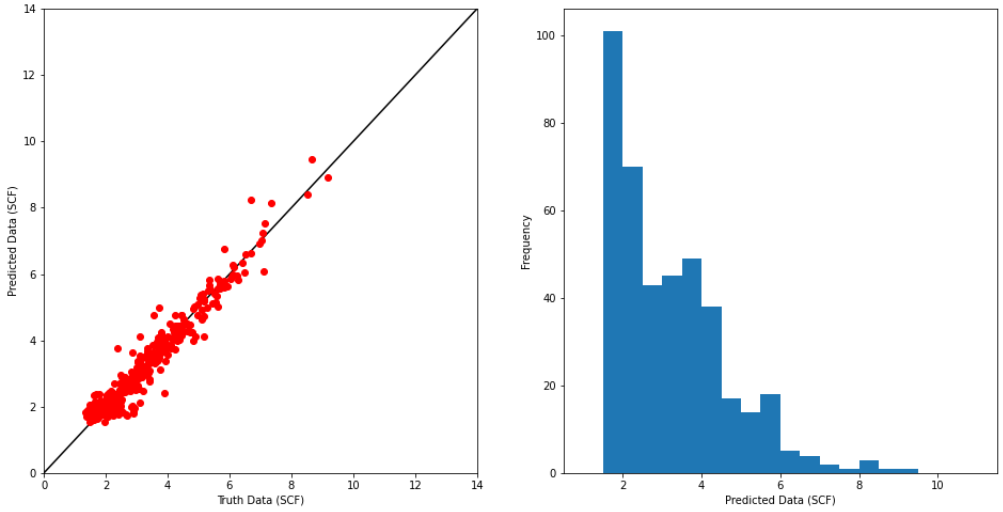


Figure A.3: Model 106: Unity plot and distribution of SCF predictions for validation data.

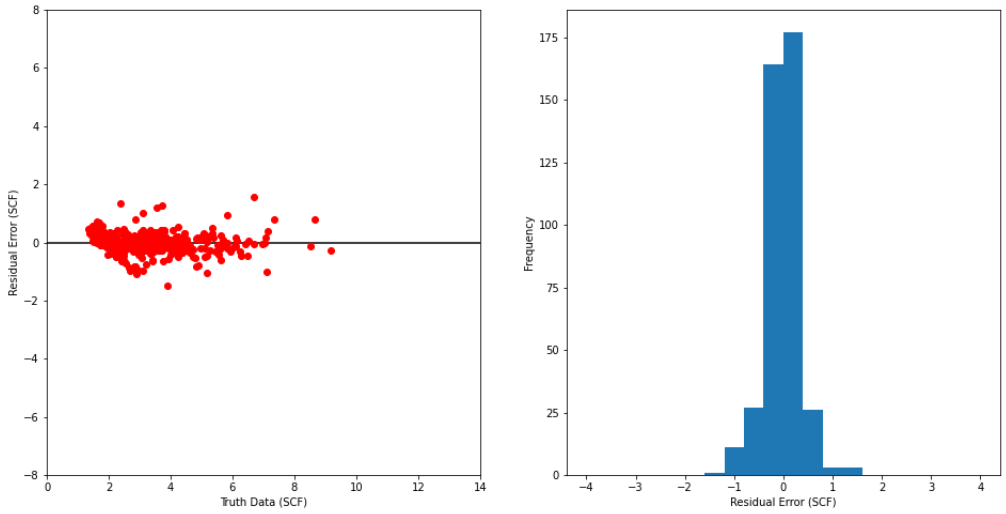


Figure A.4: Model 106: Residual plot and distribution of SCF predictions for validation data.

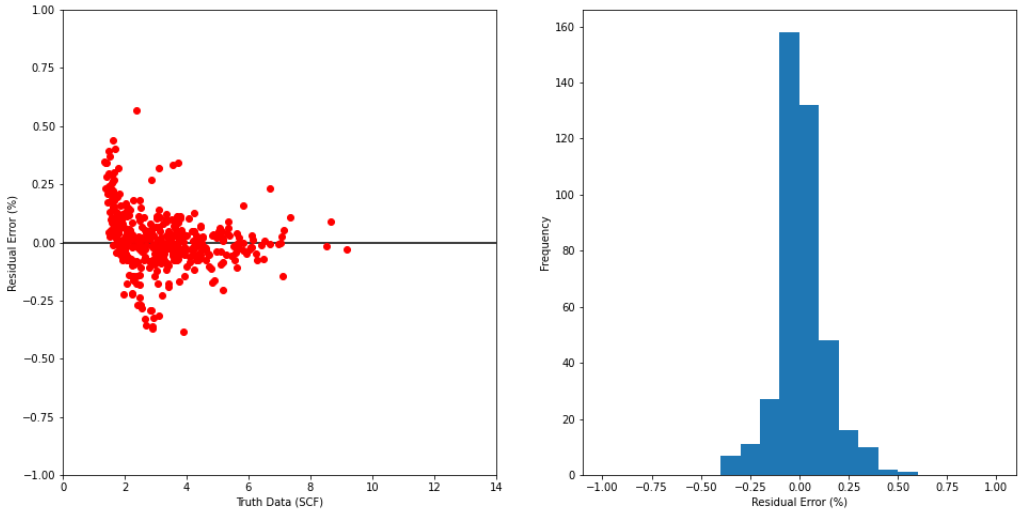


Figure A.5: Model 106: Percent residual plot and distribution of SCF predictions for validation data.

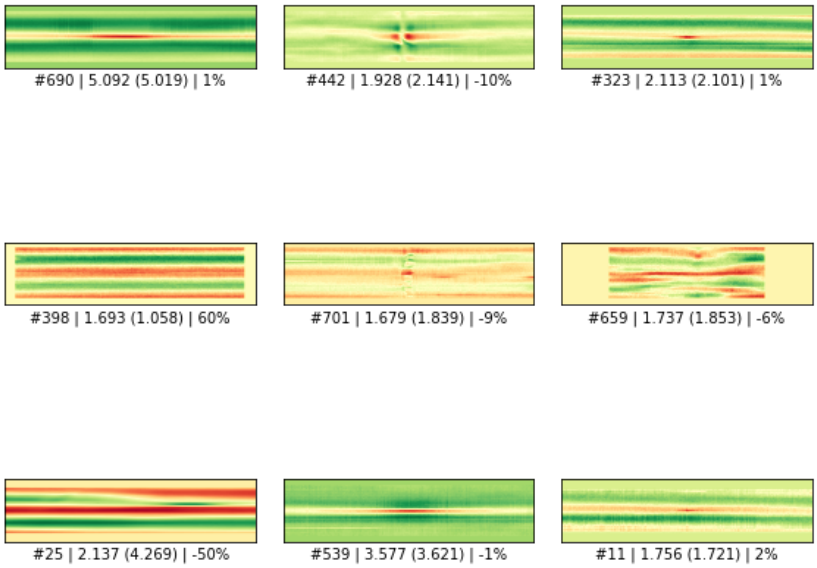


Figure A.6: Model 129: Prediction of SCF values from 9 random validation data points.

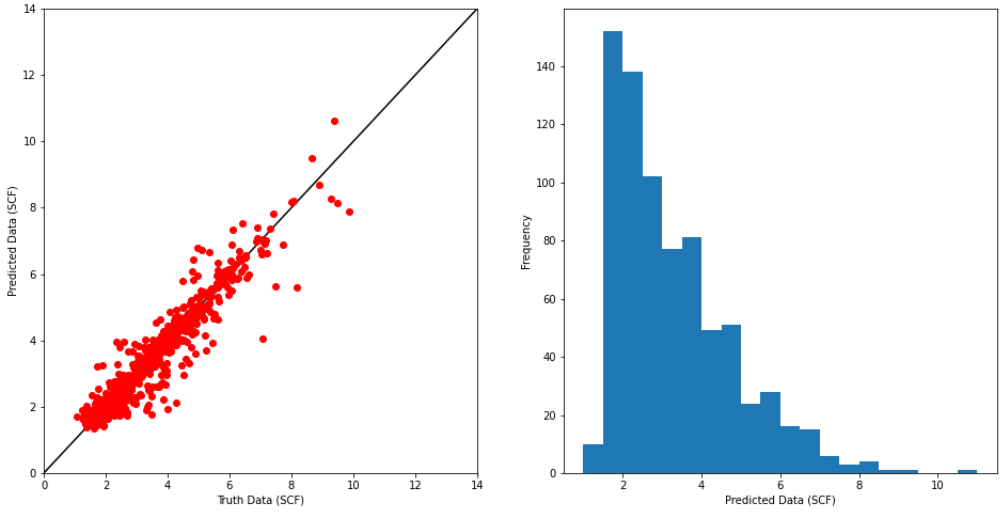


Figure A.7: Model 129: Unity plot and distribution of SCF predictions for validation data.

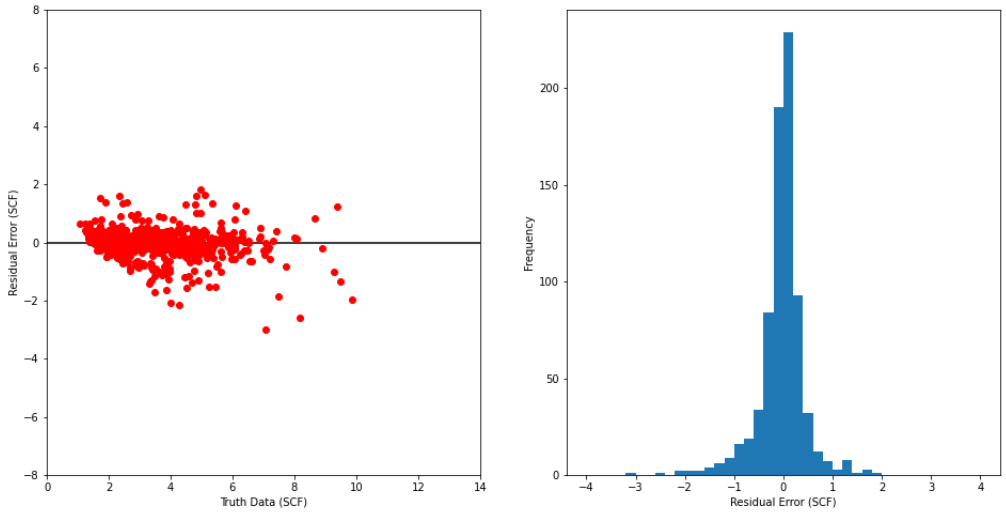


Figure A.8: Model 129: Residual plot and distribution of SCF predictions for validation data.

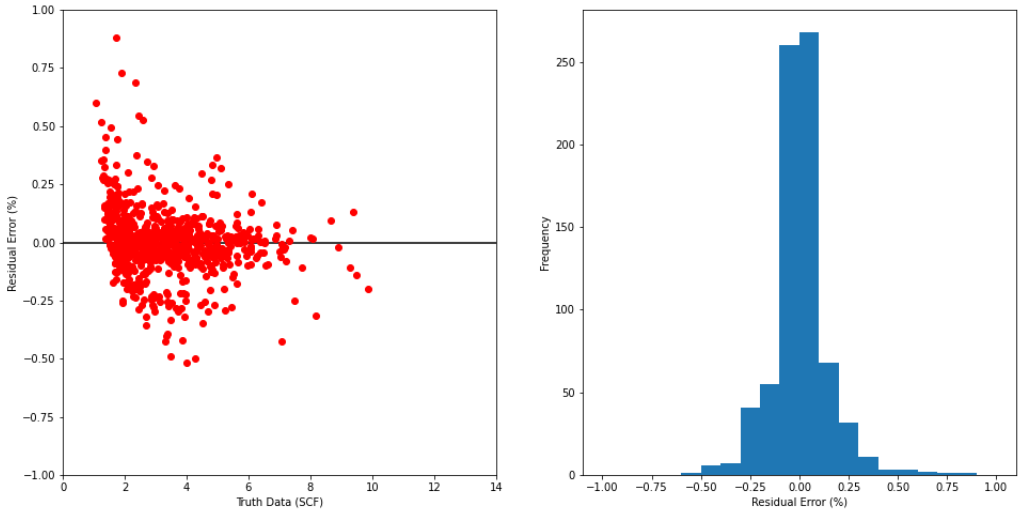


Figure A.9: Model 129: Percent residual plot and distribution of SCF predictions for validation data.

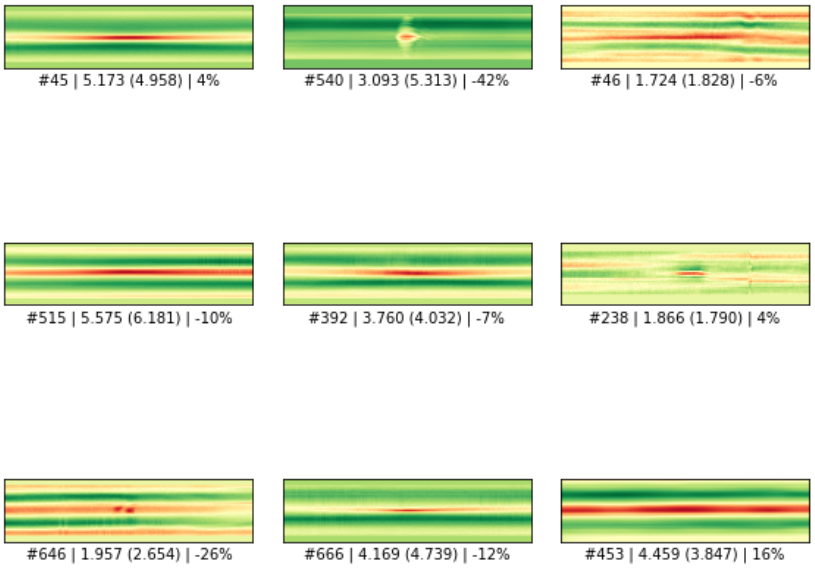


Figure A.10: Model 109: Prediction of SCF values from 9 random validation data points.

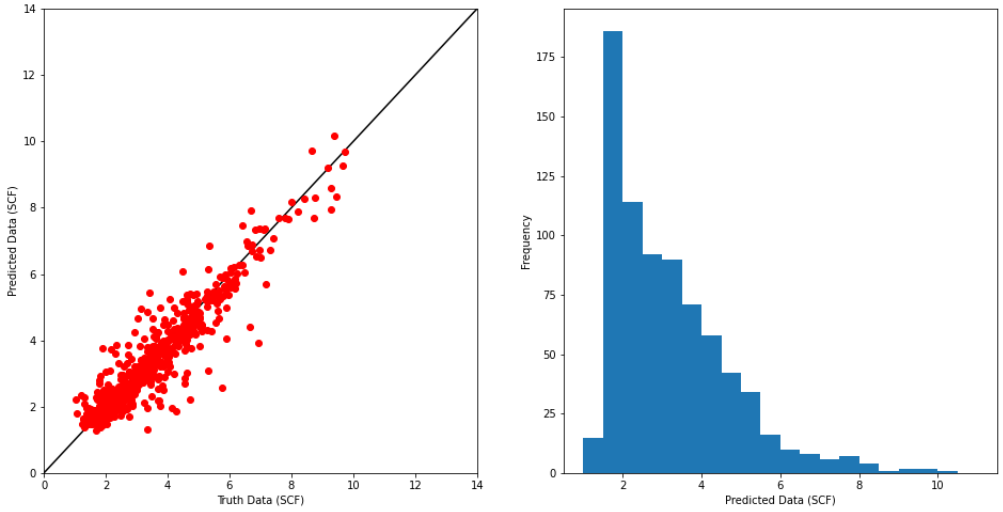


Figure A.11: Model 109: Unity plot and distribution of SCF predictions for validation data.

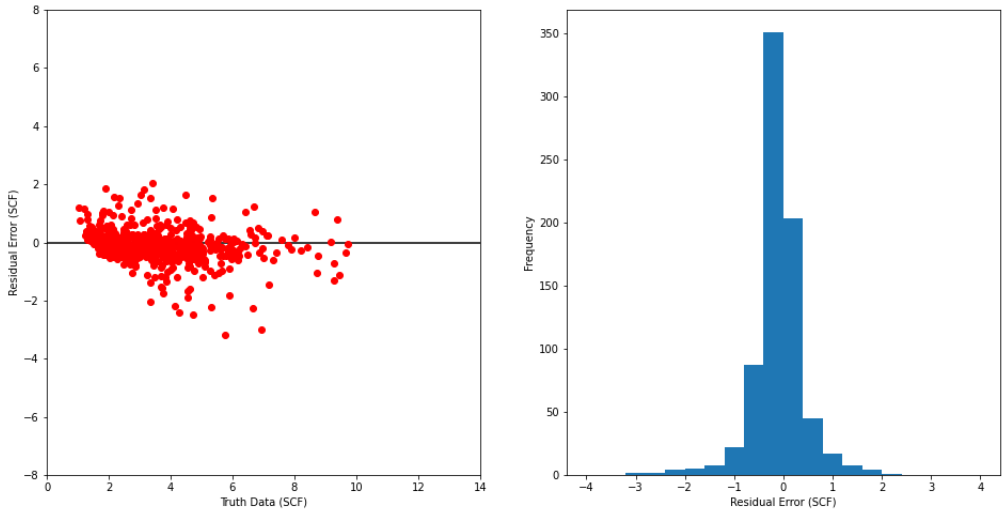


Figure A.12: Model 109: Residual plot and distribution of SCF predictions for validation data.

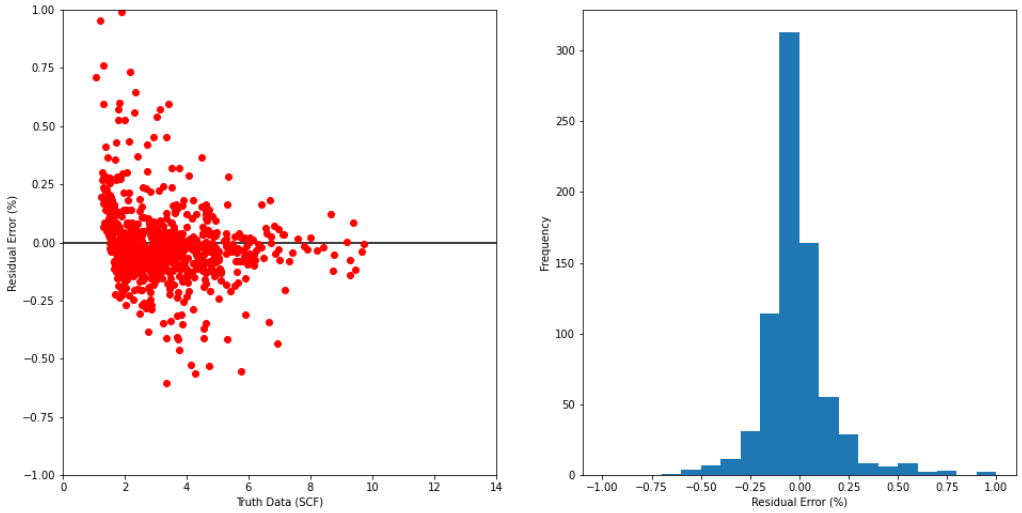


Figure A.13: Model 109: Percent residual plot and distribution of SCF predictions for validation data.

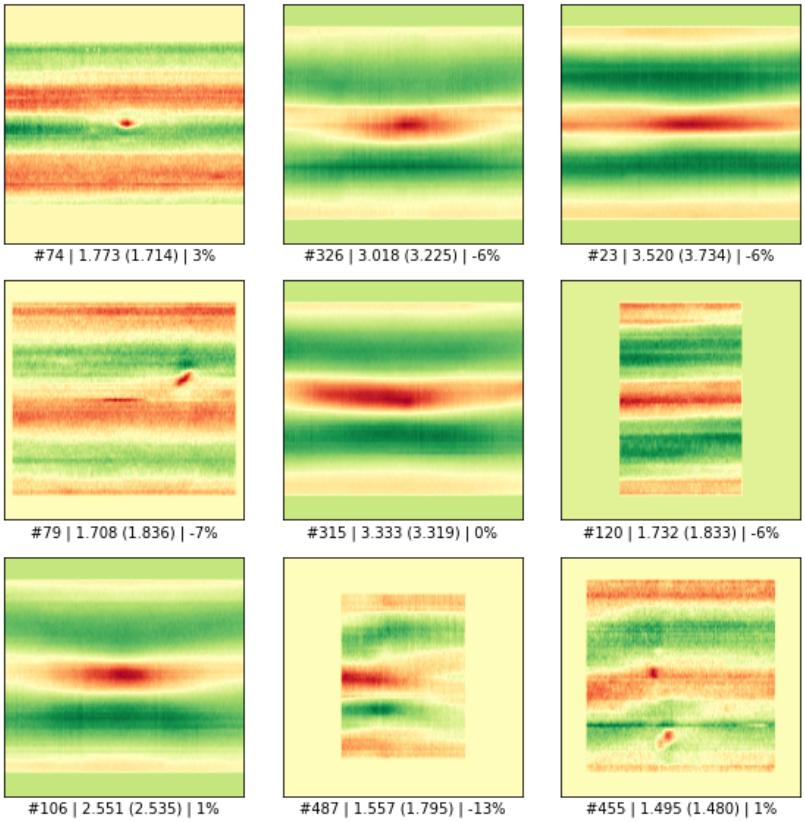


Figure A.14: Model 173: Prediction of SCF values from 9 random validation data points.

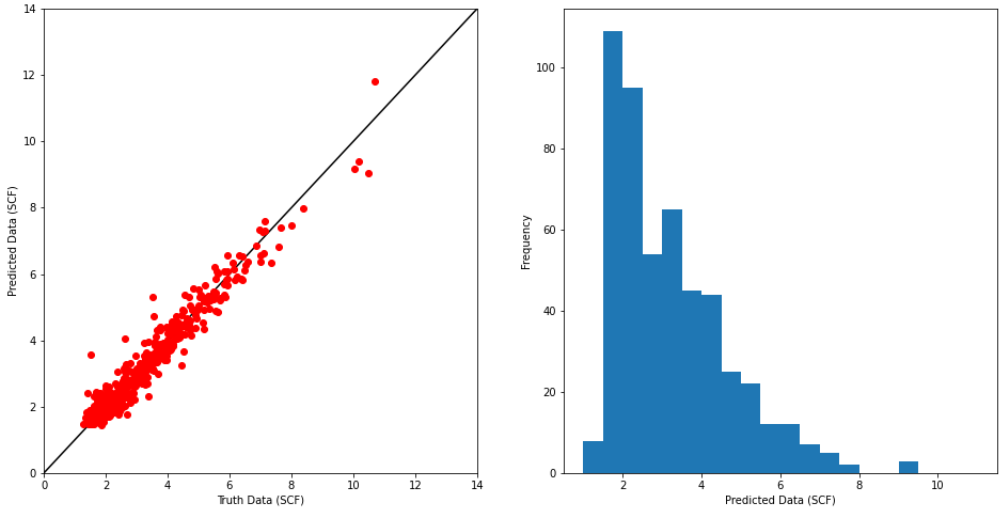


Figure A.15: Model 173: Unity plot and distribution of SCF predictions for validation data.

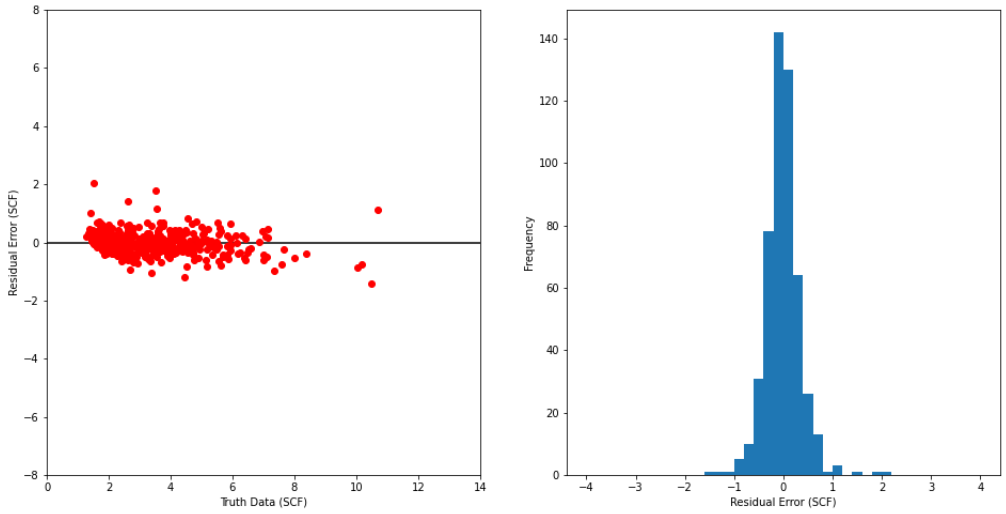


Figure A.16: Model 173: Residual plot and distribution of SCF predictions for validation data.

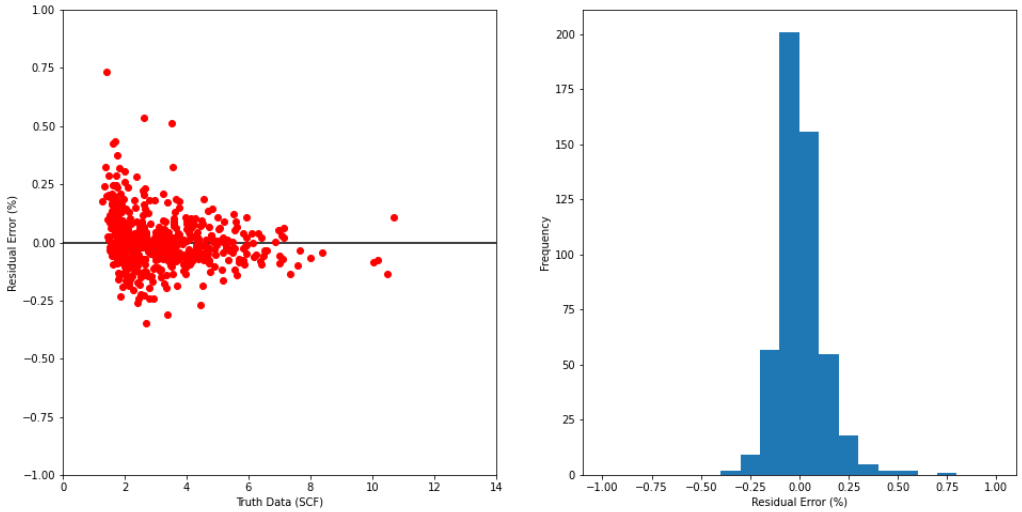


Figure A.17: Model 173: Percent residual plot and distribution of SCF predictions for validation data.

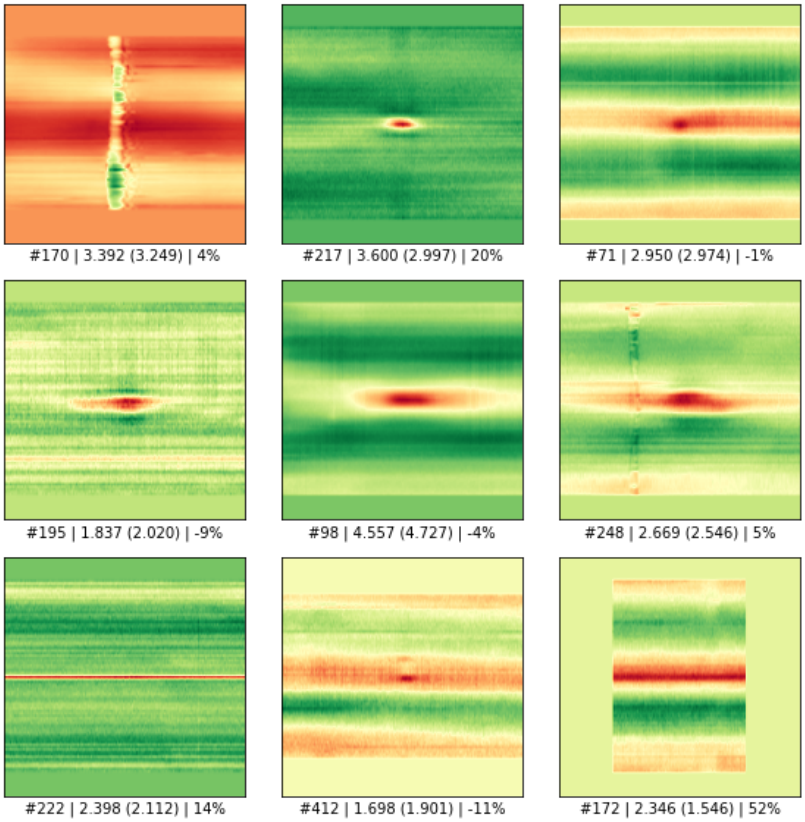


Figure A.18: Model 172: Prediction of SCF values from 9 random validation data points.

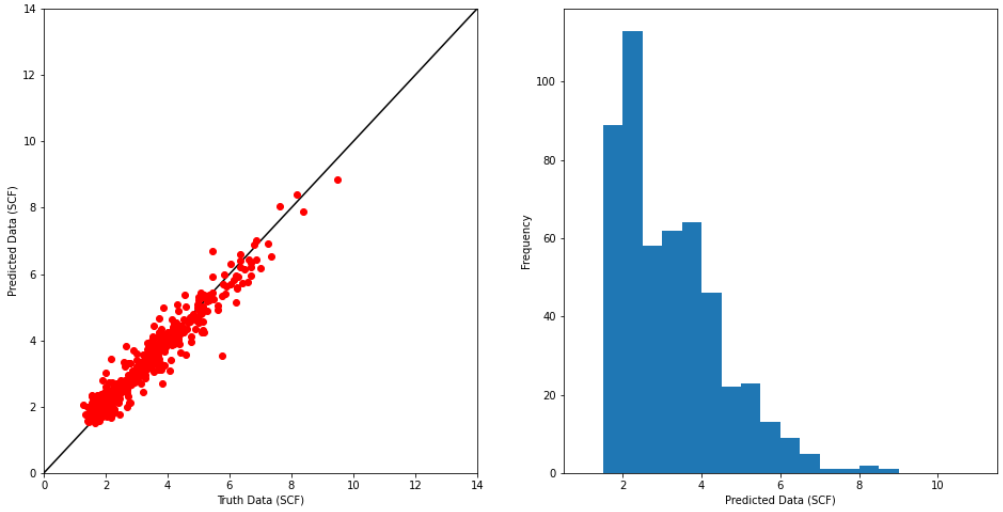


Figure A.19: Model 172: Unity plot and distribution of SCF predictions for validation data.

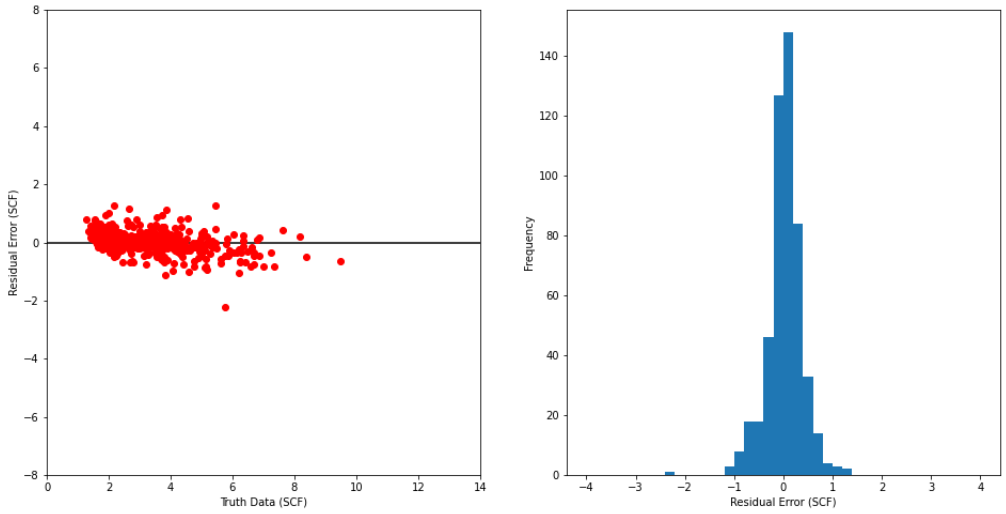


Figure A.20: Model 172: Residual plot and distribution of SCF predictions for validation data.

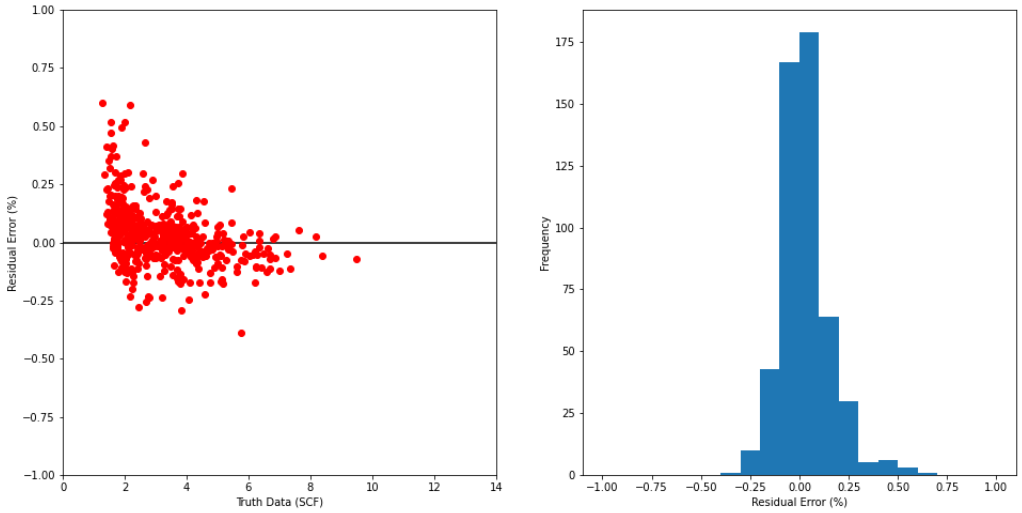


Figure A.21: Model 172: Percent residual plot and distribution of SCF predictions for validation data.

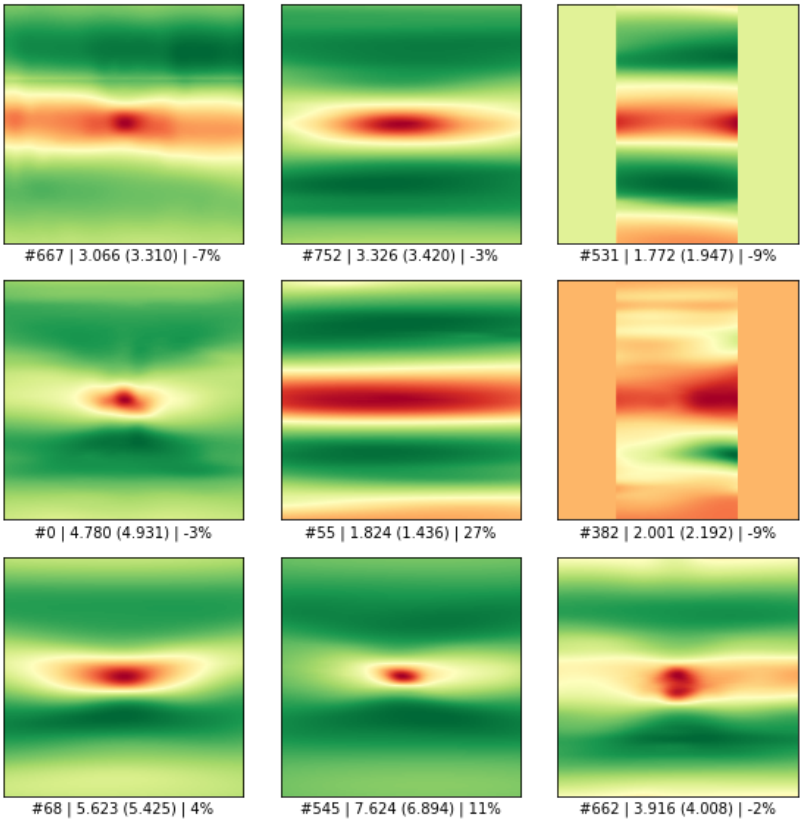


Figure A.22: Model 127: Prediction of SCF values from 9 random validation data points.

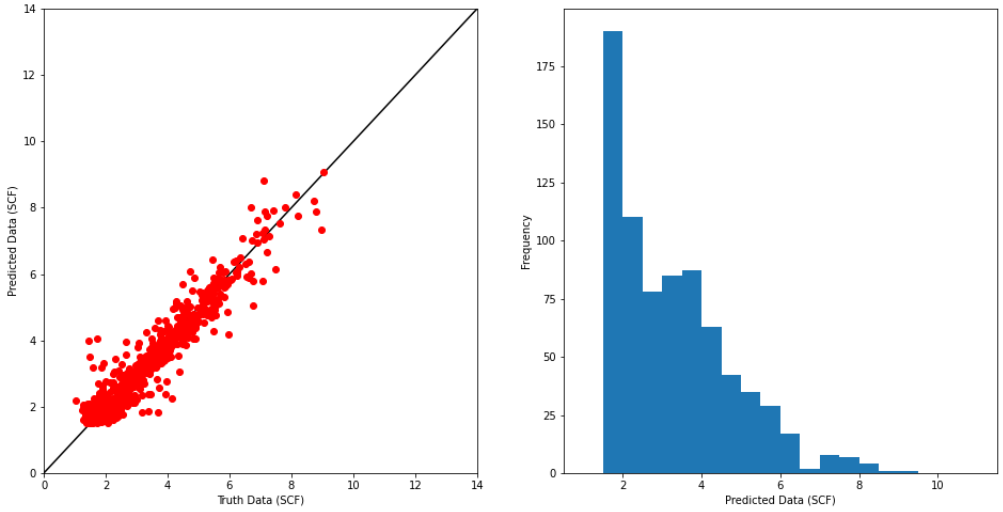


Figure A.23: Model 127: Unity plot and distribution of SCF predictions for validation data.

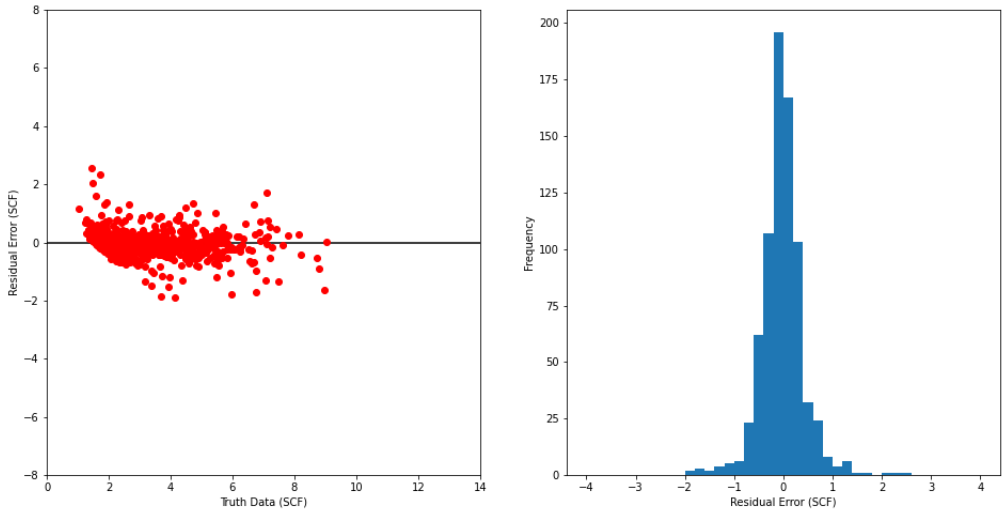


Figure A.24: Model 127: Residual plot and distribution of SCF predictions for validation data.

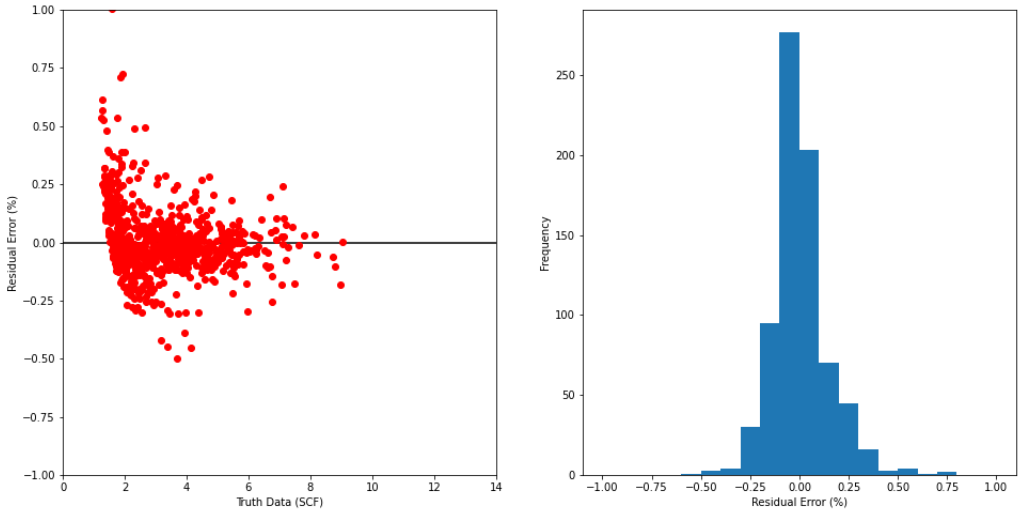


Figure A.25: Model 127: Percent residual plot and distribution of SCF predictions for validation data.

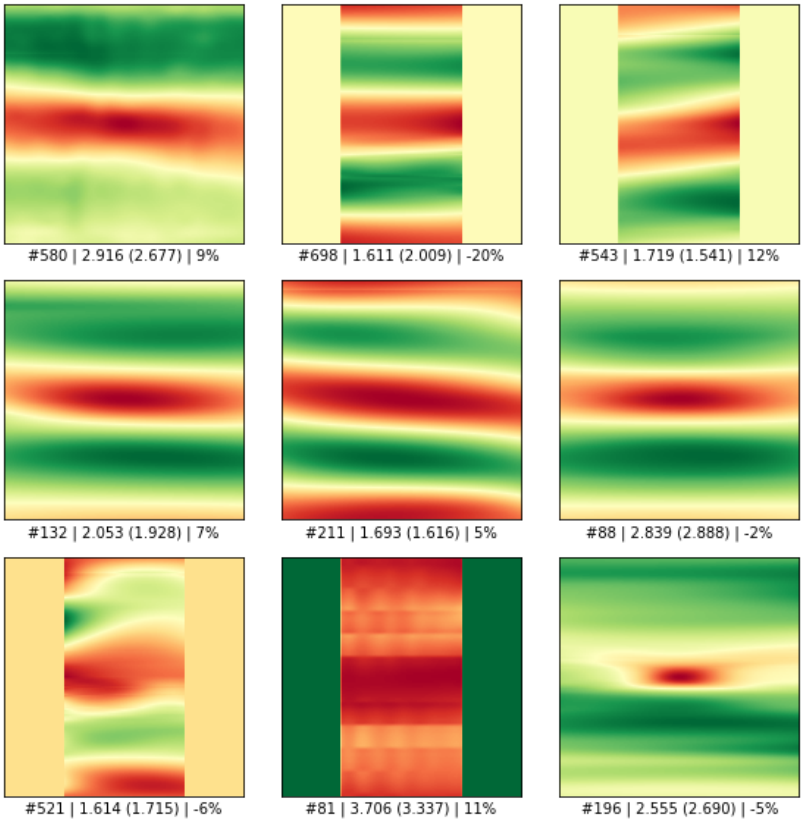


Figure A.26: Model 124: Prediction of SCF values from 9 random validation data points.

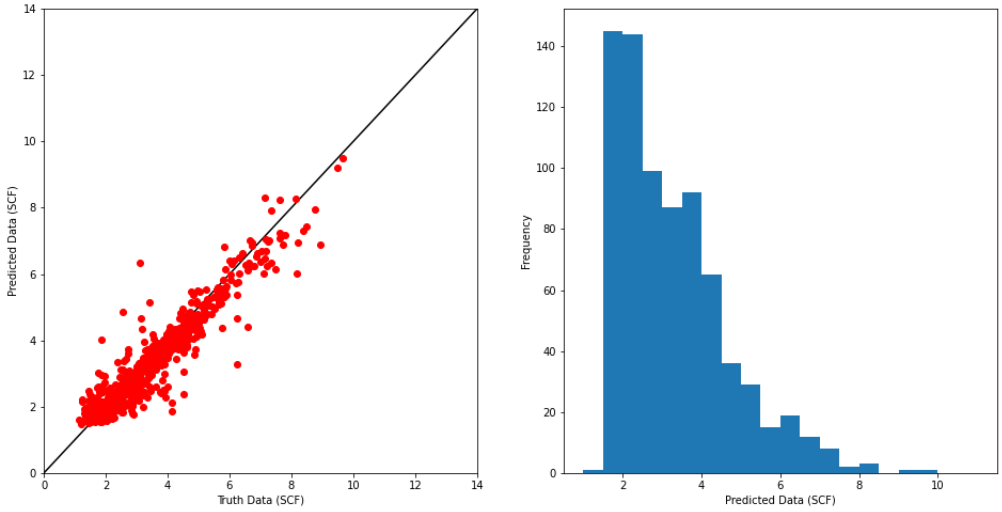


Figure A.27: Model 124: Unity plot and distribution of SCF predictions for validation data.

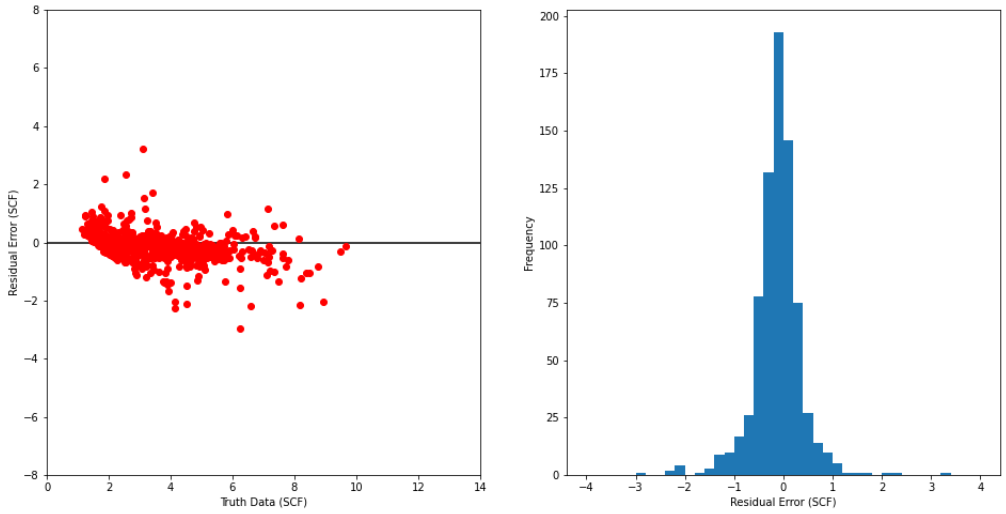


Figure A.28: Model 124: Residual plot and distribution of SCF predictions for validation data.

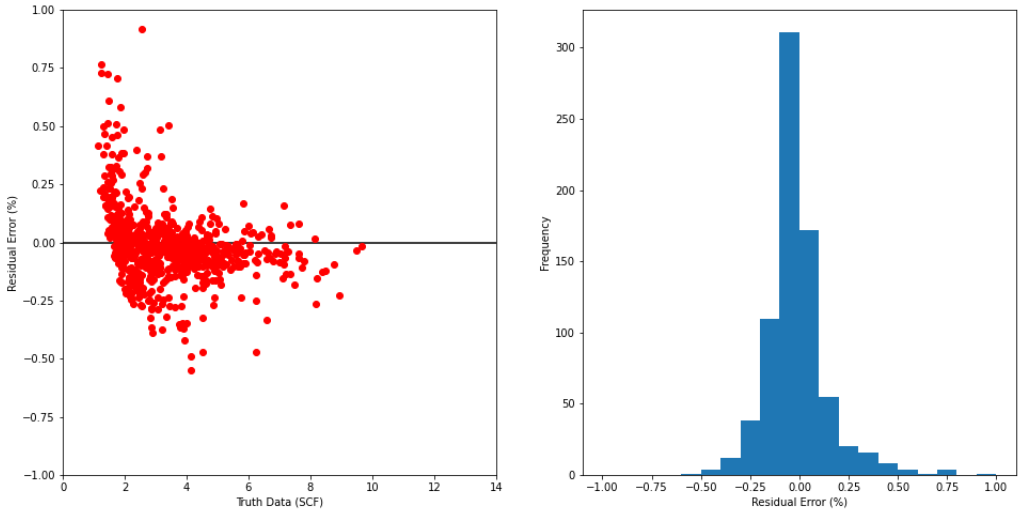


Figure A.29: Model 124: Percent residual plot and distribution of SCF predictions for validation data.

APPENDIX B

MODEL EVALUATION TABULAR DATA

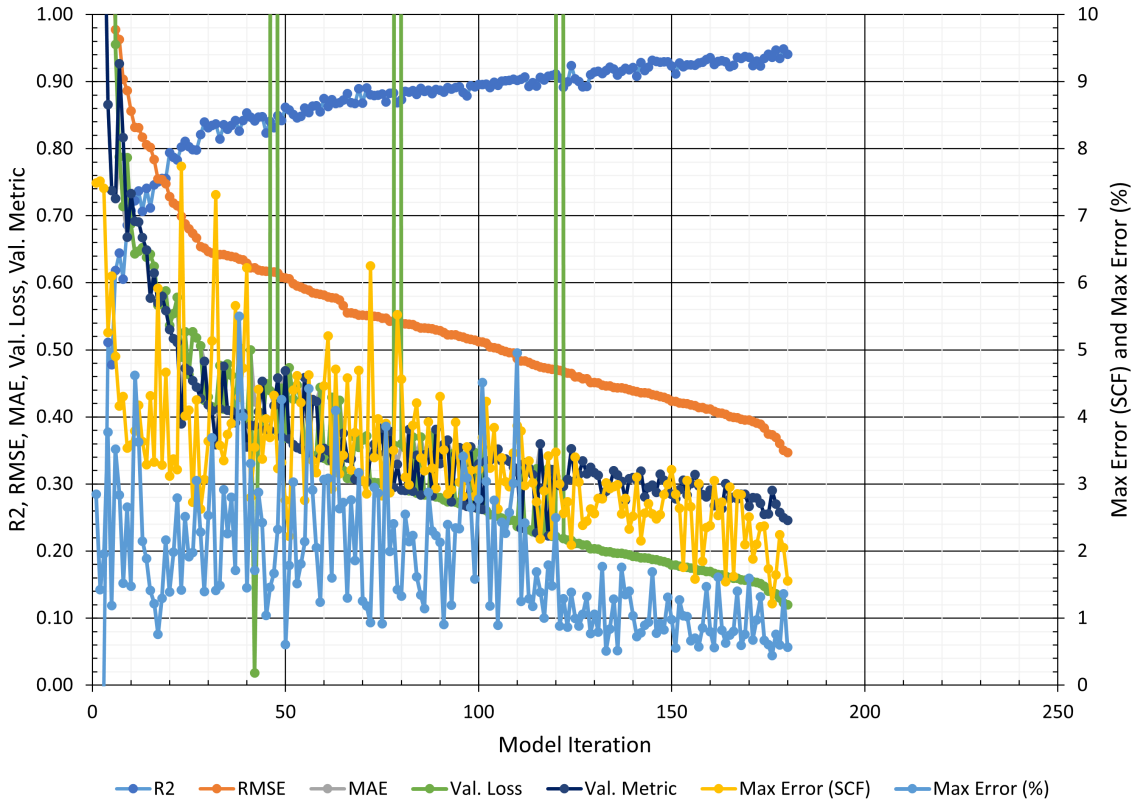


Figure B.1: Graphical representation of evaluation criteria for all model iterations.

	Model ID		Model Parameters								Model Training					Model Validation									
Run	Dataset	Conv2D Shapes	Conv2D Windows	Conv2D Activations	Dense Shapes	Dense Activations	Compile Optimizer	Compile Loss	Compile Metrics	Dataset Size	Training Size	Train Loss	Train Metrics	Training Time	Validation Size	Val. Loss	Val. Metrics	R2	MSE	RMSE	MAE	Max Error	Max Error Perc
1	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4683	3746	0	0	554	937	0.3700	0.2838	0.8902	0.2838	0.5327	0.3700	3.3885	1.3459
2	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	624	913	0.3697	0.3071	0.8672	0.3071	0.5542	0.3697	3.7684	1.8568
3	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.4732	0.3671	0.8579	0.3671	0.6059	0.4732	2.2204	1.7865
4	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	554	913	0.4290	0.3335	0.8732	0.3335	0.5775	0.4290	2.9271	1.5965
5	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.5540	0.5169	0.7870	0.5169	0.7190	0.5540	3.3740	1.9828
6	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.3467	0.2647	0.8923	0.2647	0.5145	0.3467	2.8196	1.5798
7	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.5365	0.5302	0.7936	0.5302	0.7282	0.5365	3.1186	1.3880
8	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.4226	0.3520	0.8480	0.3520	0.5933	0.4226	4.2123	1.8101
9	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	517	913	0.3229	0.2226	0.9082	0.2226	0.4718	0.3229	3.4169	1.7914
10	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	516	913	0.3595	0.2908	0.8729	0.2908	0.5393	0.3595	4.5652	1.3289
11	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	516	913	0.6875	0.7331	0.7074	0.7331	0.8562	0.6875	3.5807	1.4768
12	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	517	913	0.5269	0.4538	0.7985	0.4538	0.6737	0.5269	2.7267	1.9791
13	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	517	913	0.3444	0.2732	0.8889	0.2732	0.5226	0.3444	2.9158	1.1955
14	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	514	913	0.6519	0.6677	0.7065	0.6677	0.8171	0.6519	3.6290	2.1448
15	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	519	913	0.4474	0.3540	0.8456	0.3540	0.5950	0.4474	4.6115	1.5131
16	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.5061	0.4278	0.8211	0.4278	0.6540	0.5061	2.6254	2.2826
17	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	527	913	0.4245	0.3297	0.8682	0.3297	0.5742	0.4245	3.1510	2.6298
18	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	522	913	0.6246	0.6142	0.7458	0.6142	0.7837	0.6246	3.3255	1.2160
19	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.3699	0.2892	0.8863	0.2892	0.5378	0.3699	3.8724	2.2315
20	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	525	913	0.4494	0.3813	0.8234	0.3813	0.6175	0.4494	3.9688	1.0380
21	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	524	913	0.4300	0.4180	0.8315	0.4180	0.6465	0.4300	3.6385	2.5329
22	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	526	913	0.3226	0.2273	0.8933	0.2273	0.4768	0.3226	2.7332	1.6869
23	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	524	913	0.3549	0.3044	0.8892	0.3044	0.5518	0.3549	4.6905	3.1701
24	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	525	913	0.4997	0.3873	0.8470	0.3873	0.6223	0.4997	2.7954	3.3050
25	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	527	913	0.5267	0.4728	0.8108	0.4728	0.6876	0.5267	4.0136	2.5136
26	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	563	913	0.3637	0.2996	0.8693	0.2996	0.5474	0.3637	3.9035	3.8572
27	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.5029	0.4032	0.8260	0.4032	0.6350	0.5029	4.2933	5.4990
28	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.4768	0.4125	0.8142	0.4125	0.6422	0.4768	3.5743	1.4837
29	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	524	913	1.2307	2.6157	-0.0335	2.6157	1.6173	1.2307	7.5153	1.4214
30	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.4438	0.3396	0.8547	0.3396	0.5828	0.4438	3.5259	1.2400
31	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	524	913	0.3715	0.2825	0.8820	0.2825	0.5315	0.3715	3.2343	2.3018
32	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.3397	0.2668	0.8786	0.2668	0.5165	0.3397	3.5548	3.0906
33	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.7878	0.9267	0.6440	0.9267	0.9626	0.7878	4.1626	2.8373
34	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	531	913	0.3547	0.2620	0.8958	0.2620	0.5118	0.3547	3.6964	4.5140
35	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	529	913	0.3537	0.2905	0.8854	0.2905	0.5390	0.3537	3.1031	2.5466
36	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	530	913	0.3718	0.3035	0.8903	0.3035	0.5509	0.3718	2.8535	1.1826
37	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.6470	0.6904	0.7367	0.6904	0.8309	0.6470	4.1720	3.6272
38	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	529	913	0.5176	0.4445	0.7981	0.4445	0.6667	0.5176	4.2506	3.0519
39	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	531	913	0.3415	0.2692	0.8838	0.2692	0.5188	0.3415	2.8214	3.4114
40	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	528	913	0.5785	0.5109	0.7830	0.5109	0.7148	0.5785	3.2113	2.7920
41	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.4792	0.4102	0.8291	0.4102	0.6405	0.4792	3.7392	2.2610
42	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.6434	0.6917	0.7222	0.6917	0.8317	0.6434	3.7899	4.6197
43	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.6380	0.6491	0.7409	0.6491	0.8057	0.6380	3.2854	1.8853
44	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.3579	0.2947	0.8830	0.2947	0.5429	0.3579	3.4996	2.4078
45	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	525	913	0.4396	0.3802	0.8407	0.3802	0.6166	0.4396	3.6917	1.4567

Figure B.2: Hyperparameter settings, resultant training and validation for all 180 CNN model iterations (1 of 3).

	Model ID		Model Parameters								Model Training					Model Validation								
	Run	Dataset	Conv2D Shapes	Conv2D Windows	Conv2D Activations	Dense Shapes	Dense Activations	Compile Optimizer	Compile Loss	Compile Metrics	Dataset Size	Training Size	Train Loss	Train Metrics	Training Time	Validation Size	Val. Loss	Val. Metrics	R2	MSE	RMSE	MAE	Max Error	Max Error Perc
48	46	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	526	913	0.4430	0.3702	0.8421	0.3702	0.6084	0.4430	3.3237	4.2572
49	47	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	525	913	0.4264	0.3585	0.8510	0.3585	0.5988	0.4264	4.3998	3.0271
50	48	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.5878	0.5587	0.7556	0.5587	0.7475	0.5878	4.6651	2.1641
51	49	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	525	913	0.7139	0.8165	0.6054	0.8165	0.9036	0.7139	4.3059	1.5214
52	50	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	524	913	0.4619	0.4089	0.8348	0.4089	0.6395	0.4619	3.9035	2.8027
53	51	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4562	3649	0	0	564	913	0.3237	0.2339	0.9037	0.2339	0.4837	0.3237	3.7920	1.2505
54	52	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4683	3746	0	0	525	937	0.5666	0.5696	0.7502	0.5696	0.7547	0.5666	5.9189	0.7571
55	53	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4683	3746	0	0	525	937	1.2524	2.7031	-0.0645	2.7031	1.6441	1.2524	7.4887	2.8454
56	54	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mae	mse	4683	3746	0	0	550	937	0.3656	0.3042	0.8679	0.3042	0.5516	0.3656	3.0519	1.2522
57	55	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	581	937	0.2034	0.3167	0.9139	0.2034	0.4510	0.3167	2.5581	1.0558
58	56	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	624	937	0.2475	0.3370	0.9012	0.2475	0.4975	0.3370	3.0327	2.2681
59	57	Smooth & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3,3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	571	937	0.3816	0.4527	0.8477	0.3816	0.6177	0.4527	3.3720	2.4211
60	58	Smooth & All	[32, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	323	937	0.3018	0.3456	0.8795	0.3018	0.5494	0.3456	3.9709	2.8636
61	59	Smooth & All	[32, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	280	937	0.2338	0.3315	0.9067	0.2338	0.4835	0.3315	2.9774	2.4170
62	60	Smooth & All	[32, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	280	937	0.3424	0.4258	0.8633	0.3424	0.5852	0.4258	3.3466	2.9096
63	61	Smooth & All	[32, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	564	937	0.3491	0.4584	0.8607	0.3491	0.5908	0.4584	2.7584	2.1388
64	62	Smooth & All	[32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	498	937	0.3828	0.4078	0.8472	0.3828	0.6187	0.4078	4.4088	2.8774
65	63	Smooth & All	[32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	503	937	0.3023	0.3592	0.8794	0.3023	0.5498	0.3592	3.3933	2.9502
66	64	Smooth & All	[32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	504	937	0.3200	0.3755	0.8723	0.3200	0.5657	0.3755	3.4303	2.7210
67	65	Smooth & All	[32, 64, 128]	[(4, 4), (4, 4), (4, 4)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	744	937	0.3776	0.4580	0.8493	0.3776	0.6145	0.4580	3.2328	2.3212
68	66	Smooth & All	[32, 64, 128]	[(4, 4), (4, 4), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	684	937	0.2535	0.3431	0.8988	0.2535	0.5034	0.3431	3.8381	2.7557
69	67	Smooth & All	[32, 64, 128]	[(4, 4), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	525	937	0.2659	0.3552	0.8939	0.2659	0.5157	0.3552	3.2025	2.6479
70	68	Smooth & All	[32, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[50, 10, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	564	937	0.2729	0.3440	0.8911	0.2729	0.5224	0.3440	3.9194	2.3458
71	69	Smooth & All	[32, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	564	937	0.2218	0.3109	0.9115	0.2218	0.4709	0.3109	3.4752	2.4952
72	70	Smooth & All	[32, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	564	937	0.4119	0.4757	0.8356	0.4119	0.6418	0.4757	3.3507	2.9131
73	71	Smooth & All	[32, 64, 128]	[(6, 6), (6, 6), (6, 6)]	['relu', 'relu', 'relu', 'relu']	[100, 50, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	744	937	0.3412	0.4229	0.8638	0.3412	0.5841	0.4229	3.1708	2.0525
74	72	Smooth & All	[32, 64, 128]	[(12, 12), (6, 6), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[100, 50, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	744	937	0.2732	0.3653	0.8910	0.2732	0.5226	0.3653	2.8515	2.3955
75	73	Smooth & All	[32, 64, 128]	[(12, 12), (6, 6), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	744	937	0.2899	0.3813	0.8843	0.2899	0.5384	0.3813	2.9852	2.1434
76	74	Smooth & All	[32, 64, 128]	[(12, 12), (6, 6), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	744	937	0.9556	0.7257	0.6186	0.9556	0.9776	0.7257	4.9010	3.5189
77	75	Smooth & All	[32, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	564	937	1.2256	0.8658	0.5109	1.2256	1.1071	0.8658	5.2553	3.7733
78	76	Smooth & All	[32, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	564	937	0.4148	0.4093	0.8344	0.4148	0.6441	0.4093	5.1332	3.6856
79	77	Smooth & All	[64, 64, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	4683	3746	0	0	678	937	0.2948	0.3784	0.8823	0.2948	0.5430	0.3784	2.8682	1.9956
80	78	Smooth & All	[128, 64, 32]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	4683	3746	0	0	1104	937	0.2623	0.3300	0.8953	0.2623	0.5121	0.3300	3.2132	2.7753
81	79	Smooth & All	[128, 64]	[(3, 3), (3, 3)]	['relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	4683	3746	0	0	1103	937	0.2828	0.3468	0.8872	0.2828	0.5317	0.3468	3.9245	2.8775
82	80	Smooth & All	[128, 64]	[(3, 3), (3, 3)]	['relu', 'relu']	[120, 60, 1]	['relu', 'relu']	RMSprop	mse	mae	4683	3746	0	0	1223	937	0.7864	0.6679	0.6862	0.7864	0.8868	0.6679	3.5382	2.6539
83	81	Smooth & All	[64, 128]	[(3, 3), (3, 3)]	['relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	4683	3746	0	0	983	937	0.2706	0.3542	0.8920	0.2706	0.5202	0.3542	3.0299	2.3341
84	82	Smooth & All	[128, 128, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	4683	3746	0	0	1463	937	0.2602	0.3341	0.8962	0.2602	0.5101	0.3341	4.2308	3.0377
85	83	Smooth & All	[128, 64, 32]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	4683	3746	0	0	1132	937	0.2457	0.3336	0.9019	0.2457	0.4957	0.3336	2.9606	2.5740
86	84	Smooth & All	[32, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	4683	3746	0	0	564	937	0.2493	0.3346	0.9005	0.2493	0.4993	0.3346	3.3867	2.4316
87	85	Smooth & All	[32, 64, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	4683	3746	0	0	485	937	0.2807	0.3811	0.8880	0.2807	0.5298	0.3811	2.9317	2.2326
88	86	Smooth & All	[64, 64, 32]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.2071	0.2969	624	759	0.2794	0.3307	0.8872	0.2794	0.5285	0.3307	4.3013	2.1311
89	87	Smooth & Pristine	[64, 64, 32]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1039	0.2341	383	412	0.1629	0.3004	0.9296	0.1629	0.4036	0.3004	1.5426	0.6258
90	88	Smooth & Pristine	[64, 64, 64, 64]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1097	0.2497	383	412	0.2159	0.3522	0.9240	0.2159	0.4646	0.3522	2.0892	1.3837
91	89	Smooth & Pristine	[64, 64, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	RMSprop	mse	mae	2541	1874	0.1493	0.2878	338	412	0.3685	0.4686	0.8617	0.3685	0.6070	0.4686	2.7571	0.6036
92	90	Smooth & Pristine	[64, 64, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.0756	0.2066	383	412	0.1695	0.2807	0.9327	0.1695	0.4117	0.2807	2.3473	1.4670

Figure B.3: Hyperparameter settings, resultant training and validation for all 180 CNN model iterations (2 of 5).

	Model ID		Model Parameters								Model Training					Model Validation								
	Run	Dataset	Conv2D Shapes	Conv2D Windows	Conv2D Activations	Dense Shapes	Dense Activations	Compile Optimizer	Compile Loss	Compile Metrics	Dataset Size	Training Size	Train Loss	Train Metrics	Training Time	Validation Size	Val. Loss	Val. Metrics	R2	MSE	RMSE	MAE	Max Error	Max Error Perc
93	91	Smooth & Pristine	[64, 64, 16]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1126	0.2490	323	412	0.2037	0.3250	0.9100	0.2037	0.4513	0.3250	2.6255	0.7700
94	92	Smooth & Pristine	[64, 32, 32]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1055	0.2429	277	412	0.1759	0.2834	0.9250	0.1759	0.4194	0.2834	3.0510	1.0256
95	93	Smooth & Pristine	[32, 32, 32]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.0937	0.2255	203	412	0.1571	0.2845	0.9342	0.1571	0.3963	0.2845	2.8424	0.5940
96	94	Smooth & Pristine	[32, 32, 32]	[(6, 6), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1137	0.2545	181	412	0.1712	0.2973	0.9276	0.1712	0.4137	0.2973	3.0030	0.5695
97	95	Smooth & Pristine	[32, 32, 32]	[(12, 12), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.0852	0.2192	263	412	0.1772	0.2913	0.9280	0.1772	0.4210	0.2913	2.6407	1.2745
98	96	Smooth & Pristine	[16, 32, 16]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1149	0.2526	143	412	0.1924	0.3097	0.9212	0.1924	0.4386	0.3097	2.5219	1.0323
99	97	Smooth & Pristine	[16, 16, 16]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1051	0.2450	143	412	0.1841	0.3057	0.9292	0.1841	0.4291	0.3057	2.8390	0.8234
100	98	Smooth & Pristine	[16, 16, 8]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1274	0.2678	112	412	0.1748	0.3005	0.9246	0.1748	0.4181	0.3005	2.6585	0.6638
101	99	Smooth & Pristine	[16, 8, 8]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1231	0.2590	103	412	0.2029	0.3129	0.9155	0.2029	0.4504	0.3129	2.7832	0.7913
102	100	Smooth & Pristine	[8, 8, 8]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[120, 60, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1260	0.2602	144	412	0.1914	0.3056	0.9078	0.1914	0.4374	0.3056	3.1002	0.7221
103	101	Smooth & Pristine	[8, 8, 8]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[60, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1422	0.2797	82	412	0.1792	0.2844	0.9233	0.1792	0.4234	0.2844	3.2166	0.9756
104	102	Smooth & Pristine	[8, 8, 8]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[20, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1524	0.2916	80	412	0.1970	0.3199	0.9185	0.1970	0.4438	0.3199	2.9601	1.2813
105	103	Smooth & Pristine	[32, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1048	0.2404	261	412	0.1529	0.2789	0.9303	0.1529	0.3910	0.2789	2.0995	0.9720
106	104	Smooth & Pristine	[64, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.0999	0.2374	383	412	0.1942	0.3080	0.9175	0.1942	0.4407	0.3080	2.3281	1.4034
107	105	Smooth & Pristine	[128, 64, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1009	0.2392	623	412	0.1968	0.3135	0.9096	0.1968	0.4436	0.3135	2.9865	0.5173
108	106	Smooth & Pristine	[128, 128, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.0970	0.2306	719	412	0.1201	0.2459	0.9404	0.1201	0.3465	0.2459	1.5518	0.5686
109	107	Smooth & Pristine	[128, 128, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1099	0.2460	364	412	0.2249	0.3596	0.9060	0.2249	0.4742	0.3596	2.1793	1.3859
110	108	Raw & All	[128, 128, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.1611	0.2696	1343	759	0.3351	0.3499	0.8628	0.3351	0.5789	0.3499	5.2081	3.0862
111	109	Raw & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.2059	0.3188	564	759	0.2834	0.3436	0.8855	0.2834	0.5323	0.3436	3.1959	1.1415
112	110	Raw & All	[64, 64, 128, 128]	[(5, 5), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.1624	0.2712	684	759	0.3030	0.3400	0.8801	0.3030	0.5504	0.3400	6.2490	0.9318
113	111	Raw & All	[32, 64]	[(3, 3), (3, 3)]	['relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.1458	0.2625	444	759	0.4888	0.3900	0.8025	0.4888	0.6992	0.3900	7.7371	1.4196
114	112	Raw & All	[32, 64]	[(3, 3), (3, 3)]	['relu', 'relu']	[50, 10, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.1660	0.2742	443	759	0.3466	0.3450	0.8536	0.3466	0.5888	0.3450	4.6241	4.4245
115	113	Raw & Pristine	[32, 64]	[(3, 3), (3, 3)]	['relu', 'relu']	[50, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.0909	0.2253	233	412	0.1990	0.2976	0.9214	0.1990	0.4460	0.2976	2.9315	0.8358
116	114	Raw & Pristine	[16, 32]	[(3, 3), (3, 3)]	['relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.0987	0.2287	143	412	0.1512	0.2710	0.9231	0.1512	0.3889	0.2710	2.3627	1.3147
117	115	Raw & Pristine	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1178	0.2427	143	412	0.1473	0.2546	0.9348	0.1473	0.3837	0.2546	2.3718	0.6649
118	116	Raw & Pristine	[16, 16, 32]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1140	0.2478	143	412	0.1694	0.2909	0.9354	0.1694	0.4116	0.2909	2.3781	0.7958
119	117	Raw & All	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.1693	0.2816	263	759	0.3082	0.3372	0.8682	0.3082	0.5551	0.3372	3.2060	2.7595
120	118	Raw & All	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.2072	0.3102	324	759	0.3331	0.3649	0.8671	0.3331	0.5771	0.3649	4.7104	4.0954
121	119	Raw & All	[16, 16, 16, 16]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.2451	0.3354	207	759	0.4072	0.3917	0.8421	0.4072	0.6381	0.3917	5.6600	1.7109
122	120	Raw & All	[32, 32, 32, 32]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.2161	0.3073	338	759	0.3082	0.3389	0.8820	0.3082	0.5552	0.3389	4.5798	1.2990
123	121	Raw & All	[32, 32, 32, 32]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.2185	0.3088	384	759	0.4019	0.4054	0.8420	0.4019	0.6340	0.4054	4.7262	4.0024
124	122	Raw & All	[32, 64, 128, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[512, 1]	['relu']	adam	mse	mae	4683	3455	0.3829	0.4144	504	759	0.2863	0.3430	0.8810	0.2863	0.5351	0.3430	4.2091	1.6147
125	123	Raw & All	[32, 32, 32, 32]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.1891	0.2924	383	759	0.3385	0.3532	0.8745	0.3385	0.5818	0.3532	4.4594	3.0671
126	124	Smooth & All	[32, 32, 32, 32]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.1629	0.2873	384	759	0.2369	0.3180	0.9018	0.2369	0.4867	0.3180	3.8685	4.9549
127	125	Smooth & All	[16, 32, 64, 64]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.1666	0.2922	274	759	0.2541	0.3357	0.8908	0.2541	0.5041	0.3357	3.2343	1.1799
128	126	Smooth & All	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.1690	0.2721	323	759	0.2455	0.3246	0.9036	0.2455	0.4955	0.3246	3.4627	3.0105
129	127	Smooth & All	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.1506	0.2724	286	759	0.1967	0.3037	0.9152	0.1967	0.4435	0.3037	2.5509	1.7551

Figure B.4: Hyperparameter settings, resultant training and validation for all 180 CNN model iterations (3 of 5).

	Model ID		Model Parameters								Model Training					Model Validation								
	Run	Dataset	Conv2D Shapes	Conv2D Windows	Conv2D Activations	Dense Shapes	Dense Activations	Compile Optimizer	Compile Loss	Compile Metrics	Dataset Size	Training Size	Train Loss	Train Metrics	Training Time	Validation Size	Val. Loss	Val. Metrics	R2	MSE	RMSE	MAE	Max Error	Max Error Perc
131	129	Raw & All	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mean_absolute_percentage_error	mae	4683	3455	8.9676	0.2898	139	759	9.5561	0.2976	0.9052	0.2203	0.4693	0.2976	3.0001	0.8829
132	130	Raw & All	[32, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mean_absolute_percentage_error	mae	4683	3455	8.2996	0.2673	203	759	10.7800	0.3292	0.8683	0.2912	0.5397	0.3292	5.5245	1.4237
133	131	Raw & All	[32, 64, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mean_absolute_percentage_error	mae	4683	3455	8.9330	0.2937	448	759	12.9937	0.4169	0.8312	0.3799	0.6163	0.4169	4.3203	1.6667
134	132	Raw & All	[11, 64, 64, 128]	[(4, 4), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mean_absolute_percentage_error	mae	4683	3455	17.8627	0.6518	564	759	19.5493	0.7374	0.4776	1.1968	1.0940	0.7374	6.0987	1.1878
135	133	Raw & All	[11, 64, 64, 128]	[(4, 4), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.9983	0.7208	623	759	0.6426	0.5773	0.7114	0.6426	0.8016	0.5773	4.3154	1.4155
136	134	Raw & All	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[128, 64, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.0653	0.1801	203	759	0.4633	0.4691	0.8029	0.4633	0.6806	0.4691	4.1013	1.9165
137	135	Raw & All	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.1818	0.2812	282	759	0.3960	0.3553	0.8534	0.3960	0.6293	0.3553	6.2228	1.4549
138	136	Raw & All	[32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	4683	3455	0.1579	0.2684	503	759	0.4128	0.3585	0.8369	0.4128	0.6425	0.3585	7.3107	1.4110
139	137	Raw & Pristine	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1481	0.2746	203	412	0.1544	0.2793	0.9235	0.1544	0.3930	0.2793	1.8789	0.6746
140	138	Raw & Pristine	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.0222	0.1114	503	412	0.1868	0.2875	0.9297	0.1868	0.4322	0.2875	2.4784	0.7751
141	139	Raw & Pristine	[16, 32, 64, 128, 256]	[(3, 3), (3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.0956	0.2235	263	412	0.1991	0.2890	0.9166	0.1991	0.4463	0.2890	3.0172	0.5093
142	140	Raw & Pristine	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.4598	0.5040	263	412	0.5685	0.5798	0.7553	0.5685	0.7540	0.5798	3.2792	1.2963
143	141	Raw & Pristine	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	2.4089	1.2302	83	412	2.5165	1.2188	-0.0029	2.5165	1.5864	1.2188	7.4111	1.9564
144	142	Raw & Pristine	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.2981	0.3910	65	412	0.2997	0.3835	0.8812	0.2997	0.5475	0.3835	2.8246	0.9165
145	143	Raw & Pristine	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[500, 100, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.2431	0.3528	83	412	0.2765	0.3594	0.8840	0.2765	0.5258	0.3594	3.5098	0.9020
146	144	Raw & Pristine	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[500, 100, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1987	0.3184	83	412	0.1565	0.2844	0.9375	0.1565	0.3956	0.2844	2.0968	0.7592
147	145	Raw & Pristine	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[500, 100, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.0654	0.1942	263	412	0.1955	0.2757	0.9196	0.1955	0.4422	0.2757	2.7864	1.3480
148	146	Raw & Pristine	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[500, 100, 10, 1]	['relu', 'relu', 'relu']	adam	mse	mae	2541	1874	0.1073	0.2446	222	412	0.4249	0.4830	0.8394	0.4249	0.6518	0.4830	3.0647	1.3937
149	147	Raw & Pristine	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[500, 100, 10, 1]	['relu', 'relu', 'relu']	adam	mse	mae	2541	1874	0.0488	0.1679	263	412	0.1401	0.2553	0.9406	0.1401	0.3743	0.2553	1.7335	0.6032
150	148	Raw & Pristine	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[500, 100, 10, 1]	['relu', 'relu', 'relu']	adam	mse	mae	2541	1874	0.0725	0.2033	264	412	0.1999	0.2810	0.9117	0.1999	0.4472	0.2810	2.7796	1.7677
151	149	Raw & Pristine	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[500, 100, 10, 1]	['relu', 'relu', 'relu']	adam	mse	mae	2541	1874	0.2255	0.3341	143	412	0.2318	0.3239	0.8925	0.2318	0.4814	0.3239	3.3462	1.2866
152	150	Raw & Pristine	[16, 32, 64, 128]	[(3, 3), (3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu', 'relu']	[500, 100, 10, 1]	['relu', 'relu', 'relu']	adam	mse	mae	2541	1874	0.3007	0.3928	101	412	0.2166	0.3066	0.8994	0.2166	0.4654	0.3066	2.7327	0.8670
153	151	Raw & Pristine	[64, 128]	[(3, 3), (3, 3)]	['relu', 'relu']	[500, 100, 10, 1]	['relu', 'relu', 'relu']	adam	mse	mae	2541	1874	0.1024	0.2381	383	412	0.1852	0.3143	0.9287	0.1852	0.4304	0.3143	2.5415	0.9216
154	152	Raw & Pristine	[64, 128]	[(3, 3), (3, 3)]	['relu', 'relu']	[500, 100, 10, 1]	['relu', 'relu', 'relu']	adam	mse	mae	2541	1874	0.2776	0.3857	383	412	0.2089	0.3345	0.8922	0.2089	0.4571	0.3345	2.3850	1.0570
155	153	Raw & Pristine	[64, 128]	[(3, 3), (3, 3)]	['relu', 'relu']	[500, 200, 10, 1]	['relu', 'relu', 'relu']	adam	mse	mae	2541	1874	0.1123	0.2497	381	412	0.1791	0.2778	0.9112	0.1791	0.4233	0.2778	2.8473	0.5567
156	154	Raw & Pristine	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[500, 200, 10, 1]	['relu', 'relu', 'relu']	adam	mse	mae	2541	1874	0.1605	0.2908	143	412	0.1591	0.2778	0.9362	0.1591	0.3989	0.2778	2.8538	1.4023
157	155	Raw & Pristine	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[500, 200, 10, 1]	['relu', 'relu', 'relu']	adam	mse	mae	2541	1874	0.0632	0.1903	203	412	0.1894	0.2948	0.9213	0.1894	0.4352	0.2948	2.7058	0.9392
158	156	Raw & Pristine	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[500, 100, 10, 1]	['relu', 'relu', 'relu']	adam	mse	mae	2541	1874	0.2120	0.3340	203	412	0.1901	0.3194	0.9281	0.1901	0.4360	0.3194	2.1552	0.7785
159	157	Raw & Pristine	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[500, 100, 10, 1]	['relu', 'relu', 'relu']	adam	mse	mae	2541	1874	0.1210	0.2503	443	412	0.2219	0.3211	0.9089	0.2219	0.4711	0.3211	2.2318	1.4782
160	158	Raw & Pristine	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1491	0.2829	143	412	0.2113	0.3051	0.9037	0.2113	0.4597	0.3051	3.3991	0.9975
161	159	Raw & Pristine	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1323	0.2589	143	412	0.1603	0.2780	0.9221	0.1603	0.4004	0.2780	2.9534	0.7475
162	160	Raw & Pristine	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1360	0.2662	143	412	0.1650	0.2711	0.9301	0.1650	0.4062	0.2711	2.5330	1.6109

Figure B.5: Hyperparameter settings, resultant training and validation for all 180 CNN model iterations (4 of 5).

	B	C	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	Model ID		Model Parameters								Model Training					Model Validation								
2	Run	Dataset	Conv2D Shapes	Conv2D Windows	Conv2D Activations	Dense Shapes	Dense Activations	Compile Optimizer	Compile Loss	Compile Metrics	Dataset Size	Training Size	Train Loss	Train Metrics	Training Time	Validation Size	Val. Loss	Val. Metrics	R2	MSE	RMSE	MAE	Max Error	Max Error Perc
163	161	Raw & Pristine	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1087	0.2377	203	412	0.2287	0.3138	0.8982	0.2287	0.4782	0.3138	3.0167	1.1762
164	162	Raw & Pristine	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['tanh', 'tanh', 'tanh']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1028	0.2352	143	412	0.1564	0.2668	0.9364	0.1564	0.3955	0.2668	2.5092	1.5957
165	163	Raw & Pristine	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.0288	0.1256	143	412	0.2520	0.3517	0.8944	0.2520	0.5020	0.3517	2.6246	0.8920
166	164	Raw & Pristine	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1319	0.2605	128	412	0.1765	0.3056	0.9219	0.1765	0.4201	0.3056	1.7546	1.0374
167	165	Raw & Pristine	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1491	0.2809	143	412	0.1711	0.2842	0.9290	0.1711	0.4137	0.2842	1.8494	0.8514
168	166	Raw & Pristine	[128, 128, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1419	0.2662	353	412	0.2088	0.3150	0.8929	0.2088	0.4570	0.3150	2.4453	1.3222
169	167	Raw & Pristine	[128, 128, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1060	0.2426	351	412	0.1594	0.2804	0.9243	0.1594	0.3992	0.2804	1.6211	0.8012
170	168	Raw & Pristine	[64, 128, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1874	0.1179	0.2530	248	412	0.1668	0.2677	0.9255	0.1668	0.4084	0.2677	3.0508	0.5624
171	169	Raw & Pristine	[64, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1632	0.0874	0.2210	185	654	0.2241	0.2939	0.9020	0.2241	0.4734	0.2939	2.8978	1.0033
172	170	Raw & Pristine	[64, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1777	0.1208	0.2576	187	509	0.1395	0.2904	0.9359	0.1395	0.3736	0.2904	1.2145	0.4416
173	171	Raw & Pristine	[128, 128, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1777	0.1038	0.2353	383	509	0.1878	0.2983	0.9315	0.1878	0.4333	0.2983	2.5510	1.6896
174	172	Raw & Pristine	[64, 128, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1777	0.0761	0.2056	263	509	0.1299	0.2576	0.9346	0.1299	0.3604	0.2576	2.2419	0.6004
175	173	Raw & Pristine	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1777	0.1155	0.2430	83	509	0.1227	0.2499	0.9486	0.1227	0.3503	0.2499	2.0523	1.3593
176	174	Raw & Pristine	[64, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1777	0.0178	0.1037	559	509	0.1736	0.3141	0.9246	0.1736	0.4166	0.3141	1.5848	0.7030
177	175	Raw & Pristine	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1777	0.0303	0.1325	263	509	0.1365	0.2699	0.9466	0.1365	0.3694	0.2699	1.6427	0.7606
178	176	Raw & Pristine	[64, 64, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1777	0.1145	0.2443	203	509	0.1645	0.2630	0.9312	0.1645	0.4056	0.2630	2.7269	0.8188
179	177	Raw & Pristine	[64, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1777	0.1124	0.2480	203	509	0.1822	0.2878	0.9296	0.1822	0.4269	0.2878	2.9918	1.3130
180	178	Raw & Pristine	[64, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1777	0.1111	0.2441	188	509	0.1899	0.2806	0.9166	0.1899	0.4358	0.2806	2.5634	0.8941
181	179	Raw & Pristine	[16, 32, 64]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1777	0.1251	0.2543	83	509	0.2185	0.2968	0.8915	0.2185	0.4675	0.2968	2.5579	1.2862
182	180	Raw & Pristine	[64, 64, 128]	[(3, 3), (3, 3), (3, 3)]	['relu', 'relu', 'relu']	[100, 10, 1]	['relu', 'relu']	adam	mse	mae	2541	1777	0.1285	0.2616	188	509	0.2113	0.3143	0.9001	0.2113	0.4597	0.3143	3.0248	0.8810

Figure B.6: Hyperparameter settings, resultant training and validation for all 180 CNN model iterations (5 of 5).