Application of Artificial Neural Nets to a Laser Additive Manufacturing Process for Fault Detection using Powder Bed Height Data

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Benjamin S. Terry

August 2020

Copyright © 2020 by Benjamin Terry
All Rights Reserved

ACKNOWLEDGEMENTS

Thank you to Dr. Chakraborty, Dr. Babu, and Dr. Qi, who acted as my professors in the primary fields my work dealt with and for acting as my committee.

Thank you to Brandon Baucher, who was the primary contact with at Applied Optimization, the sponsor of the project.

Thank you to the Engineering Fundamentals Program and the Haslam College of Business, who provided funding for my education.

And thank you to Julie Harden at the Graduate Office and Julie Knoefel in the MABE Department, who worked diligently with me to make sure all of my forms were correctly submitted.

ABSTRACT

The purpose of this paper is to report recent results demonstrating feasibility of active monitoring and fault probability estimation in the Selective Laser Melting (SLM) process in a Renishaw AM250 machine, through analysis of layer-by-layer surface profile data of Fe_3Si powder. The data was collected in-situ during the metal additive manufacturing of a Heat Exchanger section, comprised of a series of conformal channels. Specifically, a shallow artificial neural net (ANN) was trained with high-resolution powder bed surface height data from a laser profilometer and then linked to post-print CT scans which provided the truth-data labelling of each site as faulty or nominal. Various measures of accuracy and performance demonstrate excellent performance of the ANN, suggesting that the ANN is capable of discovering strong correlations between surface roughness characteristics and the presence and size of faults.

These results were generated by grouping the profile data using post scan CT data, which would not be available in-situ. As such, further work was performed to apply the NN and use insights gained from development of the NN to identify faults in-situ using only the available in-situ data. First an application of the NN was tried on un-preprocessed data but failed to reach satisfactory levels of accuracy. Next, a deeper understanding of the internal process was developed by systematically studying fault sights, their roughness values, and interaction with the NN. This data and insight guided future steps. It was found that faults tended to have higher peaks and lower valleys in close proximity than nominal regions and were being correctly classified as such by the NN. As such, the next step involved developing an algorithm to a priori determine what to consider as a grouped region using the presence of extreme profile data height values. This is referred to as “min-max” stitching later on in the report. The final step involved iterating on this algorithm and revisiting the raw data to try and uncover any other noticeable trends.

TABLE OF CONTENTS

Chapter One Introduction and General Information 1
Overview 1
Chapter Two Literature Review 2
Additive Manufacturing 2
Overview 2
Determining Properties of Additively Manufactured Parts 4
Traditional Monitoring and Validation Methods for Additive Manufacturing 11
Machine Learning Overview 12
Neural Networks 13
Evaluating Performance 15
Current Applications of Machine Learning to Additive Manufacturing 19
Chapter Three Neural Net training and Associated Methods and Results 21
Experimental Methods and Procedures 21
Build Setup and Parameters 21
Data Preprocessing 25
Alignment of Sensor Passes 25
Masking 28
Mapping and Format Differences 29
Neural Networks 30
Basic Inputs and Parameters 30
Processing of CT Scan Data to Provide Ground Truth 31
Stitching of Faulty Supercells 33
Neural Net Training and Parameters 44
Chapter Four Results and Discussion of Training 46
Parametric Study 46
Training Results 51
Multiclass Classification 56
Chapter Five Neural Net training and Associated Methods and Results 61
Neural Nets Used for Application Efforts 61
Oak Ridge National Lab (ORNL) Data and Neural Net 61
Penn State Data and Neural Net 65
Application Results 70
Baseline Results 70
Manual faults Investigation 71
Min-Max Stitching Algorithm and Impact of Using Neural Net 76
Manual Investigation of Stitching Algorithm 84
Chapter Six Discussion and Future Work 93
List of References 94
Vita 99

LIST OF TABLES

Table 2.1. Confusion Matrix [29] 16
Table 3.1. Build Parameters for Renishaw AM250 23
Table 3.2. Resolution and accuracy of the laser surface profilometer 24
Table 3.3. Notations and Variable Names for Algorithms 1 and 2 36
Table 3.4. Confusion Matrix 45
Table 4.1. Test Matrix for Parametric Study 46
Table 4.2. Confusion Matrix for High Cell Resolution 51
Table 4.3. Confusion matrices for (a) training set, (b) validation set, (c) testing set, and (d) all of the data 54
Table 4.4. Comparison of test set results using a NN that was trained a) with a validation set and b) without a validation set. Cells were 8-by-19 pixels, with a threshold for faults of 3 SD above the mean across 5 layers. 55
Table 4.5. Multiclass Confusion matrices for (a) training set, (b) testing set, and (c) all of the data 59
Table 5.1. Confusion Matrices for NN-ORNL used in application. (a) training set (b) testing set (c) all samples. 64
Table 5.2. Confusion Matrices for NN-PS used in application. (a) training set (b) testing set (c) all samples. 68
Table 5.3. Confusion Matrix for application of NN-PS to the PS part. As this was application, the data was not subset into training, validation, and testing sets. 71
Table 5.4. Manually Found Fault Scores for Layer 21 of a fault in ORNL HeX B Cell by Cell 73
Table 5.5. Surface Parameters (equations 1-5) 74
Table 5.6. Manually Found Fault Scores of a fault in ORNL HeX A by Layer when Considering Entire Fault as Region 75
Table 5.7. Confusion Matrix applying a Net trained without stitching to ORNL HeX-B (using the NN trained on stitched ORNL data had similar results). 81
Table 5.8. Confusion Matrix of applying a NN-ORNL to ORNL HeX-B 84
Table 5.9. Roughness values for individual and combined regions of fault visualized in Figure 5.8. 86
Table 5.10. NN Scores, using both NNs, for individual cells of the fault visualized in Figure 5.8. The first number represents the score for how nominal the NN thinks the cell is; the second for how faulty. 88
Table 5.11. Summary of NN scores and roughness values when all highlighted cells are grouped and considered a single region 89
Table 5.12. Summary of NN scores and roughness values for cells that were grouped by the min-max algorithm. These cells are colored blue and red in Figure 5.8 and Table 5.10. NN Scores, using both NNs, for individual cells of the fault visualized in Figure 5.8. 90

Table 5.13. Summary of NN scores and roughness values for cells that were identified as faulty using the post-build CT data. As such, these cells, combined, are the best representation of how the NN scores an actual fault. These cells are colored grey and red in Figure 5.8 and Table 5.10. NN Scores, using both NNs, for individual cells of the fault visualized in Figure 5.8.....91
Table 5.14. Summary of NN scores and roughness values for cells that were identified as faulty using the post-build CT data and stitched using the min-max stitching method. In other words, these values are representative of the NN working correctly to identify faults. These cells are colored red in Figure 5.8 and Table 5.10. NN Scores, using both NNs, for individual cells of the fault visualized in Figure 5.8. 92

LIST OF FIGURES

Figure 2.1. Characterization of various (a) polymer- and (b) metal-based AM processes. 3
Figure 2.2. (a) Schematic showing the generic model of a weld melt pool[6]. (b) Schematic showing calculated Isotherms in a melt pool [7]. Note how these images are similar to the processes seen in a metal AM process. 5
Figure 2.3. Schematic showing factors and how they influence final part quality in welding [7]....6
Figure 2.4. (a) Graph showing how solidification parameters affect grain structure [9] (b) Examples of basic grain structures [10]. 8
Figure 2.5. Phase Diagram of Iron and Silicon [14]....9
Figure 2.6. (a)Single-layer Neural Network or Linear Discriminant Function [27] and (b) multi-layer Neural Network [28]. 14
Figure 2.7. Sigmoid Function 15
Figure 2.8. Example ROC Curve [30]. 17
Figure 2.9. Visualization of overfitting 18
Figure 3.1. (a) 3D model of the part, a heat exchanger with conformal channels (b) The finished build plate of the parts used for this research. 22
Figure 3.2. A schematic showing the timeline for scanning a single layer 23
Figure 3.3. Example of Raw CT Scan Output as a (a) tif File used for primary development , and (b) as a MATLAB Binary Array 25
Figure 3.4: (a) Overlapping area of the forward and backward scans are arranged next to each other to demonstrate the shifting and distortion that have to be corrected in order to stich the two scans together. Here data is filtered to reveal only the higher levels of un-melted powder inside the channels. (b)Labelled contiguous regions are isolated based on connectedness and size for both scan passes. The centroid for each contiguous region is identified along with major and minor axes for best fit.26
Figure 3.5. Profile Data (a) of the entire build before allignment and a single part after allignment and filtering displayed as a (b) 3D isometric view and (c) colormap for a single layer....27
Figure 3.6. (a) Mask Extracted from Part File (b) Usable Height Data, broken into cells 28
Figure 3.7. (a) Raw CT Scan Data imported to MATLAB and extracted to match profile data dimensions; faults are represented by black spots. (b) Grey cells are used to indicate cells that should be marked as faulty. Red bounding boxes indicate the edges of that fault on the given layer; note the apparent discontinuity in some regions. This is due to connectivity being determined 3-dimensionally. 32
Figure 3.8. Example of extracted fault using the method of connected components. 34

Figure 3.9. Scatter plot of data points generated using Algorithm 1. Note that there is little to no apparent, systematic separation between nominal and faulty data. 39
Figure 3.10. Histograms showing interesting trends in the height distributions for nominal and faulty cells 41
Figure 3.11. Demonstration of how number of layers affects detected faults: (a) 2 layers (b) 5 layers 43
Figure 3.12. Example ROC Curve [30]. 45
Figure 4.1. Results of Parametric study, displayed as a colormap. Rows: F1 Score, Accuracy, Performance. Columns: No Threshold, Threshold as Average Size, Threshold as 3 SD above mean. 47
Figure 4.2. High Cell Resolution across 11 layers. (a) Example output of stitching for Layer 23 (b) ROC curve 50
Figure 4.3. ROC curve for (a) training set, (b) validation set, (c) testing set, and (d) all of the data 52
Figure 4.4. Multiclass ROC curve for (a) training set, (b) testing set, and (c) all of the data 57
Figure 5.1. ROC Curves for NN-ORNL used in application. (a) training set (b) testing set (c) all samples. 62
Figure 5.2. ROC Curves for NN-PS used in application. (a) training set (b) testing set (c) all samples. 66
Figure 5.3. ROC Curves for for application of NN-PS to the PS part. As this was application, the data was not subset into training, validation, and testing sets. 71
Figure 5.4. Close-up of investigated fault 72
Figure 5.5. Application of NN-PS to ORNLL HeX-B 76
Figure 5.6. Results of applying a Net trained without stitching to ORNL HeX-B (using the NN trained on stitched ORNL data had similar results). a) ROC curve b) visualization of stitching and NN labeling 79
Figure 5.7. Results of applying a NN-ORNL to ORNL HeX-B a) ROC curve b) visualization of stitching and NN labeling 82
Figure 5.8. Visualization of fault. (a) raw CT scan (b) Effect of Algorithms and NN; grey cells are identified by the CT scan as ground truth, blue represents data stitched by the algorithm and identified by the NN, red is both. 85

CHAPTER ONE

INTRODUCTION AND GENERAL INFORMATION

Overview

The purpose of this use is to investigate correlations between powder bed height in a selective laser sintering process and faults in the final product using a machine learning (ML) techniques, namely neural networks. The work performed consists of data collection, multiple phases of data processing, neural net training, and application of trained neural nets to other parts.

The novelty of this approach is the use of powder bed height data. To date, traditional monitoring techniques have been focused on thermal modeling of the melt pool (cite). This approach further distances itself from current ML based approaches have largely restricted themselves to the use of optical imaging.

The work done is separated into two parts. The first focusing on training the neural net and the second focuses on attempts to apply it to novel parts. The first section involves discussions on how the data was collected and processed, the neural network was trained, the results of this training, and other insights gained during this process. The second overarching section shows attempts to apply the NN to a novel part, separate from the initial data set. Many issues were found in this attempt and efforts were largely put toward manually examining individual faults and the results of applying the NN.

Overviews of relevant, current theory and practices in the fields of AM and ML will be provided in the Literature Review.

CHAPTER TWO

LITERATURE REVIEW

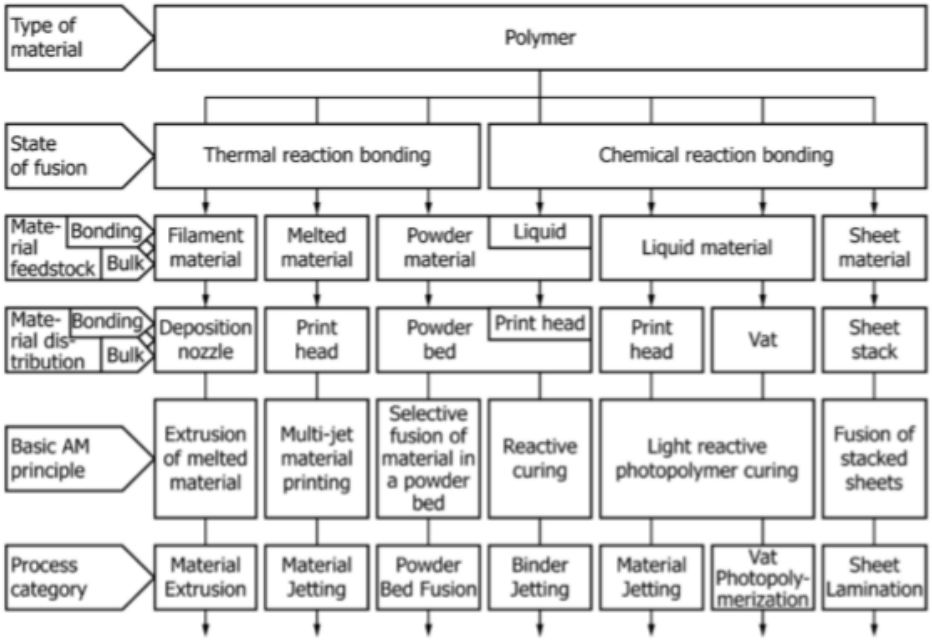
This project focused on applying machine learning (ML) techniques to an additive manufacturing (AM) process. As such, the literature review will be broken down into 4 sections. First will be an overview of AM, looking a little at the history of the technology and applicable theoretical models. Second will be an overview of recent research into monitoring the AM process; in general, this is limited to correlating and controlling input parameters to the properties and quality of the finished part. Third, an overview of relevant technique from ML will be discussed. Last, the fields will be joined, examining current research in applying ML to AM processes.

Additive Manufacturing

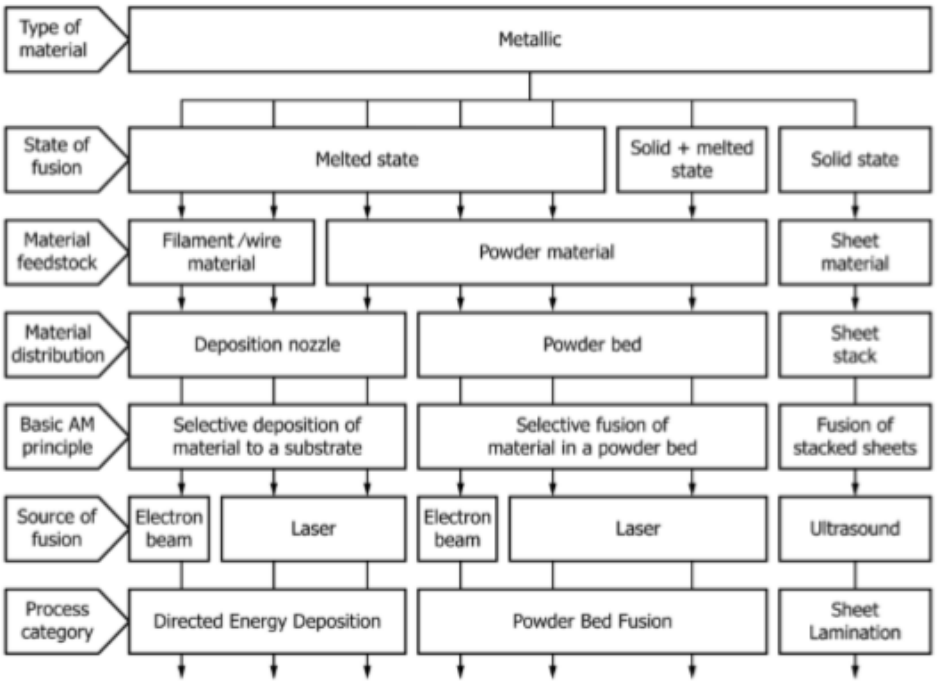
Overview

AM generally refers to the process of depositing and binding material layer-by-layer to build a final part. From early applications to today, it has been referred to as rapid prototyping. While a traditional CNC machine or process may be faster for a single step, the setup and planning for a single step as well as the number of steps required makes AM a timelier solution. However, the quality of AM parts has increased over time, to the point that some end parts are produced in this manner [1]. Others still require only certain post-build processing (i.e. Hot Isostatic Pressing, Surface Finishing, etc.) to be considered finished parts [2].

As of 2015, ASTM International and ISO, two international standards organizations, have characterized AM processes fist by material (metallic, polymer, ceramic, and composite) and then by a number of process details; the overarching categories for these details are the same for each material, but each material has different details. Charts detailing this characterization for polymers, one of the most popular materials, especially for hobbyists, and metals, the current manufacturing materials of choice and focus for this research, are presented below in Figure 2.1. As with most attempts at characterization and classification, this is just one framework amongst many. For example, 3D printing is attributed as having first been invented in 1983 by Charles Hull. His technology, stereolithography, was patented in 1986, when Hull founded 3D Systems. However, stereolithography is not on these charts; by this categorization, the technology would be considered a form of vat polymerization. Of course, that may just be that the charts need to be delineated further; however, it is an interesting note.



(a)

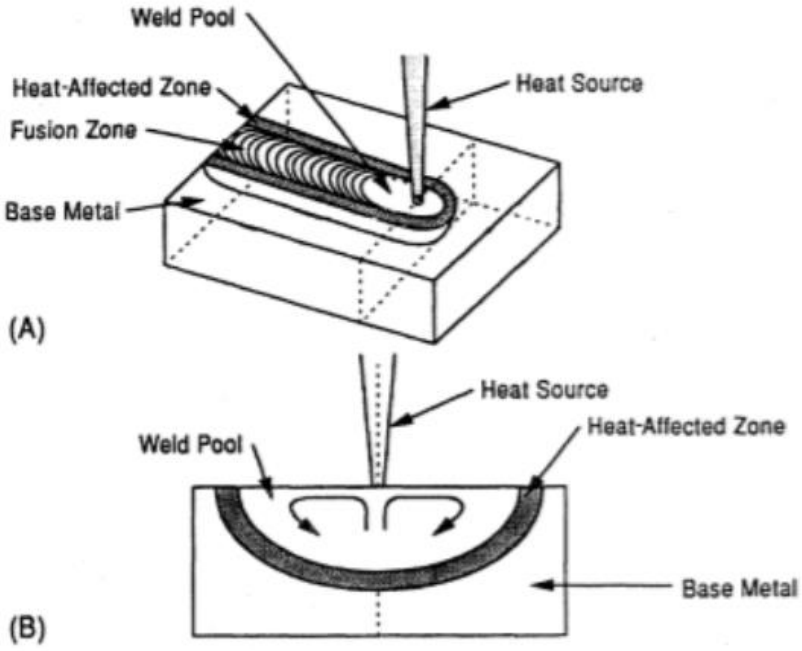


(b)

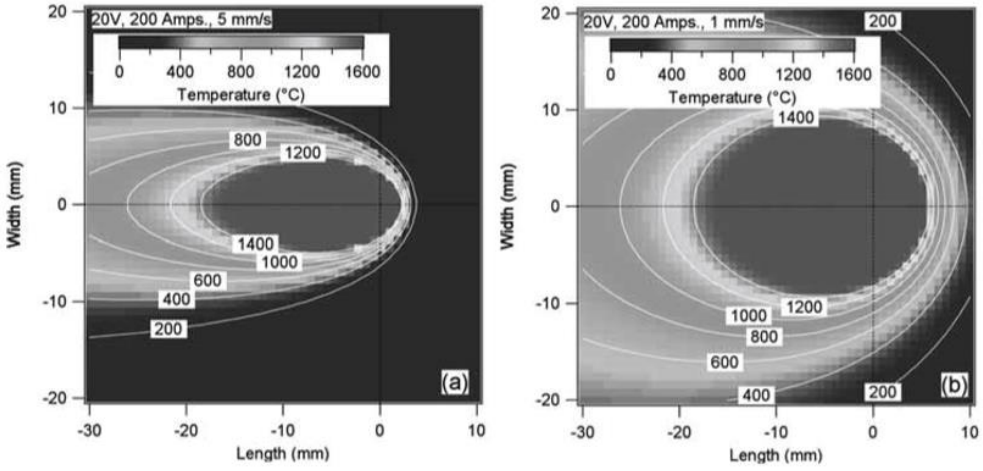
Figure 2.1. Characterization of various (a) polymer- and (b) metal-based AM processes.

Determining Properties of Additively Manufactured Parts

The relevant theory used in modern powder-based metal AM is derived primarily from welding literature discussing moving sources of heat and how it drives phase changes and grain formation (which in turn determine mechanical properties and applicability as a manufactured part). This is due to similarities in the physics involved. To see this visually, consider Figure 2.2; note how they are similar to a metal AM process. Further, several instances in the literature have used welding as the basis of their models to success [3-5].



(a)



(b)

Figure 2.2. (a) Schematic showing the generic model of a weld melt pool[6]. (b) Schematic showing calculated Isotherms in a melt pool [7]. Note how these images are similar to the processes seen in a metal AM process.

There are, of course some differences. For example, a powder-bed process, the powder will conduct heat differently than the solidified part [4]. Another example is the effect of rastering. In welding, a single line is often the extent of the weld. In AM, the laser scans back and forth. As such, in a simple back-and-forth scan pattern, one side of the scan will be significantly hotter. The effects of this can be seen in [8]. Regardless, it is thus well established that welding theory and literature, which has a rich history, is well applicable to metal AM processes. In understanding weld solidification, there are two primary topics that must be considered: grain and phase. A basic overview of these relationships can be seen in Figure 2.3.

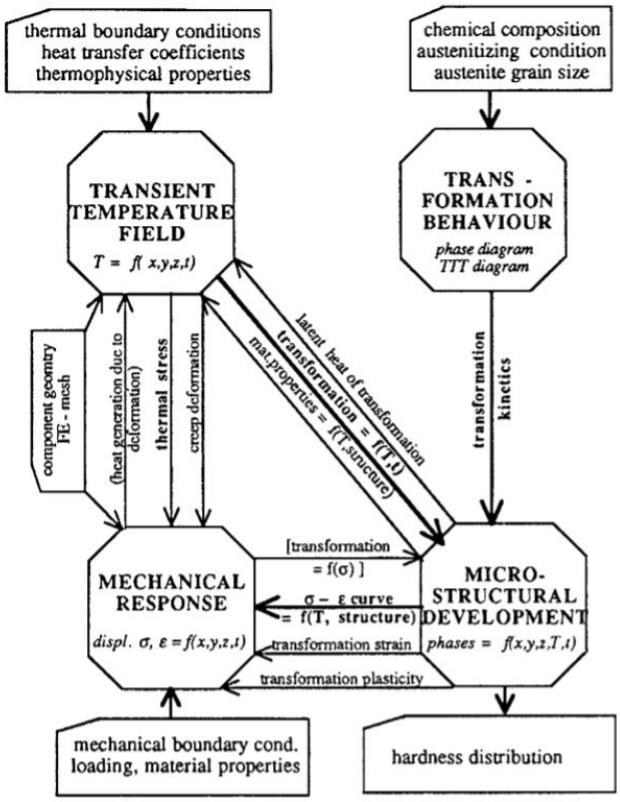


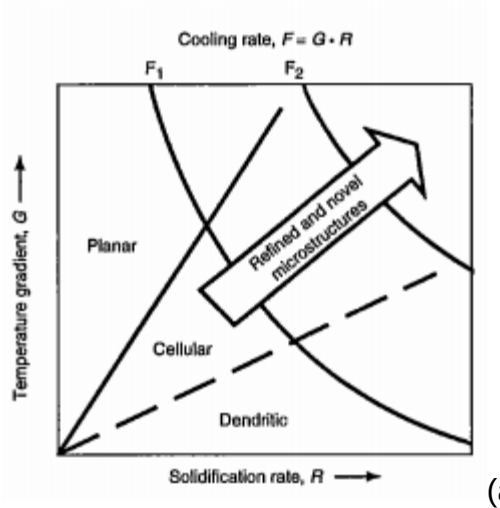
Figure 2.3. Schematic showing factors and how they influence final part quality in welding [7].

Grain Formation

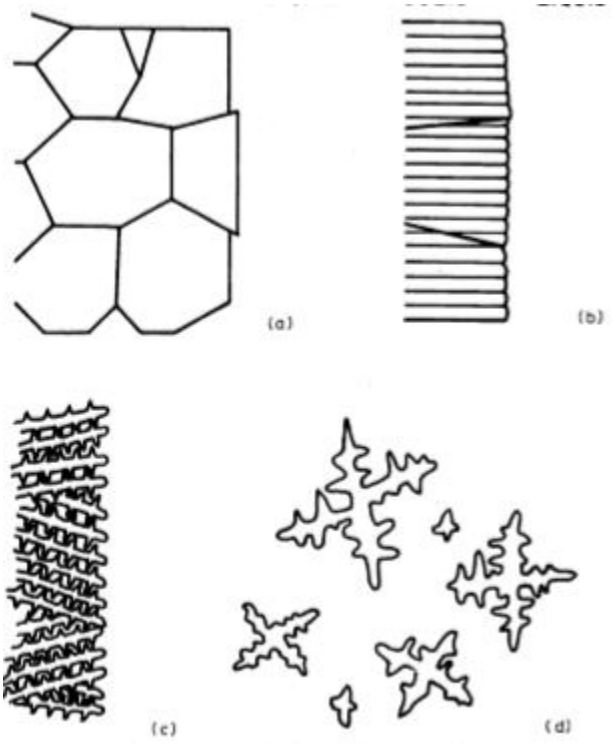
Grain formation is influenced by 4 primary factors: partition coefficient, temperature gradient, solidification rate, and cooling rate [9, 10]. These are described mathematically in Equation 2.1 through Equation 2.4.

Partition Coefficient	$k = \frac{C_S}{C_L}$	Equation 2.1
Temperature Gradient	$G_L = \frac{dT}{dx}$	Equation 2.2
Solidification Rate	$R = \frac{dx}{dt}$	Equation 2.3
Cooling Rate	$G \cdot R = \frac{dT}{dt}$	Equation 2.4

Here, C_s and C_L refer to the solute concentration of solid and liquid, respectively, at the solid-liquid interface of the melt pool. T refers to the temperature, t refers to time, and x refers to the location; solidification rate describes how fast the solid-liquid interface is moving and directly relates welding (or laser, in AM) speed [10]. Further, after much experimentation, the effect of these variables on grain formation has been well document and are summarized in Figure 2.4, which also gives a brief visualization of grain structure. The exact mechanical properties of



(a)



a plane front growth; *b* cellular growth; *c* columnar dendritic growth; *d* equiaxed dendritic growth

(b)

Figure 2.4. (a) Graph showing how solidification parameters affect grain structure [9] (b) Examples of basic grain structures [10].

Of course, every alloy will behave differently and have different specific cutoffs points, but the general schema applies. In the context of welding, this schema can be used to determine optimal weld parameter as well as predict material properties. However, the applications become much more interesting when applied to AM. As discussed prior, welding usually involves a single pass while AM involves many consecutive adjacent passes of the laser and potential heat build-up from sintering of prior layers. However, this is not required; the scan path of the laser can be modified in many AM applications. As such, the engineer can utilize this theory and directly influence the microstructure of the part through a well-planned scan strategy [11].

Phase Changes and Phase Diagrams

The next element to consider is the underlying crystal structure. The basic idea is that different compositions of materials at different temperatures will form different atomic structures. A overview of this and related topics is provided by the University of Cambridge [12]. The phase diagram of Iron-silicon, the material used in these experiments, is provided in Figure 2.5. Fe_3Si falls in the silicon rich region of the *α* solid solution region [13].

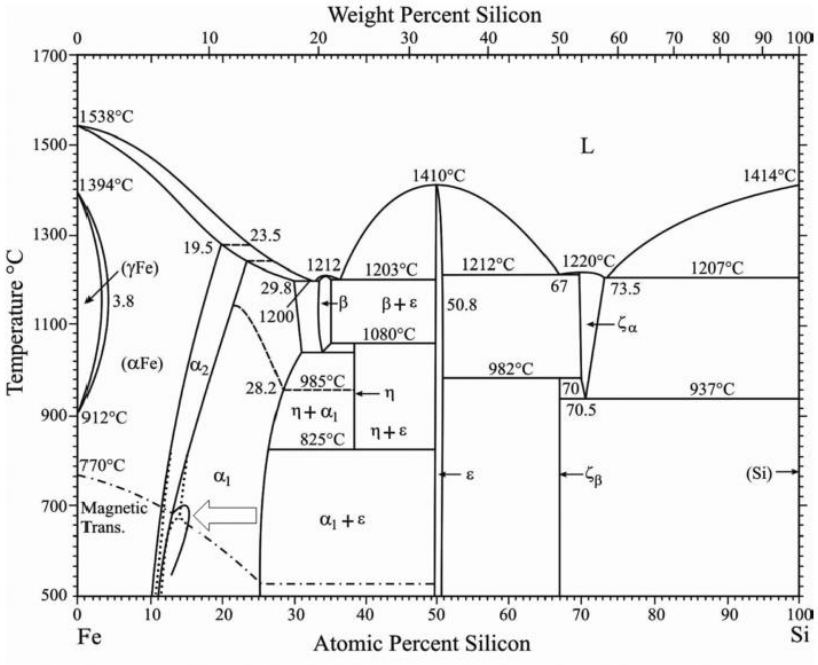


Figure 2.5. Phase Diagram of Iron and Silicon [14].

Another important aspect of phase is cooling rate. Consider, for example, annealing and quenching. Both change the underlying structure of the metal by raising it to a specified high temperature and letting it cool. However, annealing allows the part to cool slowly and in a controlled environment, allowing the atoms to realign to the phase indicated by the phase diagram; annealing is thus done to encourage changes in grain structure and to remove residual stress, but not change phase. However, quenching involves rapid cooling, which "locks" the phase in the state it achieved at the higher temperature. Examples of how cooling rate affect microstructure can be seen in [15]. In an AM process, the cooling rates are usually much higher and may result in undesirable properties. Further, the thermal cycling cause by adjacent passes of the laser must be considered. In total, properly adjusting the laser power and the heat of the printing chamber can mitigate these issues [16, 17].

Heat Transfer, Metal Pool Morphology, and Solidification

Understanding this, the importance of modeling the thermal distribution of the melt pool and surrounding area is clearly seen. The basis for most modern modeling can be traced back to the work of Rosenthal in the 1930s and 40s. Starting from the basic form given in, he applied various assumptions and situations to arrive at the basic equation for a semi-infinite plate,

Basic Equation	$$\frac{\delta^2 t}{\delta y^2} + \frac{\delta^2 t}{\delta x^2} + \frac{\delta^2 t}{\delta z^2} = 2\lambda \frac{\delta t}{\delta s}$$	Equation 2.5
Semi-Infinite Plate	$$t = t_0 + \frac{Q}{2\pi k R} e^{(-\lambda v(R+\xi))}; R = \sqrt{\xi^2 + y^2 + z^2}$$	Equation 2.6

Where x, y and z, represent the coordinate directions. ζ is used in place of x as Rosenthal derived his equations using the heat source as the origin. t is temperature; t_0 is the initial temperature. Q represents the heat source. k and λ both relate to thermal conductivity, with k actually being thermal conductivity. v is the speed of welding [18]. The model does have drawbacks, such as not considering the effects of convection and fluid flow [10]. However, many still use Rosenthal's framework, even if only as the basis for expansion [4, 7, 10, 19]. Other contemporary approaches have taken approaches relating to the conservation of mass, momentum, and energy [8, 20] Many apply and solve these equations using a finite element approach [8, 20, 21].

Traditional Monitoring and Validation Methods for Additive Manufacturing

From this perspective, it is easy to see why most of the work in monitoring AM process have focused on monitoring the melt pool and its characteristics. Tapia and Elwany provide a review of these methods as well as control efforts and common methods and materials investigate in [21]. Unsurprisingly, the field is dominated by pyrometers (photodiodes and digital IR cameras) and thermocouples. Thermocouples are useful for providing ground truth data about temperature, as will be seen, but has the drawback of requiring physical contact. Displacement sensors are used, though more so with direct energy deposition methods [21].

To explore some of the work and issues in using pyrometers, specifically IR cameras to monitor AM processes, consider first the work of Jake Raplee under Dr. Babu. His work focused on the difficulties in thermal modeling of an AM process. An IR camera uses the emittance of the surface to measure temperature. However, the thermal emittance of the liquid melt pool, powder metal, and as-printed material are different, meaning the apparent temperature will be different. The first step to fixing this problem is calibrating the IR camera to the different emittances. This was done by measuring the ground truth temperature of the as-printed and powder material via Type-K thermocouples and recording the corresponding IR camera output. From there, a 6^{th} degree polynomial was fit to the relationship. Next, the transition point must be recognized; knowing the differences in emittance is pointless if which model to apply is not. The solution is that, due to the difference emittance, the apparent temperature read by the IR camera, using the calibrated powder equation, will drop rapidly when melted. As such, looking for this drop can indicate when to apply which model. This process, however, is exacerbated by the potential that the powder may not melt and heat transfer from previously melted regions. To fix these problems, the apparent temperature of the area before and after melting can be compared. If the point is unmelted, it will still follow the powder metal emittance calibration profile and the apparent temperature using that equation after the melt will be the same. Slope is also used to resolve these question; when actual melt occurs, the apparent temperature drops much more rapidly, allowing a threshold to be set.

This work also noted how the presence of dips or rises in the powder bed can also interfere with accurate measurement, though this was noticed post build and steps were not taken at the time to compensate. The final point of note from this article is the generic difficulty in measurement. AM machines are generally enclosed environments and use proprietary software; setup of a monitoring system can be difficult. Further, AM can provide unique challenges, such as molten metal splattering and potentially damaging equipment if not protected [5].

Other investigations into this topic have been made by Plotkowski et. al, who used thermal imaging to validate models first presented by Komanduri and Hou [19]. This initial model expanded the Rosenthal Equations to consider a ring heat source. The results showed good agreeance with theory [4]. Another example is the work by Lee and Zhang, who sought to validate a model based on mass, momentum, and energy, as opposed to Rosenthal's equations. However, the monitoring was limited to post-build examination, that showed agreement with their proposed model. Of note from this work is the novel approach to modeling, 3-D generated models, and the observation that evaporation recoil pressure cause a depression of the melt pool at its leading edge [8].

Moving on, a majority of work to date has focused on process parameters and their effects on final part quality [21, 22]. Efforts have and are being made into more active control of the process and are still the source of constant research. Many research groups, such as Zheng et. al [23] have developed control systems, though work is still ongoing to implement them. Serruys et. al. even have a paten for a online monitoring laser control system, though further investigation shows limited application in the literature [24]. Kleszczynski et. al. have also proposed a feedback system, based on optical imaging, to find and correct issues common, predefined process irregularities; the paper demonstrated success with these methods. [25]

Machine Learning Overview

Moving away from traditionally based models, consider machine learning (ML) approaches. Machine Learning, at its core, is about using prior data to build a model that determines the most likely outcome based on a set of inputs (also often referred to as features). For example, let's say there is a data set. Each data point is a vector of 6 features and has a corresponding output class. These classes are often assigned numbers, 1 and 0 for the binary case of only two classes. They are also commonly referred to as positive and negative. A commonly used examples is identification of disease. A 1 indicates the *positive* presence of a disease; a 0 represents the absence (or *negative*) of the disease. From here, the data is used to train a model that will take a vector of 6 inputs and determine which class the inputs belong too. It is the generic form of these models and how their parameters are determined that is primary component of ML techniques. Some techniques can be solved more analytically, such as linear regression and Bayesian Decision Theory. Others require defining and then minimizing some form of cost function to build their model, such as Decision Trees. It is noted that more complex models often have theoretical bases in these simpler analytical models and that all are in some way based on probability and choosing the output with the maximum likelihood. The only difference is how the model determines likelihood of an outcome. All will carry some level of inaccuracy and all incorporate functional parameters that map inputs to outputs. Training adjusts these parameters to achieve the best results; what is "best" will be discussed later.

Further, the process is rarely that simple and many other steps are required. For example, while the primary topic of this work is applying a neural network (NN) to an AM process, a majority of the work was in data processing and visualization.

Further, the resulting model may not have any physical basis; thus, while practically applicable, there may be no insight gained. However, assuming the physical model has any validity, it is likely that the learned model will have elements. And regardless of this potential disparity, there is one key element that ties machine learning to physical models: the inputs. Every machine learning model requires some form of data as an input variable. While some techniques work with less physically defined inputs (convolutional Neural Nets, for example), most do. Indeed a commonly noted element of machine learning is the importance of domain knowledge in designing features [26].

While there are multiple different models for machine learning, such as simple linear regression, Bayesian Decision Theory, Decisions Trees, etc., the method used in this work is neural networks.

Neural Networks

The basic idea governing a single layer “Neural Network” is seen in Figure 2.6. Neural Networks is in quotes because while this could be classified as such, the case of as single layer corresponds to a simple linear decision rule separating the two classes. That is, the ML model produces a linear combination of the input features of a single data point and a bias unit, which has a weight of 1, to produce the output. Comparing a neural network to linear regression, the bias unit is like the intercept. A decision is then made based on this value. Weights, when discussing neural nets, are often written as $\vec{w}$, representing a single vector mapping the input to the output.

The true power and complexity of Neural Networks comes in adding more layers. In those cases, each layer results in a linear combination of outputs from the previous layer (or the input layer). Each layer functions the same as a single layer model, but by stacking them, the computer can build interesting transformations that far exceed a simple linear combination. This is especially true when one considers that each output of 1 layer feeds into every node of the next layer. That is, each node at each layer has a separate parameter vector $\vec{w}$. A multi-layer Neural Network is also shown in Figure 2.6. The first layer, comprising the features of the data point is referred to as the input layer. The middle layers are referred to as hidden layers. The final layer, which takes in the outputs of the last hidden layer and converts them to a final decision (or a final score with which to make a decision), is referred to as the output layer.

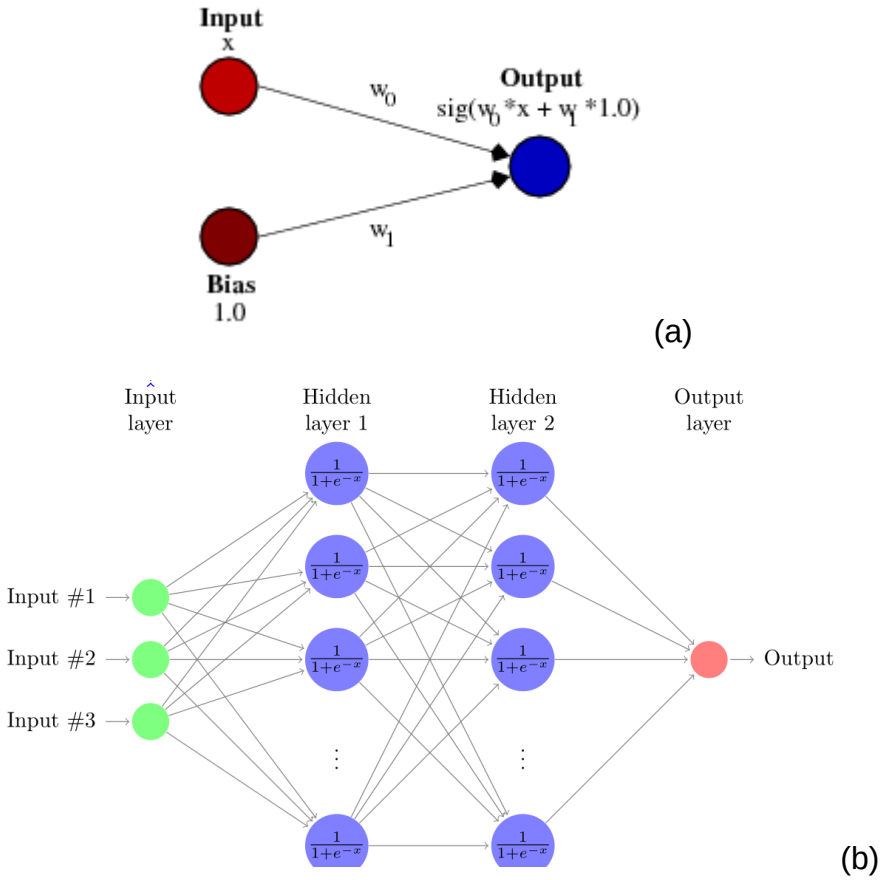


Figure 2.6. (a)Single-layer Neural Network or Linear Discriminant Function [27] and (b) multi-layer Neural Network [28].

The expression seen in Figure 2.6 is known as a sigmoid function. Mathematically, it is express in Equation 2.7 and looks like Figure 2.7.

Sigmoid Function	$S(x) = \frac{1}{1 + e^{-x}}$	Equation 2.7

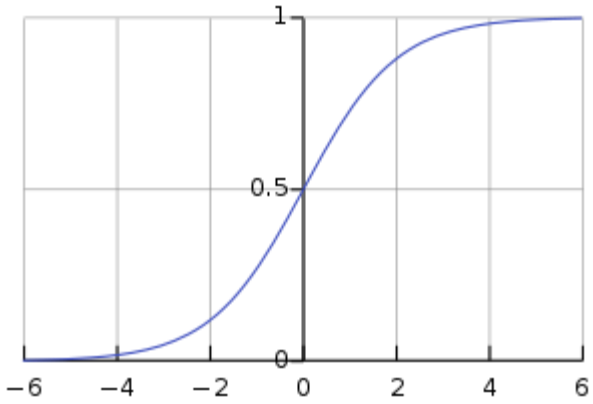


Figure 2.7. Sigmoid Function

The sigmoid function is used to drive outputs greater than zero to one and those less than zero to zero. The sigmoid function is a commonly used function with neural networks and mainly aids with computation, understanding, and avoiding bias. If left alone, the output of any layer could be extremely large or small, so it arbitrarily shrunk. This can aid programming in assuming the inputs to a given hidden layer are always between 1 and 0. It ensures that no weight dominates the output. And, most importantly, it eases understanding. Using a sigmoid function, the final score is a number between 0 and 1. This value can be seen as a score, representing how likely the neural net is to be whatever output class has been labeled as 1; note, while similar to a probability, this score is not a probability. In general, values greater than 0.5 are assigned to class 1 and those less 0.5 are assigned to class 0. Though, the cutoff value can be changed and may represent a parameter worth investigating further for any given study. In general, it is much easier for humans to understand an output of 1 for true or 0 for false.

Other functions, such as the sign or softmax function, may be used in place of the sigmoid function. The sigmoid function, compared to the sign function, has the benefit of maintaining granularity when the outputs are near 0.

Evaluating Performance

While understanding how a NN functions is important, how are the weights determined? Before answering that, there needs to a be way to evaluate the performance of the NN (or any ML model). The most obvious way is to use accuracy: how many data points, when their features were fed to the NN, were assigned to their correct/ground truth class?

However, accuracy is not always the best measure. For example, consider the issue of a skewed data set. As the data set becomes more and more skewed, it becomes more accurate for the model to just blindly guess that any given data point belongs to the class with more samples. As such, more metrics need to be considered. The first step in doing so is defining a few terms: true positive, false positive, true negative, and false negative. These are shown in Table 2.1, which is known as a Confusion Matrix.

Table 2.1. Confusion Matrix [29]

		Actual/Ground Truth Class	
		Nominal (Positive	Faulty (Negative)
Predicted Class	Nominal (Positive)	True Positive	False Positive
	Faulty (Negative)	False Negative	True Negative

Along with being useful for granting insight, the confusion matrix allows for the definition of a number of new metrics, given by Equation 2.8 through Equation 2.12 [29].

Precision, Confidence, True Positive Accuracy	$\frac{True\ Positives}{True\ Positives + False\ Positives}$	Equation 2.8
Recall, Sensitivity, True Positive Rate (TPR)	$\frac{True\ Positives}{True\ Positives + False\ Negatives}$	Equation 2.9
False Positive Rate (FPR)	$\frac{False\ Positives}{False\ Positives + True\ Negatives}$	Equation 2.10
F1 score	$2\frac{Precision\ *\ Recall}{Precision +\ Recall}$	Equation 2.11
Accuracy	$\frac{True\ Positives + True\ Negatives}{Total\ Number\ of\ Samples}$	Equation 2.12

Another metric that is currently used is the receiver operator characteristic (ROC curve). This curve plots the true positive rate and the false positive rate, with each point corresponding to a different parameter; an example is seen in Figure 2.8. What this parameter is will be different for each project and represents any singular model parameter that can be tweaked to affect performance. Returning to prior discussion, the parameter used in this research is the cutoff for classes. Traditionally, an output score of 0.5 or higher results in the inputs being assigned as class 1. However, this value can be changed, and each distinct cutoff value results in a different true positive/false positive rate pair. These are plotted to form the ROC curve.

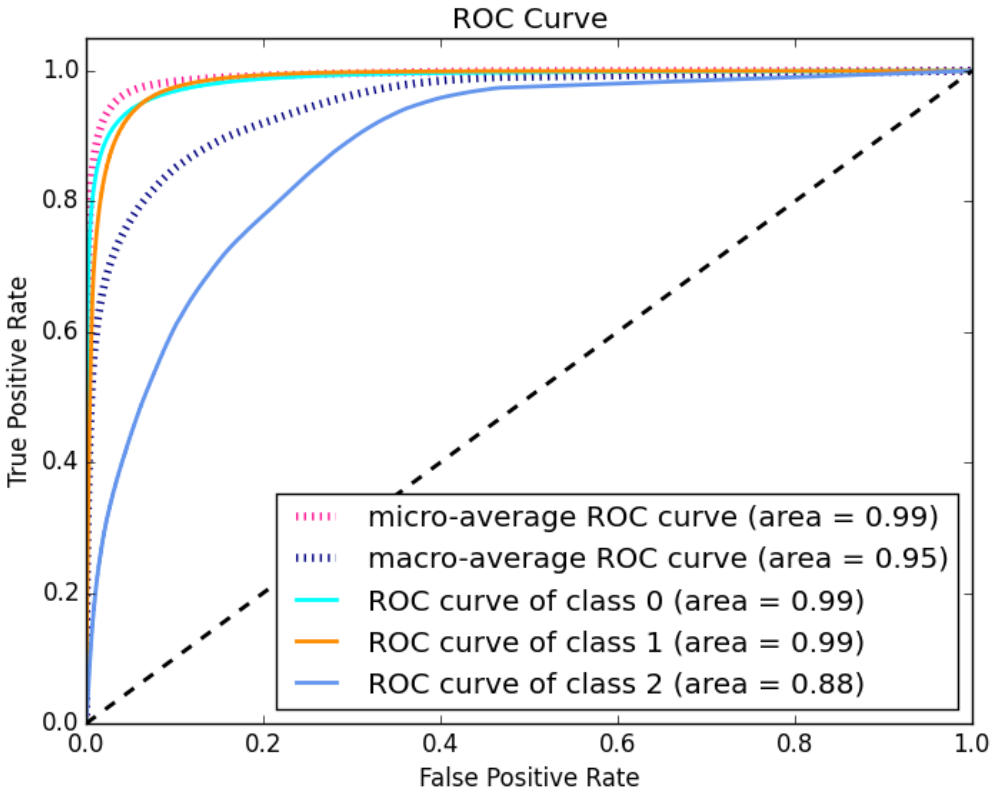


Figure 2.8. Example ROC Curve [30].

The ROC curve represents the tradeoff between correctly identifying samples as positive and blindly identifying them as positive. As the model starts to label more samples as positive—because of the parameter—the true positive rate will rise; eventually the model will label all inputs as positive, correctly identifying all positive examples and having a 100% true positive rate. However, in doing so, it also incorrectly labels all negative examples as positive, generating a high false positive rate, which is unfavorable. The ideal ROC curve jumps near immediately a true positive rate of 1, indicating there is a characteristic parameter that correctly identifies all positive samples as positive while not labeling negative examples as negative.

The final element to consider in model evaluation is what data to use. If the model is tested on the data used to train it, of course it will perform well. In fact, a common issue is in machine learning is that of overfitting, seen in Figure 2.9, in which the model is complex enough to individually account for all (or near all) training data point. However, this complexity means that when the model is applied to another data point, it is likely to guess wrong.

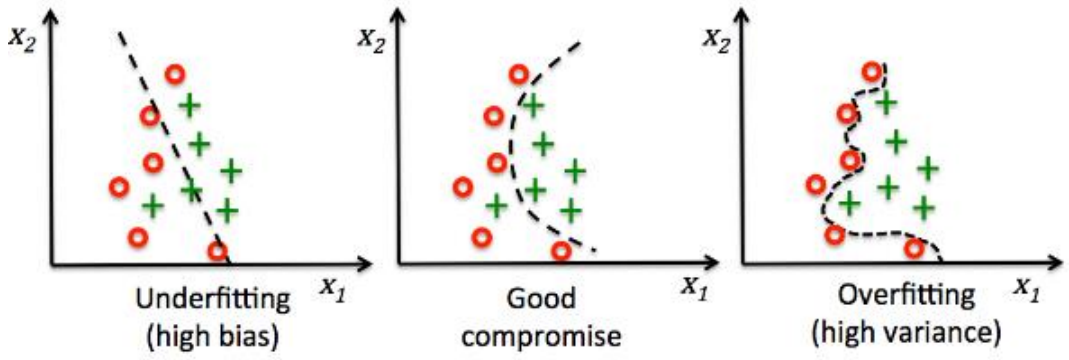


Figure 2.9. Visualization of overfitting

To avoid this problem, the data set is often broken up into three sets: training, testing, and validation. The training set is used to train the model; that is, the training set is the data used to calculate the parameters of any given model. The validation set is used to stop overfitting by indicating when to stop training. That is, the training algorithm, without a validation set, would run until it reaches a maximum level of performance. With a validation set, the training algorithm will check its performance against the validation set after each step of training. If the performance of the model when applied to the validation set decreases, it is an indication that the model is overfitting the data and that training should not continue. Last is the testing set, which is used to report the final performance of the model on a group of data that had nothing to do with training. The data is usually split such that the training set is ~70% of the data points, the validation set is ~15% of the data points, and the testing set is ~15% of the data points.

As with other aspects of machine learning, there are many methods of validating a model. For example, an alternative to the above methodology is to use m-fold cross validation. In this method, there is no testing set. Instead, the data is split into *m* disjoint sets. The model is trained *m* times, each time using a different single set of data as the validation set and the rest for training. The final performance is reported as the mean of the performance of the *m* models. For this work, validation will kept to simply breaking the data into training, testing, and validation sets [26].

Current Applications of Machine Learning to Additive Manufacturing

Current attempts at applying ML to AM largely use raw images (or similar) as the input to ML algorithms [31-33] for the purpose of identifying faults. The techniques of image processing are also commonly used to extract meaningful features from the data collected. Further, these methods are largely, in part due to the nature of machine learning, more in-situ than classical physics based techniques, which have largely focused on control of input parameters [21, 34]

Abdelrahman et. al used a set of 5 images of the powder bed, taken throughout the cycle, as the base for attempting in-situ fault detection. Intensity and gradient values were compared pair wise (5 images, 10 total pairs) to determine the assumed presence or absence of faults. These initial findings were then compared 3-dimensionally to the prior layer; a suspected fault that did not persist through two layers was discarded. The defects in question were manually added to the design of the part and confirmed by post-build CT scans; efforts were specifically focused on 0.05- 0.07 mm lack of fusion defects. This methodology made little to no use of machine learning algorithms. The results were largely positive, though the number of nominal layers skews the data. Further, there was a high rate of true positives [35]. These are two issues found in current work.

Gobert et. al, following this and working with some of the same researchers, used a similar set up, but taking 3 additional images of the powder bed (for a total of 8). Further, rather than relying on in-built defects, the image data was correlated to post-build CT scan data of the part. A gaussian kernel (of varying size corresponding to the size of standard defects) was used to extract the presence of faults in the CT data. The locations of these faults were then mapped to the image data using an affine transform and lining up known points (a technique utilized in current work). From there, features were extracted from the image data using, a 3D kernel method. A support vector machine was used to build the classifier. While performance was good using individual images, the 8 powder bed images were ultimately combined into an ensemble classifier with an accuracy of 85% [31].

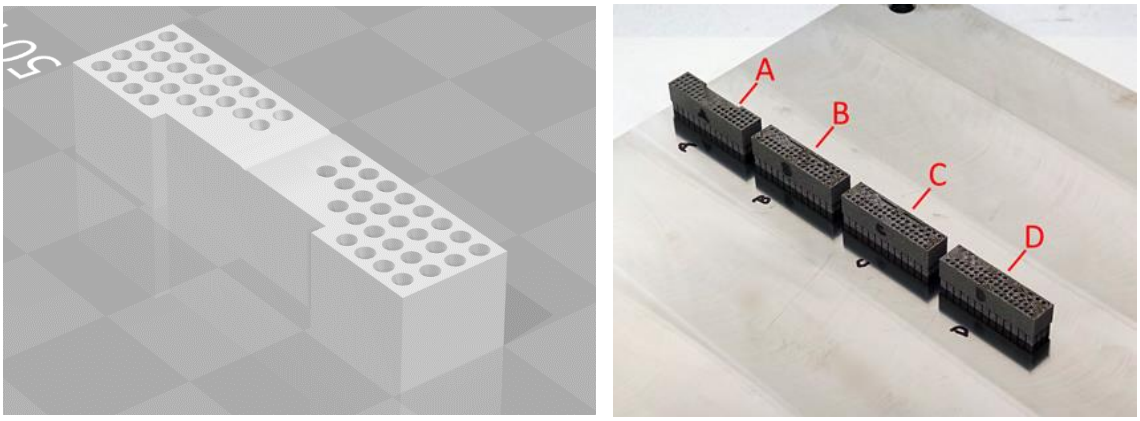
Scime and Beuth also employ in-situ optical imaging techniques, but have focused more on building a large database of fault examples to train their classifier. Further, this allows them to go beyond identifying regions as nominal or faulty, but identifying specific types of faults. At time of publishing, their database consists of “2402 image patches, composed of 1040 anomaly free patches, 264 recoater hopping patches, 228 recoater streaking patches, 187 debris patches, 314 super-elevation patches, 264 part failure patches, and 105 incomplete spreading patches” across multiple builds. Their training algorithm utilizes a bag of words approach. A given image patch is convolved with a number of filters to generate a response. These responses constitute the “word” for each pixel of that image; pixels can also be combined to form a single “word” for the image patch. These “words,” and the associated fault label for a given image patch, form a “dictionary” to which the word of new image patch can be compared. Of interesting note, and departing from other efforts, the methodology of this work is wholly 2-dimensional.

CHAPTER THREE
NEURAL NET TRAINING AND ASSOCIATED METHODS AND RESULTS

Experimental Methods and Procedures

Build Setup and Parameters

The part used for this study was a liquid-cooled heat exchanger (HeX) for cooling power-electronics components with specially designed conformal channels that would enhance the heat transfer effectiveness by wrapping the heat source with coolant channels. This HeX was designed with the intent of exploiting the unique capabilities of metal additive manufacturing, since channels that bend internally would be near impossible to fabricate using traditional methods; a 3D model of the part as well as the final build plate can be seen in Figure 3.1. The part was made from Fe_3Si powder using a Renishaw AM 250 machine at the Manufacturing Demonstration Facility at the Oak Ridge National Lab; Table 3.1 lists the parameters used for the build.



(a) (b)

Figure 3.1. (a) 3D model of the part, a heat exchanger with conformal channels (b) The finished build plate of the parts used for this research.

The raw powder bed profile height data was collected using a laser surface profilometer mounted on an in-house gantry system made to fit and work with the Renishaw machine. Once the machine had sintered a layer of powder but before spreading a new layer, the scanner moved over the area of interest, collecting and saving data in the process. In order to provide enough time for the scanner to complete the data collection process, a '0-power' or 'ghost part' was designed to be built in the unoccupied area inside the build envelope. As the laser stopped melting the HeX parts and moved to trace the ghost part using 0 laser power, the change in intensity in luminance was captured by a camera and used to trigger the scanning process. The process of scanning took about 30s. a timeline of the entire process for 1 layer can be seen in Figure 3.2.

It may be noted that the designed pause between each layer to accommodate the scanning process may affect the heat transfer and cooling rates throughout the part and result in a slightly different thermal interaction between layers. These effects are not considered since the delay created by the scanning process is not different from the delay that would be nominally incurred while manufacturing multiple parts in the same build.

Table 3.1. Build Parameters for Renishaw AM250

Region	Power (W)	Exposure time (μs)	Point-to-point distance (μm)
Core	200	110	75
Inner contour	160	110	75
Outer contour	110	80	40
Hatch distance (μm)	**Contour spacing (μm)**	**Layer thickness (μm)**	**Hatch rotation per layer (° clockwise)**
100	90	50	67

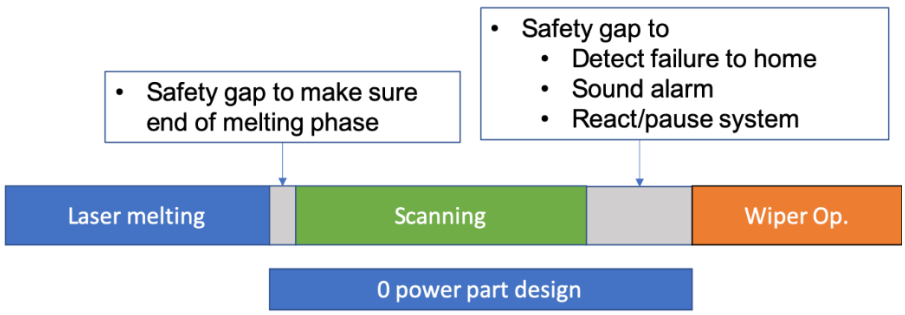


Figure 3.2. A schematic showing the timeline for scanning a single layer

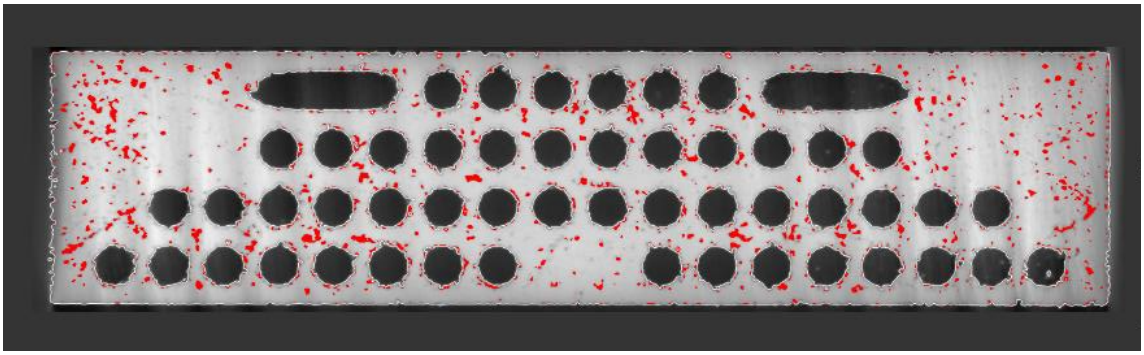
Due to limitations in width of the scannable area, two passes were made and the data later stitched together to generate a single set of data; for the build in Figure 3.1 specifically, two passes were made with 1 mm of overlap. Key details about the sensor are collected in Table 3.1.

The post-build CT scan data was created and delivered as a VGL project and analyzed using myVGL; this data was converted to .tif files for visualization and further analysis while full processing of the VGL project occurred separately. Details can be found at myVGL's website [36]. The resolution was 23.5 microns in each of x-y and z directions. An example of the output produced and used is seen in Figure 3.3. When used, the .tif files were cropped to just the part and converted to a binary stack using a simple threshold.

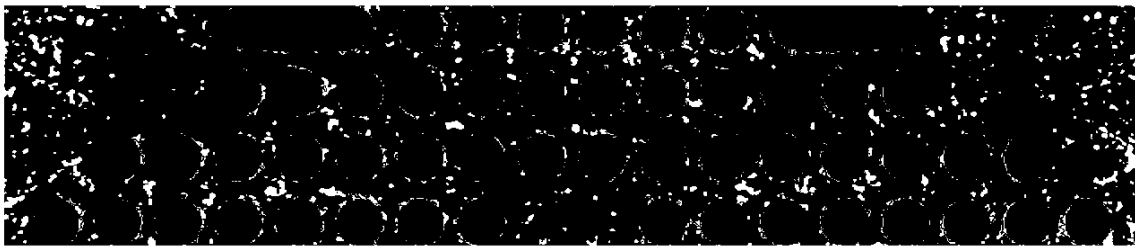
For further application, the VGL project was converted into a binary MATLAB array. The raw profilometer data files were similarly transitioned to a MATLAB array. This transition occurred during development of training and application techniques. As such, certain quantitative details like which profile layer or the exact cell resolution will vary. However, the general conclusions are still valid. The only potential discrepancy is that when working from the raw profilometer data, the data was mean-shifted to zero. This is not the case for the data when stored as a MATLAB variable.

Table 3.2. Resolution and accuracy of the laser surface profilometer

X, Y resolution	10 microns
Accuracy	0.2 microns



(a)



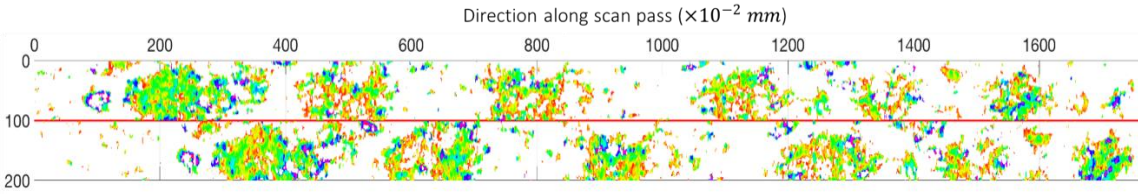
(b)

Figure 3.3. Example of Raw CT Scan Output as a (a) tif File used for primary development , and (b) as a MATLAB Binary Array

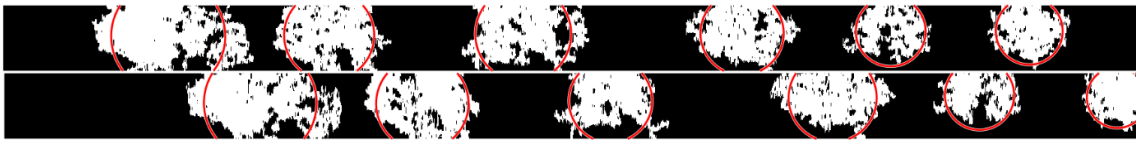
Data Preprocessing

Alignment of Sensor Passes

As mentioned in *Build Setup and Parameters*, the data was collected from a laser surface profilometer mounted on a 2-axes stage and completed in in two passes. As such, a number of preprocessing steps are needed to transform the data in a usable form. The first step in this process involves de-warping the data. For this experimental procedure, it was found that the roughness distribution of the sensor data varied qualitatively and quantitatively from the ground truth data (as determined manually using microscopy). To de-warp the data, a gaussian kernel was used to smooth and average the data retaining only the large order trend; this smoothed version of the data was then subtracted from the actual data to produce clean data that agreed well with the ground truth statistics. Further details about this process can be found in [37].



(a)



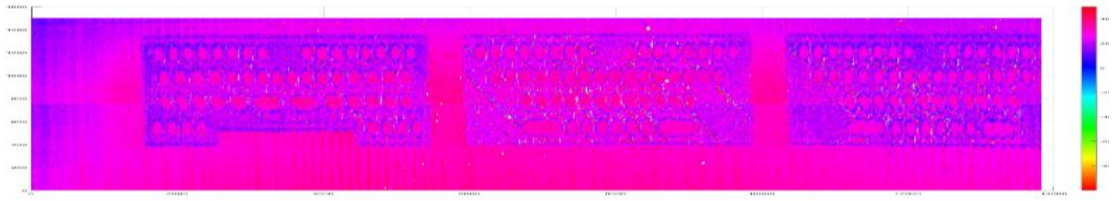
(b)

Figure 3.4: (a) Overlapping area of the forward and backward scans are arranged next to each other to demonstrate the shifting and distortion that have to be corrected in order to stich the two scans together. Here data is filtered to reveal only the higher levels of un-melted powder inside the channels. (b)Labelled contiguous regions are isolated based on connectedness and size for both scan passes. The centroid for each contiguous region is identified along with major and minor axes for best fit.

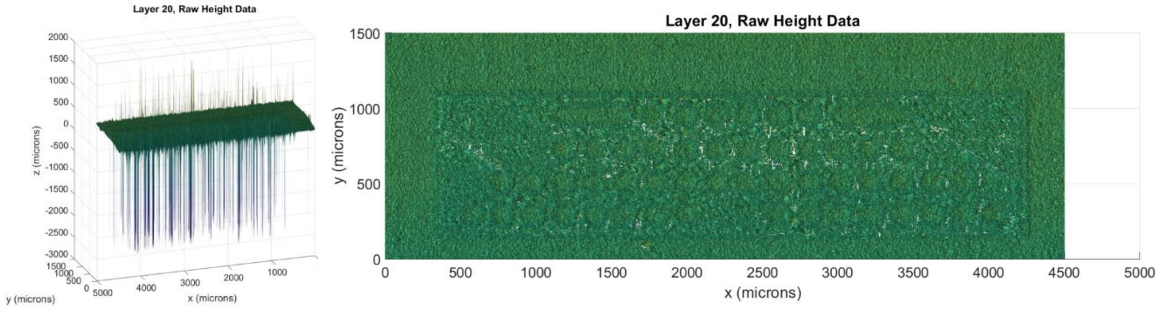
The next step in pre-processing was to align the two passes, which was a done via an affine transform between identical features found in the overlapping region of both scans (Figure 3.4), given as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_c \\ 0 & 1 & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$	Equation 3.1
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\alpha) & \sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$	Equation 3.2
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$	Equation 3.3

In the above equations, x and y denote the original coordinates, x' and y' denote the transformed coordinates x_c and y_c represent linear translations in the x and y direction, respectively, α is the angle of rotation, and s_x and s_y are scale factors. These various transforms can be applied in sequence to generate a single transform [38, 39]. The relevant coefficients were found that best aligned known features in the printed part. Specifically, referring again to the 3D model in Figure 3.1, several of the channels lay in the 1 mm overlap. Using the 3D model as a guide, the locations of the center of these channels were determined for each pass. Next, using MATLAB's image processing toolbox, the best transformation matrix was determined. An example of the data before and after it has been aligned and stitched together is shown in Figure 3.5.



(a)



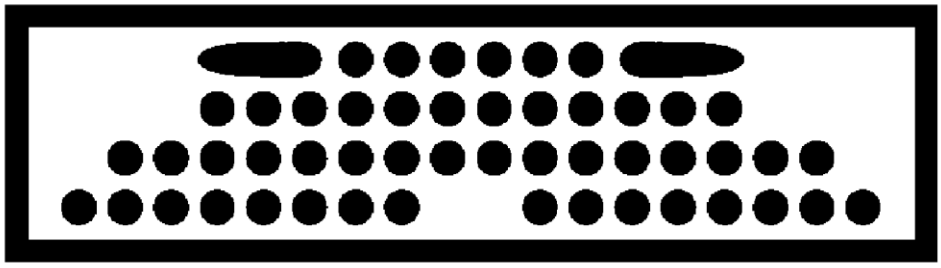
(b) (c)

Figure 3.5. Profile Data (a) of the entire build before allignment and a single part after allignment and filtering displayed as a (b) 3D isometric view and (c) colormap for a single layer

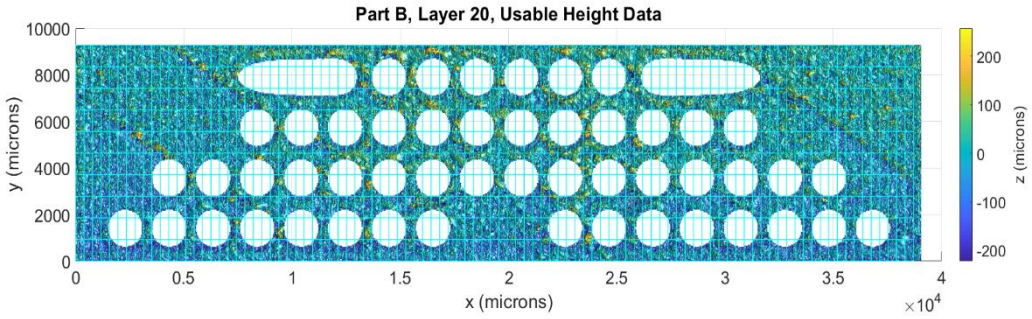
While this process could be repeated for each layer of data, it was found that the angle rotation was small enough to be negligible. Further, the x_c and y_c translation coefficients were found to be relatively consistent across layers. As such, when further processing the data, a simple translation (x and y shift) is applied to align scans from the two passes of the scanner.

Masking

Following alignment and stitching, the next step in preprocessing is removing the data representing the channel-regions of the HeX. This is important, since the data from the un-melted powder particles inside the conformal channels should not contribute to either training or testing of the machine learning algorithm. This is achieved by masking the profile data with a binary mask obtained from the corresponding slice of the part file. This mask marks the channels and boundaries of the part file by replacing the z-height data with 'NaNs' which can then be efficiently excluded from further calculations. The mask and resulting filtered data are shown in Figure 3.6.



(a)



(b)

Figure 3.6. (a) Mask Extracted from Part File (b) Usable Height Data, broken into cells

Mapping and Format Differences

For further application, the VGL project was converted into a binary MATLAB array. The raw profilometer data files were similarly transitioned to a MATLAB array. This transition occurred during development of training and application techniques. As such, certain quantitative details like which profile layer or the exact cell resolution (discussed in Chapter 3, Neural Networks, Basic Inputs and Parameters) will vary; layer 20 in the old format most likely is not layer 20 in the new format. However, the general conclusions are still valid. The only potential discrepancy is that when working from the raw profilometer data, the data was mean-shifted to zero; this is not the case for the data when stored as a MATLAB variable. This issue has been addressed and noted, and the basic effects considered.

Further, putting the data into a new format introduced a new challenge: mapping the CT data to the profile data. To generate the .tif files initially used, this mapping was done manually, matching a profile layer to its “closest” CT layer. When transferred to MATLAB arrays, all of the scanned CT layers were kept, meaning several CT layers mapped to a single profile layer. The work of mapping CT to profile data was kept in the form of a linear mapping function of CT layer to Profile Layer. This mapping would generate what profile layer a CT layer (or vice versa) would correspond to. As stated, the higher resolution CT scan would have multiple layers correspond to 1 profile layer. For example, for profile layer 20, there may be 5 CT layers that map to profile layer 19.6, 19.8, 20, 20.2 and 20.4. In this case, just using the CT layer that maps directly to profile layer 20 would be acceptable; however, in general the mapping is not that smooth and the surrounding CT scans would need to be averaged to get the “correct” CT output corresponding to the given profile layer. The same mapping could be done in reverse, finding where a profile layer would fall in the CT data.

Two solutions to this issue were established. The first used a 1D Gaussian Kernel, the equation of which is given in Equation 3.4. Using the scales of the CT scan data and the raw profile bed data, the number of CT scans, on average, in between profile layers is calculated. Next, a Gaussian Kernel of this size was constructed, as described by

$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(x-\mu/\sigma\right)^2}$	Equation 3.4

This equation would be discretized. The current profile data layer mapped to the CT domain would count as 0 or the average, μ. The x values would be the CT layers corresponding to that profile layer. Half of the CT scans corresponding to 1 layer of profile data above and the other half below the profile layer are considered. For example, consider profile layer 20 where 5 CT scans correspond to 1 layer of profile data. Profile layer 20 would map to, for example, CT layer 24.3. In this case, CT layers 22, 23, 24, 25, and 26 would be considered.

These values would then be used to calculate the weights, using Equation 4, for the surrounding CT layers. Using these weights, a weighted average is taken; if it is greater than 0.5, the transformed CT data would be considered faulty. Of final note is that the term $\frac{1}{\sigma\sqrt{2\pi}}$ is actually ignored when programming this. This term ensures that the sum of the weights is 1 and that the result of convolution is not artificially inflated or shrunk. However, that only works if an infinite (or near infinite) number of terms are considered. Using it the case of 5 terms may actually lower the magnitude of the response artificially. In this case, instead, once the weights are calculated, they are divided by their sum [39].

The second method for converting multiple CT layers to one layer corresponding to a layer of profile data is profoundly simpler, though less based in image processing. In this method, a simple voting approach is taken. Considering the same surrounding region as before, if a certain percentage (decided by the user) of them are faulty, the singular output CT layer is considered faulty. For the sake of ease, a majority of development was used using this method. The second method was developed as a more scientific option.

In general, training was done with the ORNL data using the older data format. The algorithms were updated to work with the new format, but less investigations were undertaken. All of the results concerning the Penn State data was found using the new data format.

Neural Networks

Basic Inputs and Parameters

The input to most Machine Learning algorithms and specifically an artificial neural net (ANN) is a large number of data points, each data point comprising of a vector of features. In this case, six measures of roughness calculated based on

the profile data serve as input feature vectors to the ANN and are defined by Equation 3.5 to Equation 3.9. These measures were taken from Keyence [2].

Roughness average:	$S_a = \frac{1}{n}\sum_{i=1}^{n} Z_i$	Equation 3.5
Root Mean Square Deviation:	$S_q = \sqrt{\frac{1}{n}\sum_{i=1}^{n} Z_i^2}$	Equation 3.6
Skewness:	$S_k = \frac{1}{RMS^3}\frac{1}{n}\sum_{i=1}^{n} Z_i^3$	Equation 3.7
Kurtosis:	$\mathrm{K} = \frac{1}{RMS^4}\frac{1}{n}\sum_{i=1}^{n} Z_i^4$	Equation 3.8
Maximum and Minimum Heights:	Z_{max} and Z_{min}	Equation 3.9

To transform the data into a large number of discrete input-output pairs suitable for ANN, a computationally efficient and intuitive quantization of the profile data is carried out by splitting the data into rectangular cells, illustrated in Figure 3.6 (b). The cell size (number of rows and columns) is one of the key design parameters affecting both efficiency and performance and is discussed more in the results, Chapter 4, Parametric Study.

During training, the input feature vector corresponding to each discrete cell is mapped to a label which classifies that cell as a member of a particular class. From a fault prediction perspective, each data point can be categorized as either nominal or faulty, corresponding to ground truth provided by CT data. The process of mapping CT data onto a relevant cell of the profile data is explained in the next section**.**

Processing of CT Scan Data to Provide Ground Truth

Training the neural net is essentially a process of optimizing the weights and biases to find the best parameter set (weights and bias values) that maps the inputs to the correct classes. These target classes, or ground truth data for the Neural Net are derived from CT scans of the part, carefully mapping the CT data to the corresponding $(x - y)$ location in the correct layer and querying the data for a fault.

The most straightforward way of converting this CT scan data into a large labelled training set would be to superimpose onto it the same cells used to define regions in the original profile data and treat each cell as an independent

observation, provided that the cell is not totally inside a channel. The CT scan data is used to determine whether each cell is counted as nominal or faulty. Specifically, the CT data is converted into a binary image via thresholding using in-built MATLAB commands. The channels are addressed separately, being treated the same as nominal regions until filtered out at a later step. This binary format allows for fault size and spatial information to be extracted and manipulated. An overview of this procedure is shown in Figure 3.7.

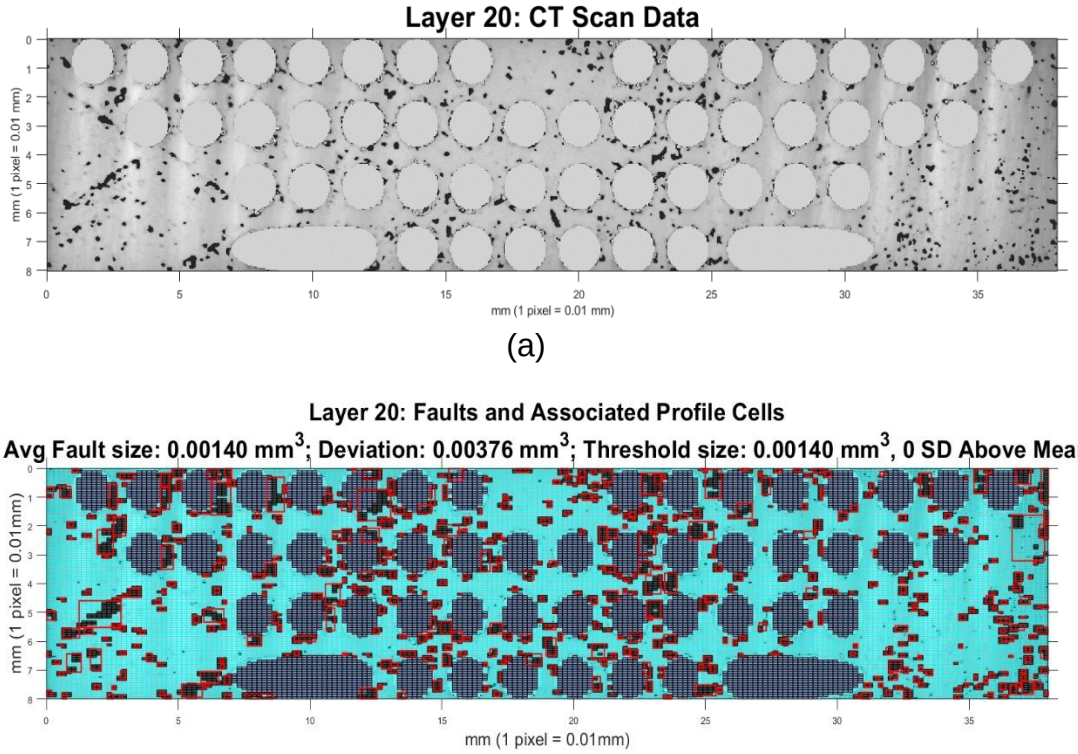


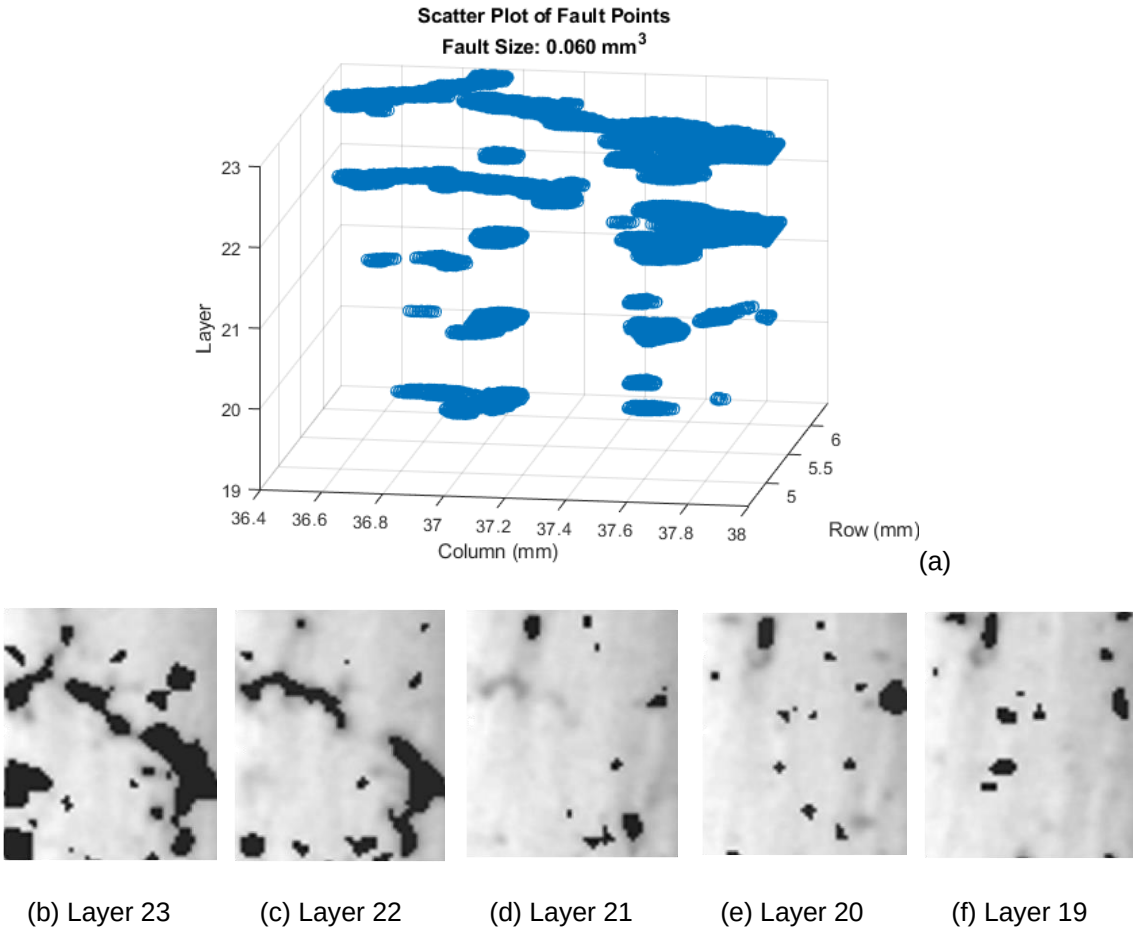
Figure 3.7. (a) Raw CT Scan Data imported to MATLAB and extracted to match profile data dimensions; faults are represented by black spots. (b) Grey cells are used to indicate cells that should be marked as faulty. Red bounding boxes indicate the edges of that fault on the given layer; note the apparent discontinuity in some regions. This is due to connectivity being determined 3-dimensionally.

However, from a close examination of Figure 3.7, it can be clearly seen that one fault typically spans multiple cells, especially when the cells are small. It was thus concluded that the rectangular cells superimposed by the grid structure did not fully capture the nature of a fault. To remedy this, a stitching algorithm was employed, which ensured that a single fault and a corresponding singular super cell were only considered once, as opposed to being artificially split up into multiple faulty cells. It is important for the reader to note that the faults also extend across layers and this z-axis span should be taken into consideration when the group-membership of faults is debated; this is shown via the red boxes in Figure 3.7 and is elaborated in the next section.

Stitching of Faulty Supercells

To facilitate robust automatic stitching of faulty super cells, the method of connected components from the field of image processing was used to first locate all of the faults and the associated pixels in the ground truth CT images. In this process, the CT data is used to generate a 3D stack of binary images, where the lower density associated with the faulty areas are represented as black spots, and are designated as “foreground”. From there, the connected components algorithm traverses the stack, labeling adjacent foreground pixels as being connected. Normally, the algorithm uses a recursive tree to resolve instances of adjacent components initially labeled as separate to ensure they are ultimately counted as one object [39].

The result of Connected Component Labeling (CCL) is an array of arrays. Each array represents one 3D object/fault found and provides the absolute indices of the pixels comprising that fault in the 3D matrix. This enables the grouping of faults that are connected in the x, y as well as z dimensions, as evidenced by a generic fault found in layers 19 to 23, and demonstrated in Figure 3.8. Figure 3.7(b) demonstrates this grouping for a given layer. It is important to note that the algorithm groups faults in three dimensions; faults that appear isolated in Layer 19 are actually contiguous due to connections across layers in the z-dimension.



(a)

(b) Layer 23 (c) Layer 22 (d) Layer 21 (e) Layer 20 (f) Layer 19

Figure 3.8. Example of extracted fault using the method of connected components.

Once the indexes of faults are extracted from the CT data, they can be used to stitch together more representative regions of the profile data. The roughness values for these stitched regions are calculated, using measures defined in Equation 3.5 through Equation 3.9, and used as input for the NN along with the roughness values for the nominal cells. It is also at this step that faults of size smaller than a threshold are ignored. While all voids and defects are undesirable, the largest commonly occurring faults spanning more than 50 microns in the CT scan are typically lack of fusion pores (50-500 microns) [22], which are the main kind of faults under investigation here. Gas pores (typically 5-20 microns) are also addressed, both by using a lower threshold and via a multi-class ANN classifier (Chapter 4), where the cell labels can be nominal, small faults or large faults. Because the size of faults will change based on how many layers are included in

the process, the threshold size must be relative. For this work, threshold size is defined as the number of standard deviations above the mean size.

Generation of inputs and outputs corresponding to these supercells can largely be broken into two steps. First, which cells are associated with which faults is established and recorded. Also, during this step, the faults are converted from absolute indices to x-y-z coordinates (if not already done so; MATLAB outputs absolute indices) and the outer bounds of a given fault for each layer are recorded. This positional information is useful for visualization and limits searching for other cells attached to a given fault to a localized area. This process is laid out in Algorithm 1.

The second step is to use this information to stitch areas corresponding to a fault together into a super cell and generate inputs and outputs for the NN. This is done by cycling through every cell and checking whether a fault was associated with it, established in step 1. If it is not associated with faults, the cell is marked as nominal and its roughness values are calculated using Equation 3.5 through Equation 3.9 and set as inputs for the NN. If it is associated with faults, other cells in a localized area (established by the bounds found in step 1) are checked for correspondence with the same fault. If they are associated with the same fault, the data for that cell is appended to the data of other cells associated with the fault. Once the data for all cells associated with a fault has been collated, metrics given by Equation 3.5 through Equation 3.9 are used to generate inputs to the NN; the corresponding output is marked as faulty. This process is laid out in Algorithm 2.

Table 3.3. Notations and Variable Names for Algorithms 1 and 2

Name	Definition	Maximum size of array
row, col	Maximum number of cells per row and column	n/a: input
layers	Maximum number of layers	n/a: input
r, c, L	Row, column, and layer index of profile cells, respectively	n/a
u,v	Row and column index of a pixel	n/a
X	Cell array of absolute indices of each fault	n/a: input
n	Number of faults	n/a: input
conLabel	List of which faults a cell is associated with	n-by-row-by-col-by-layers
faultsPos	x, y, and x (r, c, and L) indices of pixels in each fault	n-by-3-by-Number of Pixels in Each Fault
faultsBound	Index of the left, right, upper, and lower most pixels for a fault on each layer	4-by-n-by-layers
input	Array of roughness values for each NN input	6*row*col*layers
output	Corresponding array of truth labels for each input	2*row*col*layers
index	Index for counting number of input-output pairs	n/a
data	Array of profile data for region of interest	n/a

Algorithm 1. Associating Faults with Cells and Extracting x,y, and x coordinates

Input: Cell array X of absolute indices for each fault, dimensions of the raw profile data, row, col, and layers
Output: Arrays associating faults with cells: conLabel. Arrays indicating the x,y, x positions and outrebounds of faults: faultsPos, and faultsBound

2	**for all** faults in *X*			
3		**for all** pixels in current fault		
4			Calculate *u* and *v* and *r, c,* and *L* of pixel	
			Record u, v, and L in *faultsPos*	
5			Append current fault to list of faults for calculated cell in *conLabel*	
7			**if** *u* or *v* exceed current bounds for that fault for that layer	
8				Record *u, v* as new bound for fault on layer L in *faultsBound*

Algorithm 2. Use Associations Found to Stitch Cells into Super Cells

Input: Association of faults to Cells from Algorithm 1; conLabel. Outer bounds of fault: faultsBound
Output: Array of roughness values and corresponding ground truth labels to be used in training the NN: inputs, outputs

10	**for all** Profile Cells
11	**for all** faults associated with that cell
12	**if** no faults
13	Output[index] ← nominal
14	*Input[index]* ← evaluated using Eq. 4-8 for current cell
15	index ← index + 1
16	**else if** Fault has not been stitched on that layer yet
17	Initialize *data*
18	**for** cells in range of leftmost to rightmost for current fault (from *faultsBound)*
19	**for** cells in range of uppermost to bottommost for current fault (from *faultsBound)*
21	**if** current fault is associated with current cell
22	append region profile data to *data*
23	Mark Current fault as being stitched for that Layer
24	Output[index] ← faulty
25	*Input[index)* ← evaluated using Eq. 4-8 for *data*
26	index ← index + 1

There are two final notes to be made in regards to cell stitching. First is that the nominal cells are not stitched. The roughness values, for nominal examples, are generated from simple rectangular regions of profile data. Examining the values and ranges of the roughness values for input and outputs reveals no systematic error that would separate the two. This is evidenced in Figure 3.9.

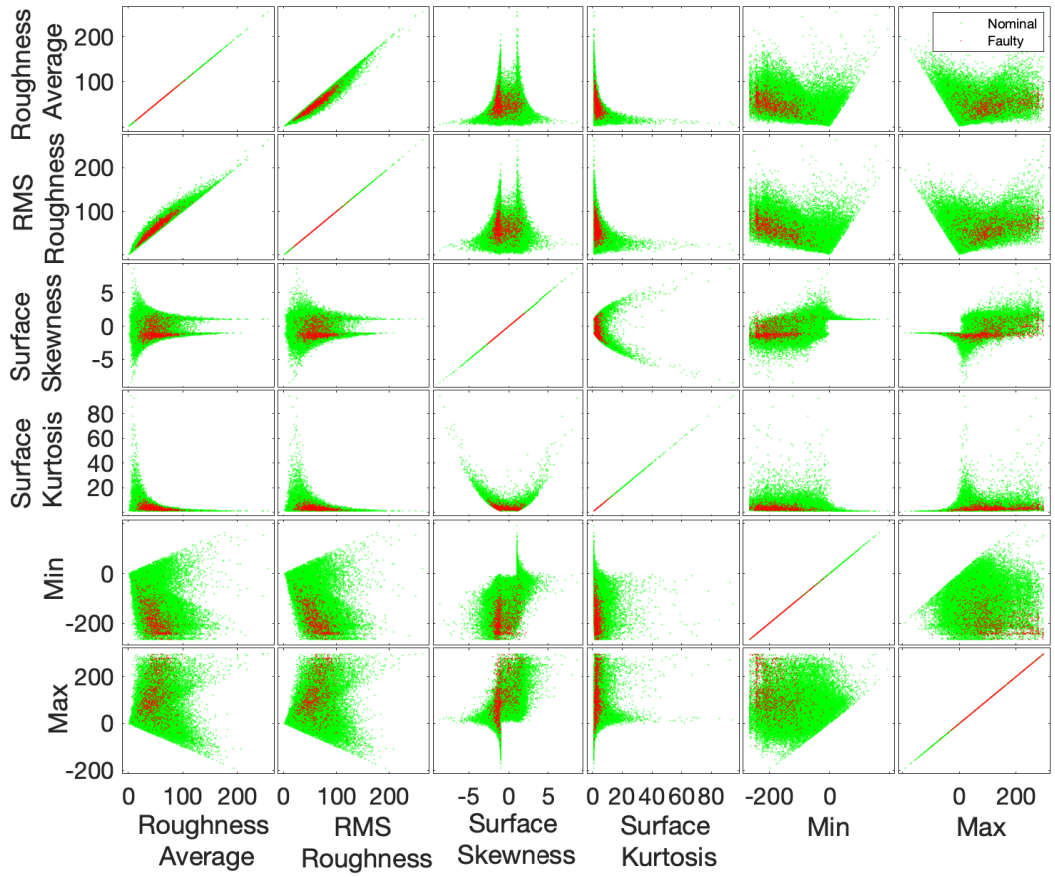


Figure 3.9. Scatter plot of data points generated using Algorithm 1. Note that there is little to no apparent, systematic separation between nominal and faulty data.

Of course, physical differences likely still exist between the powder profiles of nominal and faulty regions, and ideally a NN will be able to find them. The key here is to note that pre-processing the data as is done here does not introduce any systematic differences.

Further, Figure 3.9 grants some insight into the physical nature of powder profiles associated with faults and nominal regions. Perhaps the most apparent difference is that faulty regions tend to be characterized by the presence of high peaks and low valleys. Faults also tend towards lower skewness; however, further testing revealed this not to be a specific case for these layers and parameters, but rather a general trend. Interesting, the overall trend of skewness tends to a dual peak distribution. By definition, a skewness of 1 (or -1) indicated a uniform distribution (negative merely indicates the height values themselves were negative); a skewness of 0 indicates a roughly equal amount of powder above and below the man. Figure 3.10 highlights some of these trends by looking specifically at the associated probability distribution functions. Each pdf also includes the results of a Two-sample Kolmogorov-Smirnov test for whether two samples are from the same distribution. For these examples, faults were not thresholded based on size (as discussed in Chapter 3, Stitching of Faulty Supercells) for demonstration purposes, as no thresholding means more faulty cells to draw data from. However, it also means this may not be the most representative distribution. The trends hold for higher thresholding, though.

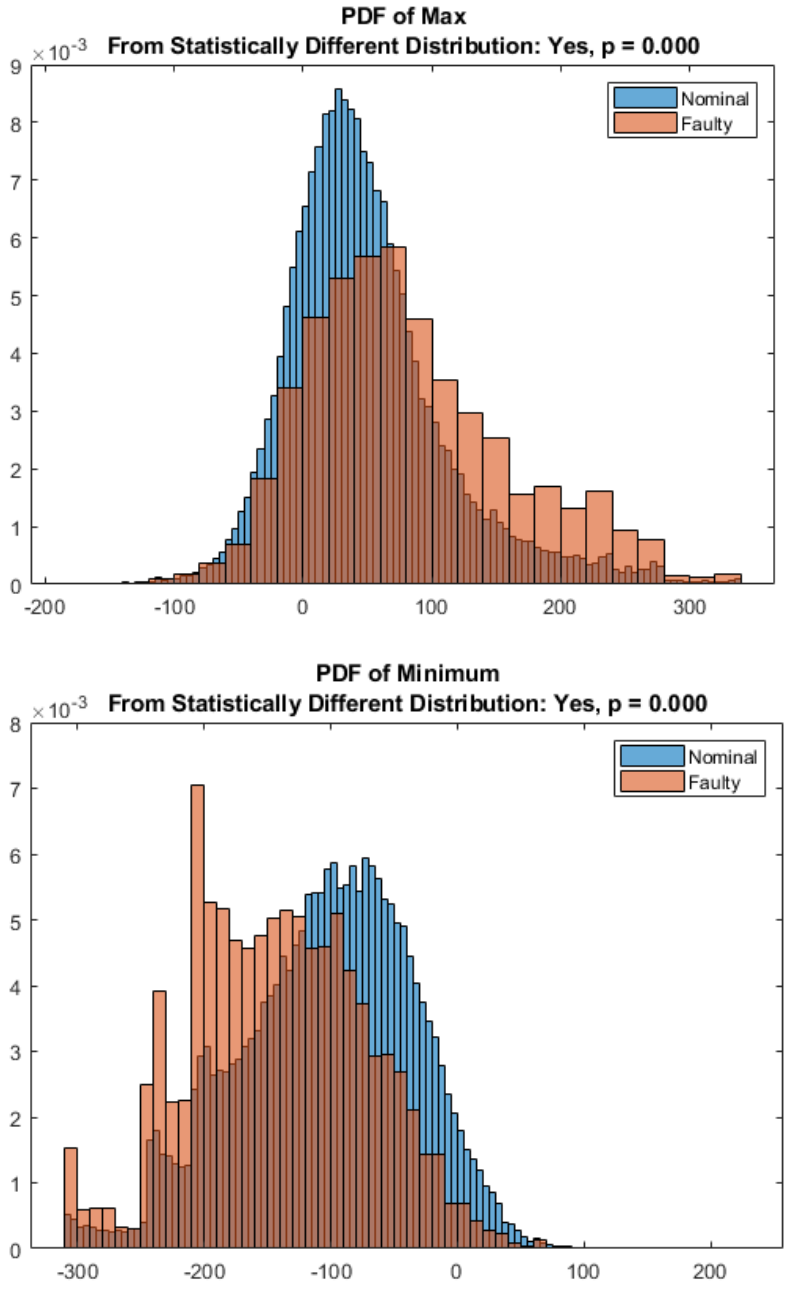
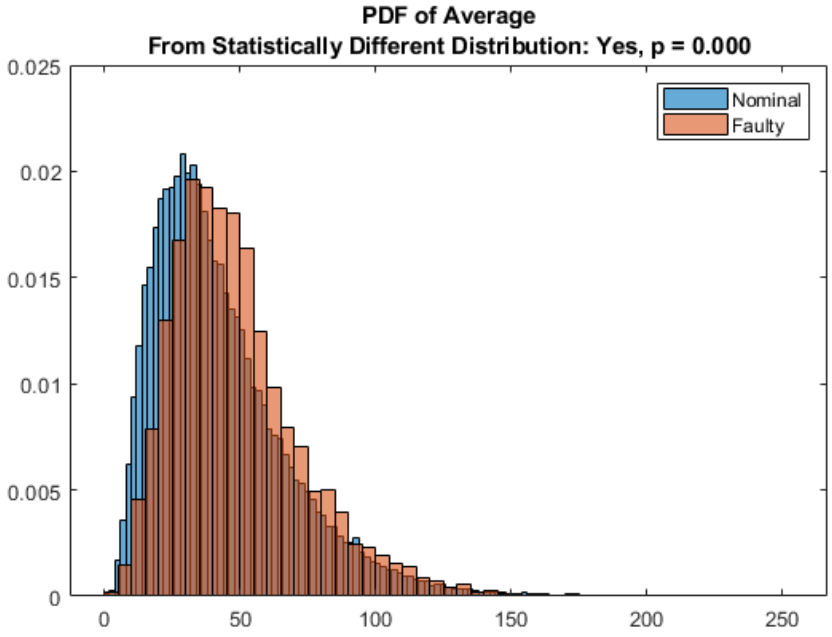


Figure 3.10. Histograms showing interesting trends in the height distributions for nominal and faulty cells



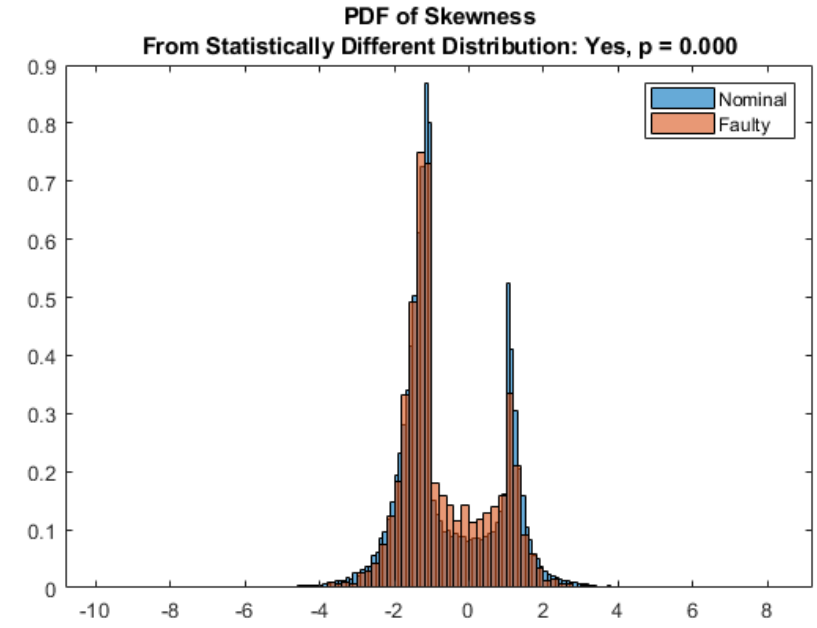
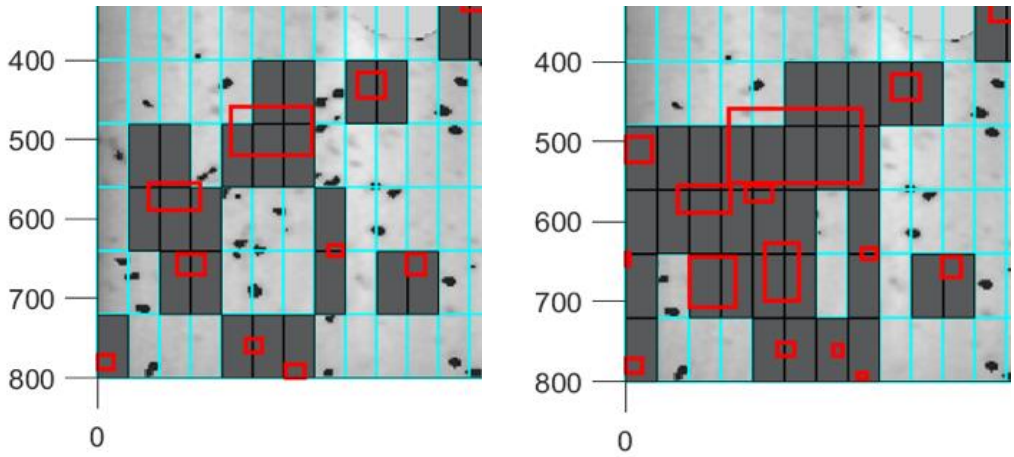


Figure 3.10 continued

The general trends of faults having higher peaks and lower valleys is clearly demonstrated. Further, looking at the values for the average roughness, peaks in the powder bed appear to be slightly more prevalent in faults than valleys.

What is interesting is that the skewness of faulty super cells appears to follow the same distribution as nominal cells. The Kolmogorov-Smirnov test indicates that this is not the case, however, likely due to the central tendency of the faulty cells. As mentioned, skewness of 1 or -1 indicates a roughly uniform distribution. The higher central tendency (towards 0) of the faulty cells indicates that for faulty regions, there is more likely an equal amount of powder above and below the mean height. Of note is that a negative skewness is favored for faulty and nominal regions; this is likely a characteristic of the Renishaw Machine itself. Further, these plots ultimately suggest that the raw average height of a cell is the most important characteristic in determining whether it is faulty or nominal.

Of final note in regard to stitching and pre-processing, as the algorithm is searching for 3-dimensional fault objects, which and how many layers are included is inherently important. While a fault may manifest as a single pixel on one layer, it could expand into a larger fault on the next. This is demonstrated in Figure 3.11Figure 3.11, which shows the growth of a stitched super cell by including more layers in the CCL algorithm. This suggests that including more layers as opposed to less is likely ideal. Though, too many layers and too fine a profile grid may raise issues with limitations in computing power.



(a) (b)

Figure 3.11. Demonstration of how number of layers affects detected faults: (a) 2 layers (b) 5 layers

Neural Net Training and Parameters

Once inputs and the outputs to the map are all calculated, the NN is initialized and trained. The network used for this pattern recognition problem is a two-layer feedforward network, with a sigmoid transfer function in the hidden layer, and a softmax transfer function in the output layer. The number of hidden neurons was set to 40 after performing a small-scale parameter sensitivity study. The number of output neurons was set to 2, to represent this binary classification problem (classification between normal and faulty cells). Bayesian regularization back-propagation is used to train the network.

The key parameters studied for generating results are cell size, number of layers, and fault size thresholding. The cell size is dictated by how many pixels of the raw profile data are used in a single cell and is discussed. Smaller cell size means the region covered by a fault can be more accurately isolated. The concept of the number of layers and why it is important is also discussed in Chapter 3, Stitching of Faulty Supercells. Thresholding refers to the number of standard deviations above the mean size below which “faults” will not be counted as faults, as discussed in Chapter 3, Stitching of Faulty Supercells.

The classification accuracy of the ANN will be measured using standard metrics defined in Chapter Two, Machine Learning Overview, Evaluating Performance. The equations and example images are reproduce here for convenience.

Precision = $\frac{True\ Positives}{True\ Positives+False\ Positives}$	Equation 3.10
Recall, Sensitivity, True Positive Rate (TPR) = $\frac{True\ Positives}{True\ Positives+False\ Negatives}$	Equation 3.11
False Positive Rate (FPR) = $\frac{False\ Positives}{False\ Positives+True\ Negatives}$	Equation 3.12
F1 score = $2\frac{Precision * Recall}{Precision+ Recall}$	Equation 3.13
Accuracy = $\frac{True\ Positives+True\ Negatives}{Total\ Number\ of\ Samples}$	Equation 3.14

Here, true positive refers to the number of samples the NN correctly predicted as being nominal. False positive is the number of faulty samples the NN incorrectly identified as nominal. True Negative represent faulty cells correctly identified and False Negative represent nominal cells misclassified as faulty by the

Neural Net [29]. Traditionally, these values are presented in a confusion matrix, an example of which is seen in Table 3.4

Table 3.4. Confusion Matrix

		Actual	
		Nominal (Positive)	Faulty (Negative)
Predicted	Nominal (Positive)	True Positive	False Positive
	Faulty (Negative)	False Negative	True Negative

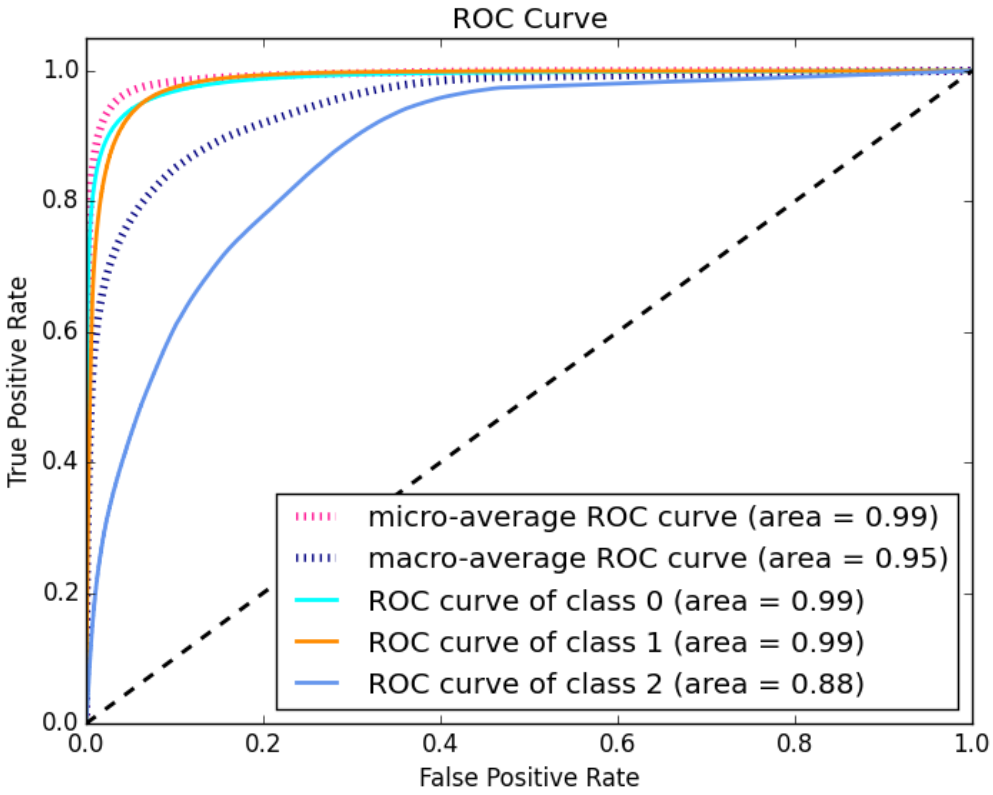


Figure 3.12. Example ROC Curve [30].

CHAPTER FOUR

RESULTS AND DISCUSSION OF TRAINING

Parametric Study

Initial tests were performed to ensure the algorithms and training were running properly. Upon ensuring that the methodology was implemented correctly and producing promising results, a parametric study was performed. As mentioned in Chapter 3

Neural Net Training and Parameters, the primary variables changed to evaluate the performance of the NN are cell resolution, the number of layers tested, and the threshold value, represented as the number of standard deviations above the mean. For the parametric study, these variables were varied in accordance with Table 4.1.

Table 4.1. Test Matrix for Parametric Study

Parameter	**Values**
Threshold	0 to 3 Standard Deviations above Average Fault Size
Cell Row Resolution	8, 10, 16, 20, 25, 32, 40, 50, and 80 Pixels per cell, y direction
Cell Column Resolution	19, 38, 50, 76, and 100 Pixels per cell, x direction
Number of Layers	2-11 layers

Initial tests showed that thresholding were valuable and the average size was chosen as a reasonable minimum value. 3 was chosen as the upper bound because, assuming a normal distribution, 49.85% of all faults would be between the average size and 3 standard deviations above it.

The resolution values were chosen out of necessity. The original data, in pixels, had certain dimensions. The create consistent rectangular cells, only factors of those values could be used. The original imported and cropped data was 800-by-3800 pixels.

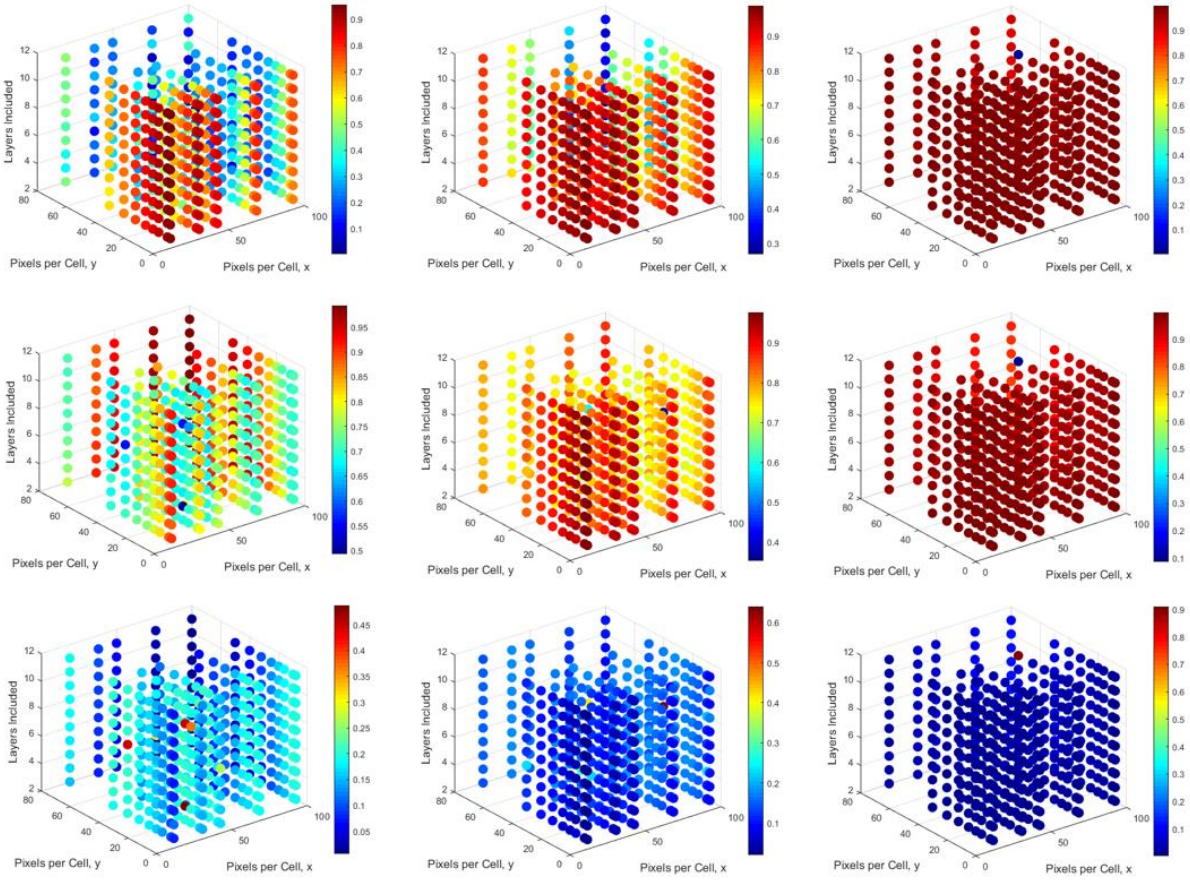


Figure 4.1. Results of Parametric study, displayed as a colormap. Rows: F1 Score, Accuracy, Performance. Columns: No Threshold, Threshold as Average Size, Threshold as 3 SD above mean.

The lower bound on the number of layers was chosen for precisely that reason. Since part of the advantage of stitching as performed here was the 3 dimensionality of it, 2 layers is the minimum. Initial tests used only up to 5 layers, due to limitations in computing power and the focus being on debugging and gathering initial results. 10 was chosen as a reasonable extension for starting to truly understand the effects of including multiple layers. The results of the parametric study are summarized in Figure 4.1.

Here, the performance metrics chosen are displayed as a colormap. For accuracy and F1 score, dark red indicates a higher score and better performance. For the mean-squared error, dark blue indicates a lower score and better performance. In all cases, the best cases occur in the bottom, forward corner. This

corresponds to finer resolution grid, with less pixels per cell in both the x and y direction. Further, the scores generally improve as the threshold is increased; specifically, all scores for the highest level of thresholding performed are excellent. This result is not unexpected, since a finer grid enables only relevant local data to be included from around a fault and a higher threshold implies that only relatively large faults are treated as faults which can be more easily correlated to the physical roughness parameters from the corresponding cell. Looking specifically, the highest accuracy found was 99.8% using a cell resolution of 8-by-19 pixels across 3 layers and thresholding at 3 standard deviations above the mean size. The highest F1 score found was 0.9990, also using a cell resolution of 8-by-19 pixels across 3 layers and thresholding at 3 standard deviations above the mean size. The lowest mean-squared error was 0.0017, using a 10-by-19 cell resolution across 3 layers and thresholding at 3 standard deviations above the mean size.

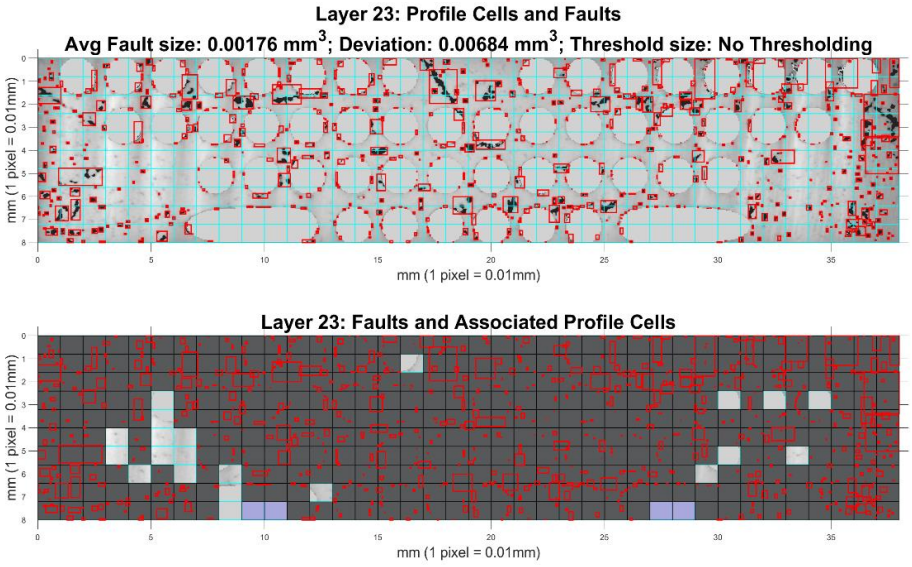
Looking closer, 2 more interesting results are noticeable. First, it is noted that more layers included generally results in slightly better performance for lower thresholds; this trend fails when a higher threshold is applied. From a physical perspective, treating a singular, multi-layer fault as just that, singular, is understandable. However, the layer-wise nature of AM should be remembered. As such, it is reasonable to claim that performance is linked less to how many layers are included and more to how well truly faulty regions are identified. In the case of small cells, this is done by precise extraction of relevant surface roughness data; for large cells, this is done by better thresholding faulty regions using 3-dimensional data.

Further, it is reasonable to assume that a fault initiated on one layer invites faults in following layers [32]. As such, it is possible that small faults on one layer, that themselves are not particularly exemplar of a fault, could propagate in larger faults. At a low layer count, ignoring these regions must done via thresholding. Hence, the trend of more layers yielding better performance breaks down for greater thresholds. For lower thresholds, these faults would be included if a single layer was considered; by considering the data in 3-dimensions, they can be removed.

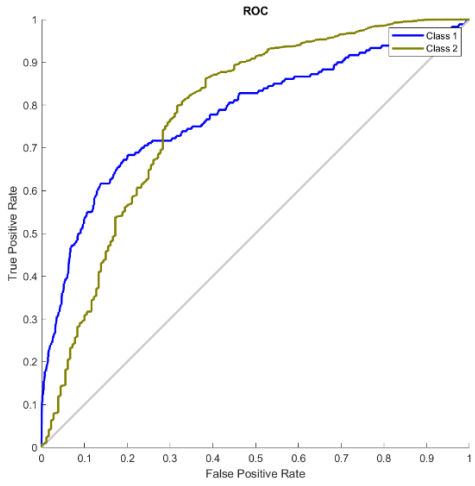
In conclusion for this point, and considering the data further, the most important elements are cell size and threshold level. Referring again to the parametric results, while the number of layers included does affect performance, the difference it causes is not as significant as the difference caused by cell size and threshold level.

The next interesting trend is that while smaller cells and higher thresholds generally result in better performance, this is not always the case. Figure 4.1 shows that accuracy and performance are also high for low thresholds and large cells. Examining this trend more closely, this is likely the result of an excess of faulty examples, as shown in Figure 4.2. These results are a good example of why

other metrics, like the F1-score, are important to consider; factors relating to how the data is processed may result in arbitrarily high accuracy or low error but affect other metrics less.



(a)



(b)

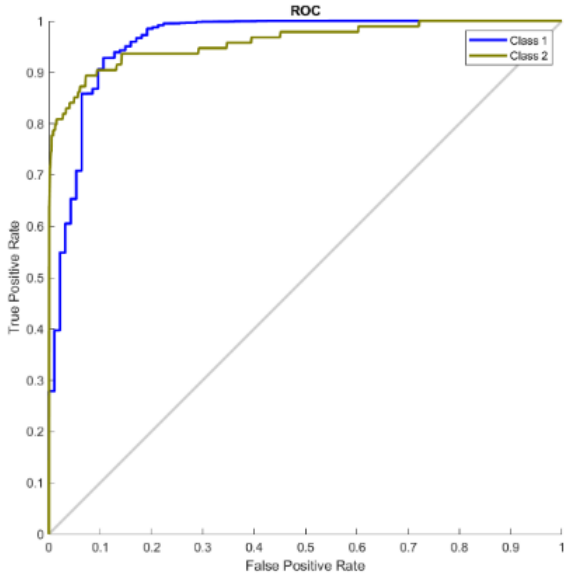
Figure 4.2. High Cell Resolution across 11 layers. (a) Example output of stitching for Layer 23 (b) ROC curve

Table 4.2. Confusion Matrix for High Cell Resolution

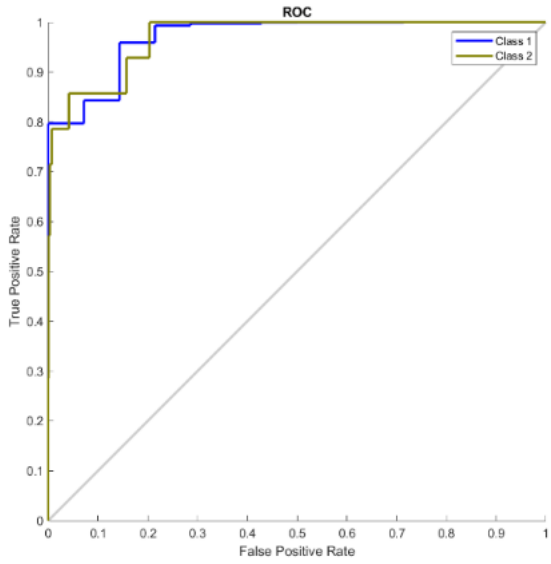
		Actual	
		Nominal (Positive)	Faulty (Negative)
Predicted	Nominal (Positive)	15 0.1%	3 0.0%
	Faulty (Negative)	165 1.4%	11540 98.4%

Training Results

Following the parametric study, training was performed using the best parameters found; 8-by-19 cell resolution across 3 layers with thresholding at 3 deviations above the mean. These results are displayed in Figure 4.3and Table 4.3.

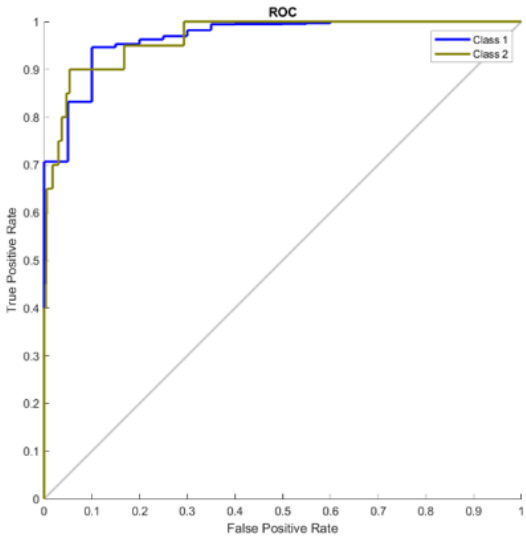


(a)

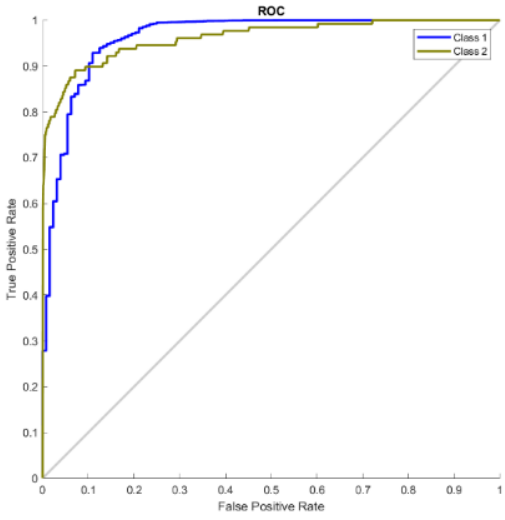


(b)

Figure 4.3. ROC curve for (a) training set, (b) validation set, (c) testing set, and (d) all of the data



(c)



(d)

Figure 4.3 continued

Table 4.3. Confusion matrices for (a) training set, (b) validation set, (c) testing set, and (d) all of the data

		Actual	
		Nominal (Positive)	Faulty (Negative)
Predicted	Nominal (Positive)	27918 99.7%	45 0.2%
	Faulty (Negative)	2 0.0%	49 0.2%

(a)

		Actual	
		Nominal (Positive)	Faulty (Negative)
Predicted	Nominal (Positive)	5986 99.7%	7 0.1%
	Faulty (Negative)	3 0.0%	7 0.1%

(b)

		Actual	
		Nominal (Positive	Faulty (Negative)
Predicted	Nominal (Positive)	5982 99.7%	12 0.2%
	Faulty (Negative)	1 0.0%	8 0.1%

(c)

		Actual	
		Nominal (Positive	Faulty (Negative)
Predicted	Nominal (Positive)	39886 99.7%	64 0.2%
	Faulty (Negative)	6 0.0%	64 0.2%

(d)

These results, in general indicate excellent performance of the NN. For these parameters, the accuracy is 99.83%, the F1 score is 0.9991, and the mean-square error is 0.0016. Of note is that, visually examining the ROC curves, these are not inherently better than the initial values yielded when testing over 5 layers. Referring again to Figure 4.1, it is noted that the NN performed well regardless of parameters chosen when a higher threshold is set.

Despite excellent scores, however, one problem is evident: the prevalence of false positives. These are examples that were faulty but classified as nominal. This trend is noted for most parameter combinations. Part of the problem may be due to the inherent nature of fault detection. NNs and machine learning in general can break down when the number of positive (or negative) examples fair outweigh the negative examples. This trend usually occurs because with so few negative examples, just labeling all examples are positive often results in the best solution. However, that the NN was able to correctly identify roughly the same proportion of faults as faults suggests that some boundary was found that performs slightly better than just assuming the part is completely nominal.

Upon further investigation, it was found that part of the problem originates with the way MATLAB trains the NN. Specifically, given the structure of the NN, MATLAB defaulted to not using a validation set. This setting was overridden for some tests. Upon further investigation and comparison to prior results, it was found that doing so exacerbated the problem of false positives. However, forgoing a validation set does not completely solve the problem, with there still being a high number of false positives. While forgoing a validation set may raise concerns about the ability of the NN itself, the results of the testing set still demonstrate some level of generalizability.

Table 4.4. Comparison of test set results using a NN that was trained a) with a validation set and b) without a validation set. Cells were 8-by-19 pixels, with a threshold for faults of 3 SD above the mean across 5 layers.

		Actual	
		Nominal (Positive)	Faulty (Negative)
Predicted	Nominal (Positive)	9765 99.5%	24 0.2%
	Faulty (Negative)	6 0.1%	17 0.2%

(a)

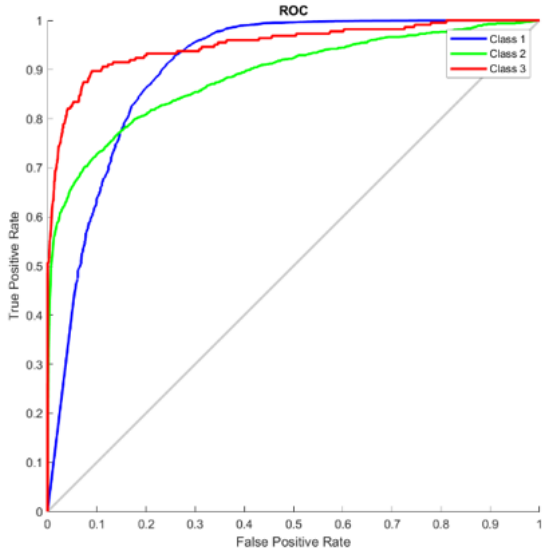
		Actual	
		Nominal (Positive)	Faulty (Negative)
Predicted	Nominal (Positive)	9754 99.4%	18 0.2%
	Faulty (Negative)	17 0.2%	23 0.2%

(b)

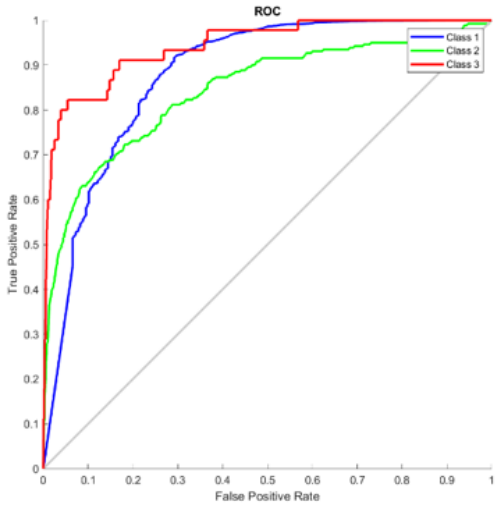
Further, it was found that thresholding is extremely important for avoiding false positives, with the number of false positives increasing as the threshold decreased.

Multiclass Classification

Last, attempts were made to extend the algorithm to a multi-class system. In these attempts, the stitching and thresholding were applied the same. However, instead of indicating simply whether a region was faulty or nominal, faulty regions were broken into sub-categories: large and small, each defined by a threshold. This simulation was performed without a validation set, following the realization that doing so results in a lower instance of false positives, discussed in Chapter 4 Training Results.

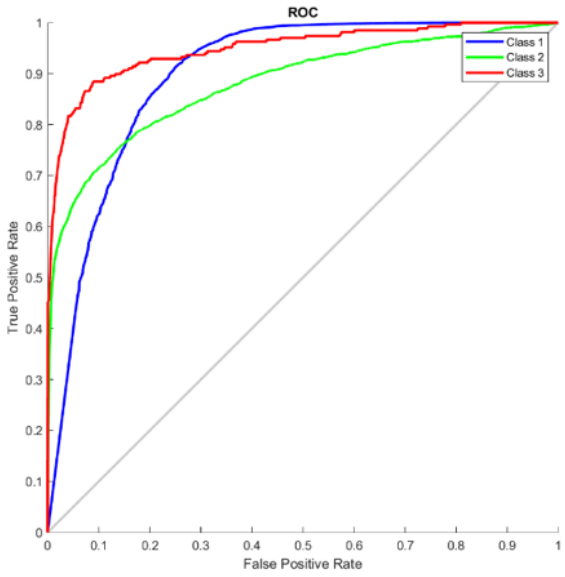


(a)



(b)

Figure 4.4. Multiclass ROC curve for (a) training set, (b) testing set, and (c) all of the data



(c)

Figure 5.4 continued.

Table 4.5. Multiclass Confusion matrices for (a) training set, (b) testing set, and (c) all of the data

		Actual		
		Nominal	Faulty; small	Faulty; large
Predicted	Nominal	46492 95.9%	908 1.9%	42 0.1%
	Faulty; small	152 0.3%	701 1.4%	68 0.1%
	Faulty; large	4 0.0%	15 0.0%	113 0.2%

(a)

		Actual		
		Nominal	Faulty; small	Faulty; large
Predicted	Nominal	8193 95.7%	171 2.0%	10 0.1%
	Faulty; small	58 0.7%	76 0.9%	27 0.3%
	Faulty; large	2 0.0%	13 0.2%	8 0.1%

(b)

		Actual		
		Nominal	Faulty; small	Faulty; large
Predicted	Nominal	54685 95.8%	1079 1.9%	52 0.1%
	Faulty; small	210 0.4%	777 1.4%	95 0.2%
	Faulty; large	6 0.0%	28 0.0%	121 0.2%

(c)

These results were generated using 3 layers with a cell resolution of 8-by-19. Class 1 is nominal cells. Class 2 are faults that are between 0 and 3 standard deviations above the average fault size. Class 3 is all faults above 3 standard

deviations above the average fault size. As can be seen in the ROC curve, the results once again appear good. The ability to discern nominal from faulty regions, represented by the curve for Class 1, is visually on par with a 2-class system. The ability to distinguish between different degrees of fault is excellent, but not as excellent. Further, the problem of a skewed data set and correctly identifying faults, originally acknowledged in the prior section, remains and appears to worsen.

CHAPTER FIVE

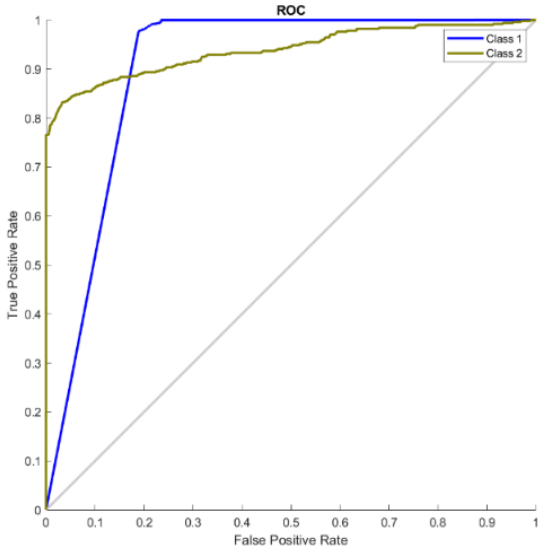
NEURAL NET TRAINING AND ASSOCIATED METHODS AND RESULTS

Neural Nets Used for Application Efforts

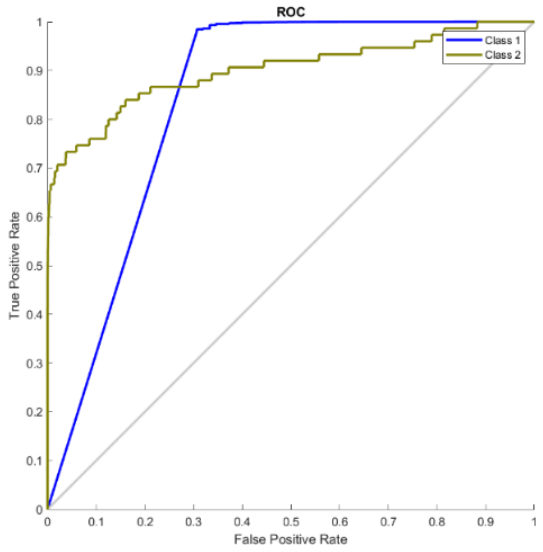
Oak Ridge National Lab (ORNL) Data and Neural Net

This section shows the results, in terms of ROC curve and confusion matrices, used when applying the NN. While any Neural Net may be used, these were specifically chosen, the reasons for which will be discussed after each set of results. One net was taken from each of the ORNL and Penn State and will thus be denoted NN-ORNL and NN-PS, respectively.

For training the NNs, Class 1 or positive is taken as a nominal example. Class 2 or negative is taken as a faulty example.

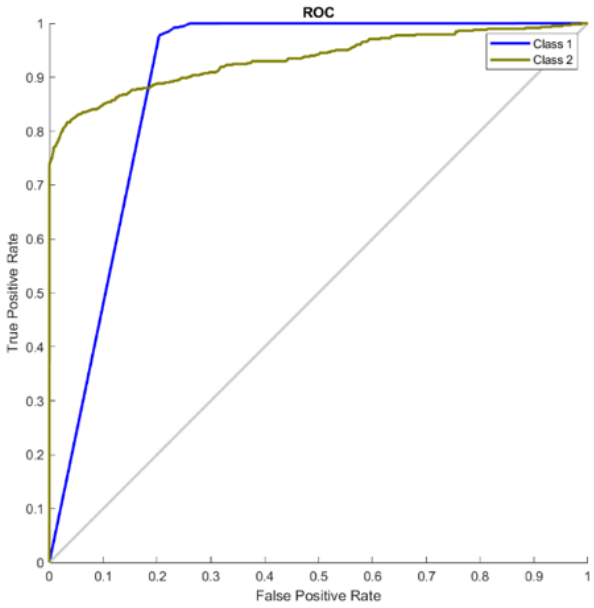


(a)



(b)

Figure 5.1. ROC Curves for NN-ORNL used in application. (a) training set (b) testing set (c) all samples.



(c)

Figure 5.1 continued

Table 5.1. Confusion Matrices for NN-ORNL used in application. (a) training set (b) testing set (c) all samples.

		Actual	
		Nominal (Positive)	Faulty (Negative)
Predicted	Nominal (Positive)	110683 99.5%	119 0.1%
	Faulty (Negative)	10 0.0%	389 0.3%

(a)

		Actual	
		Nominal (Positive)	Faulty (Negative)
Predicted	Nominal (Positive)	19534 99.5%	35 0.2%
	Faulty (Negative)	15 0.1%	40 0.2%

(b)

		Actual	
		Nominal (Positive)	Faulty (Negative)
Predicted	Nominal (Positive)	130217 99.5%	154 0.1%
	Faulty (Negative)	25 0.0%	429 0.3%

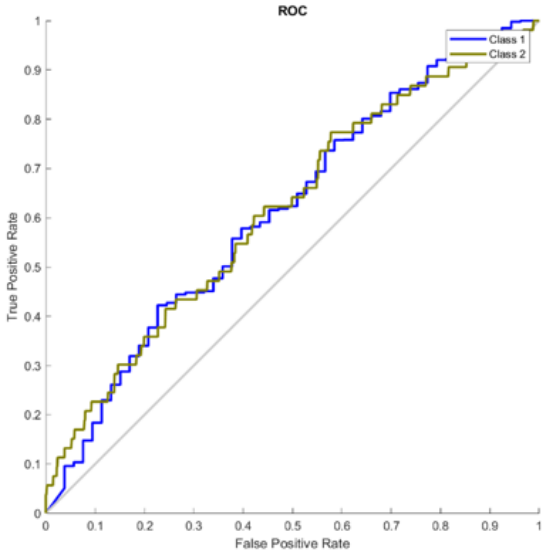
(c)

This Neural Net was trained on Layers 29 to 38 of HeX B from ORNL. The cells were 18-by-9 pixels and the threshold was set at 3 standard deviations above the mean. The parameters for cell resolution and fault size threshold were chosen because they produced the best results. In regards to the choice of layers, initial testing had been performed starting at layer 19 and attempts at application would

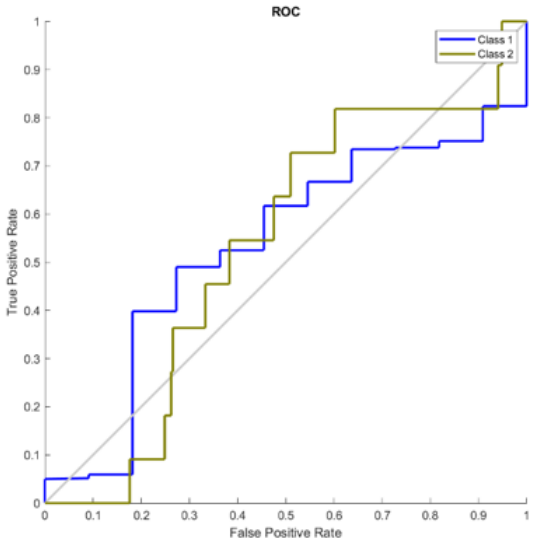
also begin there. As such, a different set of layers was chosen to simulate applying a NN trained on one set of data to another.

Of note is the prevalence of false positives in the testing set. However, the NN was still able to identify these regions correctly as faults more often than not, a trend reflected in other trained nets. Further, the ROC curve demonstrates good results for both the True and False Positive Rate, indicating that there is some threshold for the output of NN that results in the NN correctly classifying these regions as faults, though at the cost of labeling more nominal regions as faulty. This NN was trained using the older data format.

Penn State Data and Neural Net

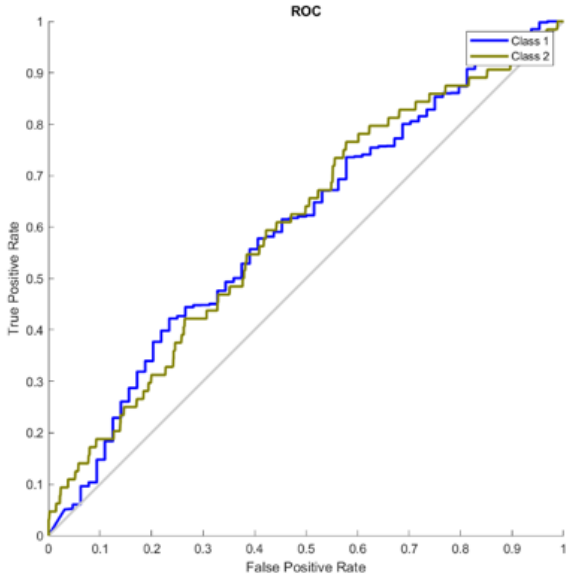


(a)



(b)

Figure 5.2. ROC Curves for NN-PS used in application. (a) training set (b) testing set (c) all samples.



(c)

Figure 5.2 continued

Table 5.2. Confusion Matrices for NN-PS used in application. (a) training set (b) testing set (c) all samples.

		Actual	
		Nominal (Positive)	Faulty (Negative)
Predicted	Nominal (Positive)	109488 100.0%	52 0.0%
	Faulty (Negative)	0 0.0%	1 0.0%

(a)

		Actual	
		Nominal (Positive)	Faulty (Negative)
Predicted	Nominal (Positive)	19320 99.9%	11 0.1%
	Faulty (Negative)	0 0.0%	0 0.0%

(b)

		Actual	
		Nominal (Positive)	Faulty (Negative)
Predicted	Nominal (Positive)	128808 100.0%	63 0.0%
	Faulty (Negative)	0 0.0%	1 0.0%

(c)

This Neural Net was trained on Layers 148 to 152 of HeX A from Penn State. The cells were 9-by-12 pixels and the threshold was set at 0 standard deviations above the mean. Of note is the prevalence of false positives in the testing set. The parameters for cell resolution were chosen because they produced the best results. As for, the choice of layers, the Penn state data was of much

higher quality, with few faults. Querying the data revealed that the layers chosen had a significant amount of faults in comparison to the rest of the part, allowing for more faulty samples. The threshold size was chosen for similar reasons; while a higher threshold generally proved to yield better results, too high of a threshold in this case would greatly reduce the number of faulty samples.

This particular combination as chosen on the merit of its ROC curve. While not particularly good, especially compared to the ROC curve for the ORNL part, it is positive. Specifically, it was better than the ROC curves for a few other combinations of parameters and layers around these. The quality of the ROC curve was used as a metric because of what it represents. A ROC curve plots the true positive rate versus the true negative rate, for some parameter. In this case, that parameter is the threshold at which a score for that class leads to the example being labeled as that class. Normally, for a two-class system, a cutoff of 0.5 is used; this would result in the seen confusion matrices. However, the given confusion matrices (and thus the standard cutoff of 0.5) demonstrate poor ability to discern faulty regions from nominal. The positive ROC curve demonstrates that while true, there is some cutoff that can be chosen that will better identify faults, albeit at the cost of more nominal regions incorrectly identified as faults. Further, the idea of thresholds for the NN can be carried over into application. Theoretically, positive results here will result in positive application results.

Last, a neural net trained without stitching the cells into super cells was considered as a baseline control, but bore no substantial results.

Of final note is that a majority of the development of these application algorithms was performed using the ORNL data and the older data format. The use of Penn State data (both for a trained NN and in application) is included more for reference, to test generalizability, and to ensure that the algorithms were correctly mapped to work with the new data format.

Of final note, the parameters for reading in the data (cell size, threshold, etc.) were set to the same as the parameters used in generating its corresponding NN; i.e. the PS part used a threshold of 0 and a cell size of 9-by-12. This was done for consistency.

Application Results

Baseline Results

With the NN trained, the next step in the process was to apply it and see if it could adequately identify faults using the in-situ data. In this, a problem arose; namely, since was testing potential in-situ application, the grouping used prior, which was based on the post-build CT data was not available. Instead, the in-situ data was divided into rectangular grid cells; each cell was passed to the NN and labeled as nominal or faulty. The results of this can be seen in Figure 5.3 and Table 5.3. Confusion Matrix for application of NN-PS to the PS part. As this was application, the data was not subset into training, validation, and testing sets.

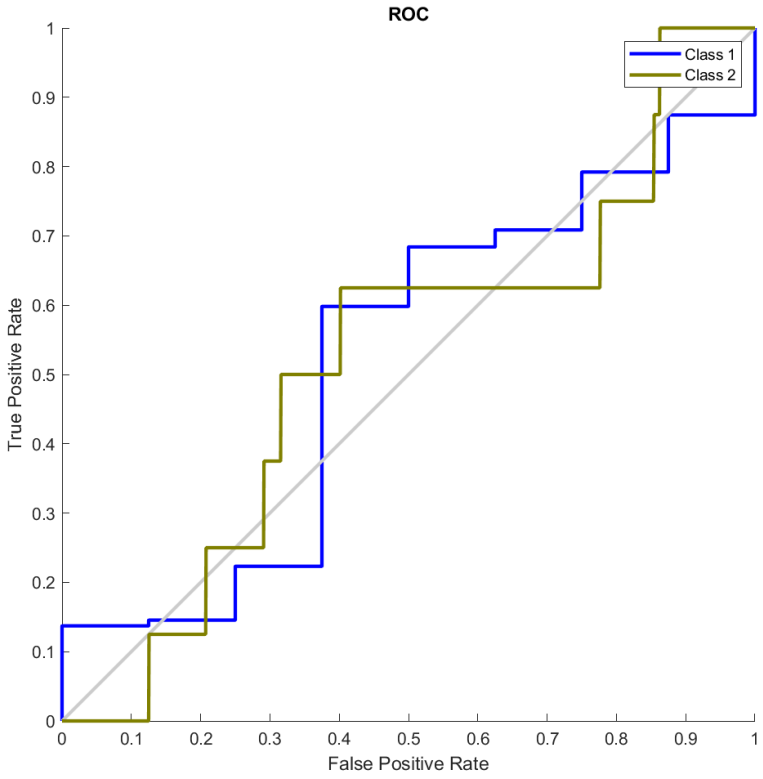


Figure 5.3. ROC Curves for for application of NN-PS to the PS part. As this was application, the data was not subset into training, validation, and testing sets.

Table 5.3. Confusion Matrix for application of NN-PS to the PS part. As this was application, the data was not subset into training, validation, and testing sets.

		Actual	
		Nominal (Positive)	Faulty (Negative)
Predicted	Nominal (Positive)	72440 47.0%	3 0.0%
	Faulty (Negative)	81612 53.0%	5 0.0%

As can be seen, the NN had trouble identifying faults, despite excellent training results. Further, the NN had a tendency to overly classify cells as being faulty. Baseline results using the NN-ORNL were even worse than shown. As stated, this is likely due to the lack of grouping. Ultimately, something would have to be done to group the in-situ data.

Manual faults Investigation

Upon recognizing that naive application of the NN on rectangular grid cells produced poor results, a more intuitive approach was taken to try and glean some insight out the data has to how to better approach the problem. To this end, the profile cell coordinates of several faults in the base part were manually identified. From there, the roughness values and output scores of the NN were considered. Initial results of these investigations are provided in Figure 5.4 and Table 5.4. Manually Found Fault Scores for Layer 21 of a fault in ORNL HeX B Cell by Cell. Table 5.4 represents the score output by the NN, representing how faulty the data is likely to be, for each individual cell in a fault for a given layer. In this table, the numbers are laid so as to match the orientation of the fault; for reference, an image of the fault is provided in Figure 5.4. After this, Table 5.5 shows the roughness values of each individual cell as well as the roughness values when the entire region is stitched together. Last, Table 5.6 displays the fault scores output by the NN when the cells comprising the fault are stitched together and the data fed to the NN, as well as the minimum and maximum value for that region on that layer; the purpose of including this data will be discussed.

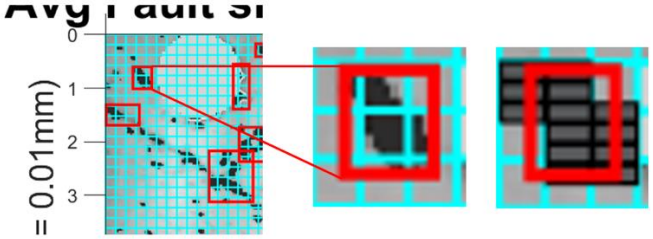


Figure 5.4. Close-up of investigated fault

Table 5.4. Manually Found Fault Scores for Layer 21 of a fault in ORNL HeX B Cell by Cell

Using NN-ORNL **Combined Score:** 1.00				**Using NN-PS** **Combined Score:** 1.00		
4.3246e-119	4.4347e-64			1	1	
7.732e-73	2.2699e-81			1	1	
	4.8886e-105	4.7263e-60		1	0.9912	
	7.5386e-49	9.1657e-55			1	1
	1.1732e-75	4.4494e-69			1	1
	9.0095e-59	5.2438e-31			1	1

Table 5.5. Surface Parameters (equations 1-5)

Combined Parameters: 75.97 (Ra), 89.24 (Rq), -1.392 (Skewness), 2.391 (Kurtosis), -221.9 (min), 146.6 (max)		
120.5446 123.5479 -1.05634 1.149275 -159.675 62.72513	37.33317 59.17184 -2.48487 7.366808 -210.625 17.67513	
100.4107 105.2912 -1.08273 1.398696 -167.975 82.37513	93.95552 119.9879 -1.44275 2.198237 -220.425 16.82513	
	100.2239 118.0228 -1.38586 2.100782 -221.925 2.625129	50.28383 61.17711 -0.16101 2.561206 -112.175 146.6251
	70.57752 83.47339 -1.38056 2.286838 -221.775 59.02513	86.42816 88.43491 -1.06289 1.16898 -139.075 -23.9749
	69.76142 80.09699 -1.45055 2.603619 -206.425 29.27513	75.56862 80.62339 -1.25176 1.855739 -204.425 -47.6749

Table 5.6. Manually Found Fault Scores of a fault in ORNL HeX A by Layer when Considering Entire Fault as Region

Using NN-ORNL			**Using NN-PS**		
Layer	**Fault Score**		**Layer**	**Fault Score**	**Minimum and Maximum**
19	1.138e-05		19	4.7832e-15	-237.3 226.5
20	1		20	3.0056e-6	-221.9 146.6
21	1		21	1	-209.3 171.1
22	0.999		22	0.3262	-230.9 217.0
23	1		23	1	-206.6 95.22

Examining this data, it was noted that the presence or absence of faults was characterized by the presence or absence of extreme powder bed heights in the profile. This is concurrent with findings demonstrated in Figure 3.10. Further, examining the stitched roughness values for each layer confirm this trend. Of note are the differences between the results from the PS and ORNL NN. The PS-NN was able to identify individual cells quite well, yet still produced poor results when blindly applied to cells. Further examination, represented in Figure 5.5, illustrates this point; the PS-NN was labeling everything as faulty. This is understandable, given how faulty the ORNL part is compared to the Penn State Part.

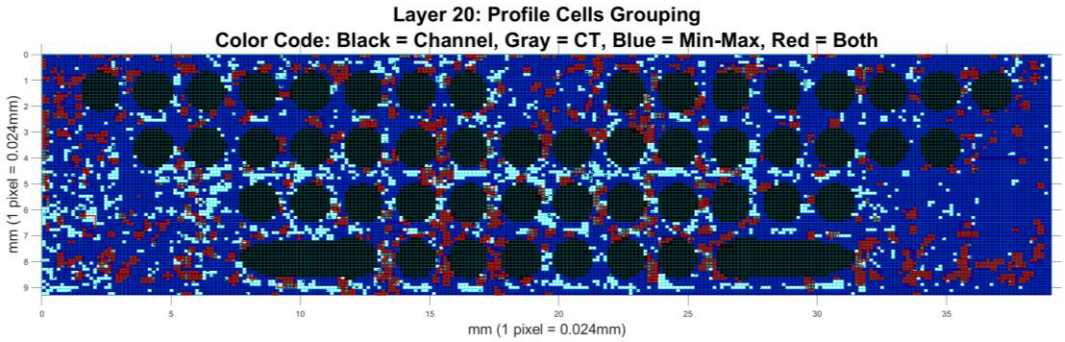


Figure 5.5. Application of NN-PS to ORNLL HeX-B

This effect illustrates the limitations of NNs to the situations from which there training data was drawn.

The success, especially when considered stitched cells, illustrates the potential for some form of identification or stitching algorithm based around the presence of extreme peaks and valleys.

Min-Max Stitching Algorithm and Impact of Using Neural Net

Given the trend that faults tend to have higher peaks and lower valleys, with an emphasis on the combination of the two, attempts were made to apriori stitch the data based on the presence of local extrema.

Specifically, an algorithm was developed for just that purpose. First, the data is queried for peaks and valleys in the profile bed data. These extrema are then thresholded. Next, a local region, defined by a search radius, for each local maximum is checked. If a local minimum is found within this region, it is marked as a single rectangular region with the extrema as the corners. Once all regions have been marked at a pixel-by-pixel level, they are transferred to a profile cell level. Finally, adjacent marked cells are combined into super cells, similar to during training; for this specific project, the Method of Connected Components is used. The NN is applied to these super cells. This process is summarized by Algorithm 3.

Algorithm 3. Stitching of in-situ data based on the presence of adjacent extrema

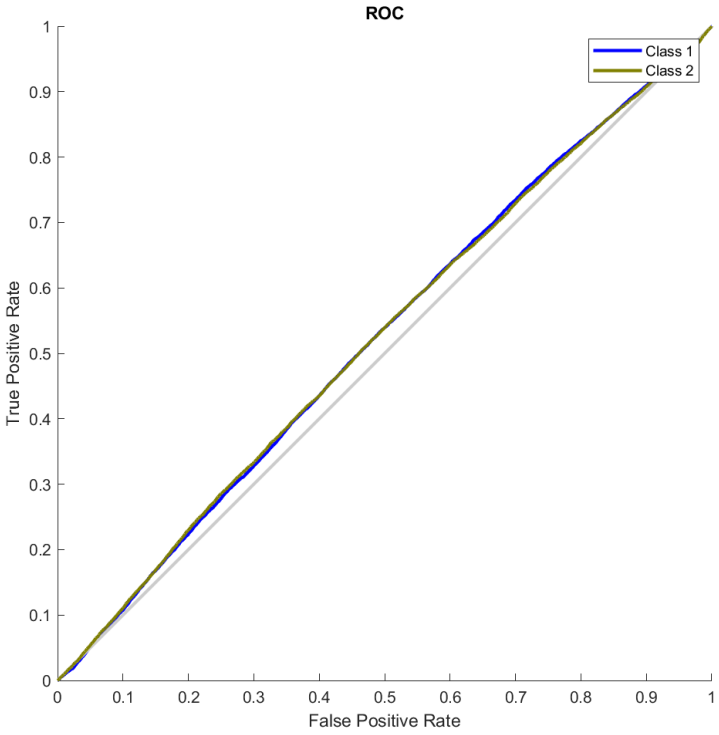
Input: Profile Data
Output: MATLAB Object detailing the location of regions in which there were adjacent extrema.

1	Find Local Extrema in Profile Data
2	Threshold Local Extrema
3	For *Each Layer*
4	Extract Locations of Mins on that Layer
5	For *Each Max on Current Layer*
6	Search for Mins, by comparing to extracted locations in a Localized Region, defined by a radius
7	If *Min within Local Region*
8	Mark Rectangular Region with Extrema as Corner
9	For *Each Marked Region*
10	Label Corresponding Profile Cells
11	For *Each Layer*
12	Use Method of Connected Components to Combine Cells into Super Cells

The key parameters for this algorithm, beyond cell size, are the threshold for the local extrema and the search radius. For local extrema, most algorithms for finding points that are higher or lower than surrounding points. However, this means even values close to 0 can be flagged as peaks and valleys worth checking. To avoid this, local extrema are thresholded; these thresholds determine how high or low an extrema has to be to actually be considered an extrema. The radius refers to how far around a found peak to search for a valley (or vice versa). The effect of these parameters was not extensively tested. Rudimentary exploration revealed that setting the threshold as roughly 150 to 200 above and below the average height worked best. Generally, the search radius was set as equal to the length of two cell diagonals. This is to ensure that any peak and valley in adjacent

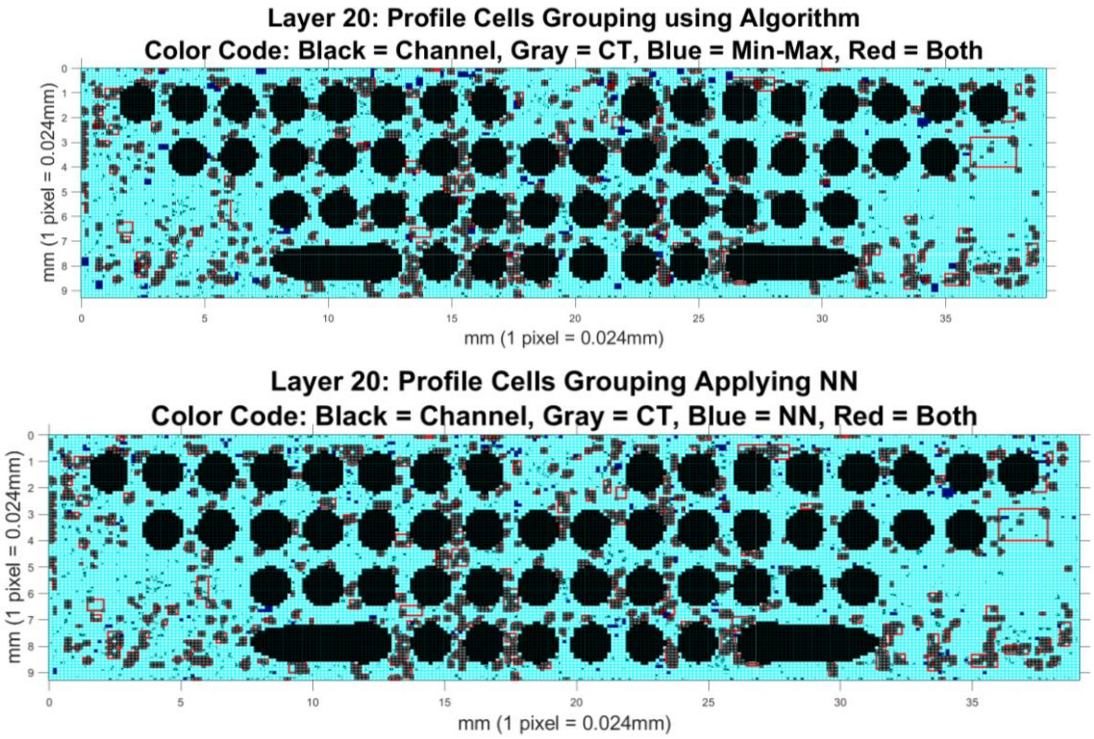
(8-way connectivity) cells would be considered “close”. Geometrically, using this radius means that it is possible for another cell, in which no min or max exists, to lie between a min and a max; thus, it is important to mark the entire region between the valley and peak.

The results of this application are seen in Figure 5.6 and Table 5.7. As mentioned when discussing which NNs to use, the NNs are applied to different layers than they were trained on to mimic application to a different part altogether. Layers 20-25 were chosen for application in the ORNL data, as they were the default layers of examination throughout the entire development process. Layers 129-134 were chosen for application on the PS data; similar to the trained NN, it was manually determined that these layers had a significant number of faults compared to the rest of the part. As with the baseline approach, the results are not very appealing. The best results came when applying a NN trained without stitching together super cells was applied to the ORNL data.



(a)

Figure 5.6. Results of applying a Net trained without stitching to ORNL HeX-B (using the NN trained on stitched ORNL data had similar results). a) ROC curve b) visualization of stitching and NN labeling



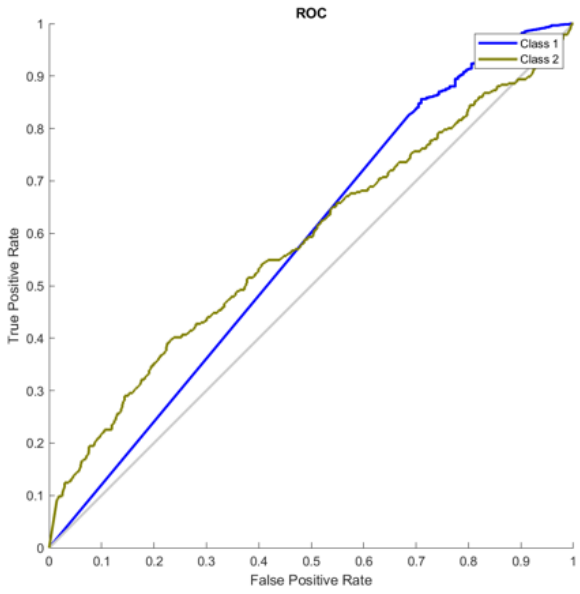
(b)

Figure 5.6 continued

Table 5.7. Confusion Matrix applying a Net trained without stitching to ORNL HeX-B (using the NN trained on stitched ORNL data had similar results).

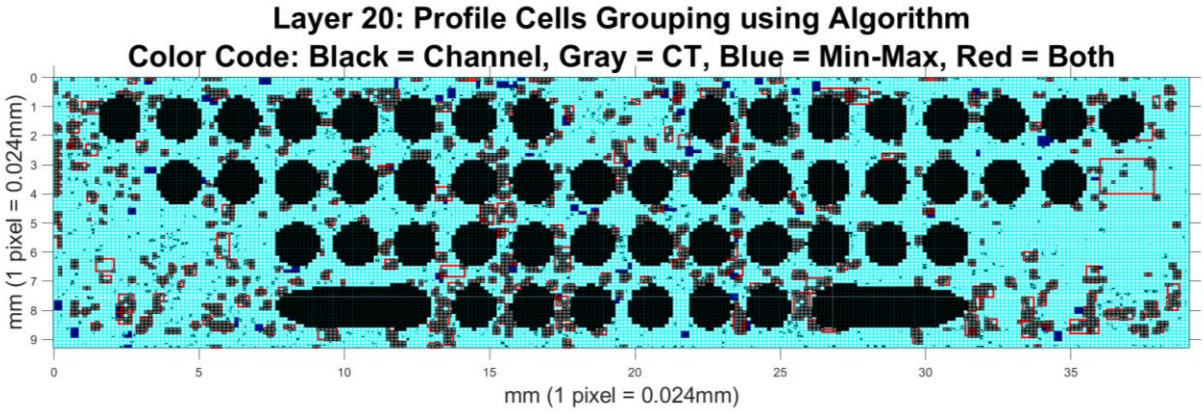
			Actual	
			Nominal (Positive)	Faulty (Negative)
Predicted		Nominal (Positive)	62939 82.7%	12263 16.1%
		Faulty (Negative)	754 1.0%	145 0.2%

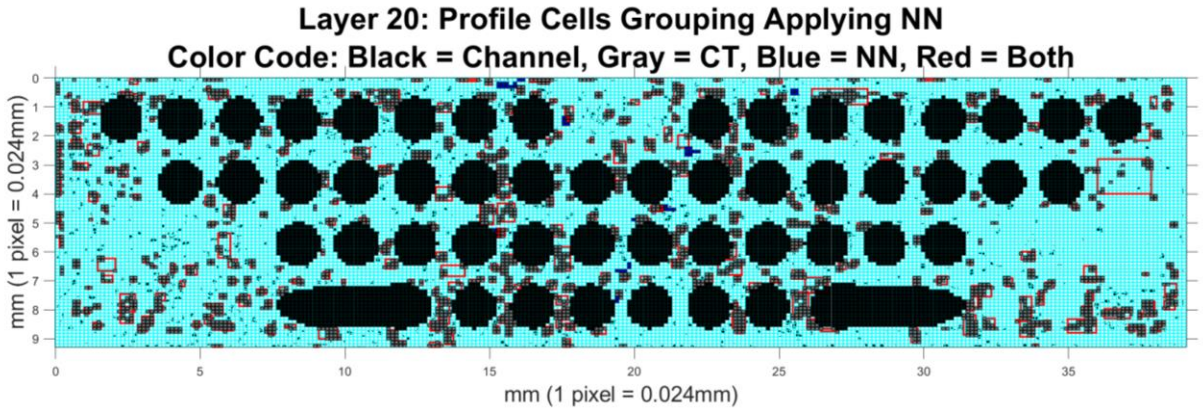
For these tests, each cell was considered individually. Cells were stitched together using the described algorithm. However, they were compared to the ground truth on a cell-by-cell basis. That is, once stitched, each cell in the super cell was assigned the values of the super cell. Each cell was then individually compared to the ground truth. The final alteration to the algorithm was to alter this approach. Instead of cell by cell, the truth was considered fault by fault. In this final approach, all nominal cells that were marked nominal were ignored. For each stitched fault, it's overlap with actual faults was considered; if the percent overlap exceeded a certain threshold, it was deemed a true negative. Otherwise, it was a false negative. Then, the ground truth faults were considered to calculate the false positives. Cells that were nominal and considered nominal by the stitching algorithm were ignored; the application of the NN still yielded some true positives, however. These represent regions that the algorithm stitched into a super cell but were then accepted as nominal by the NN. This step produced mild improvement, with the best scores being with NN-ORNL was applied to the ORNL data set; these are summarized in Figure 5.7 and Table 5.8.



(a)

Figure 5.7. Results of applying a NN-ORNL to ORNL HeX-B a) ROC curve b) visualization of stitching and NN labeling





(b)

Figure 5.7 continued

Table 5.8. Confusion Matrix of applying a NN-ORNL to ORNL HeX-B

		Actual	
		Nominal (Positive)	Faulty (Negative)
Predicted	Nominal (Positive)	1987 100.0%	331 0.0%
	Faulty (Negative)	111 0.0%	55 0.0%

Manual Investigation of Stitching Algorithm

Following the initial tests of this algorithm, as can be seen, the results were positive, meaning a step was taken in the right direction, but still far from perfect. To remedy this, efforts were taken to examine the process manually, cell by cell.. Specifically, areas where the algorithm completely missed a fault, partially found a fault, and mostly found a fault were inspected. In general, it was found that misidentification of a region as a fault by the NN tends not to persist over layers. That is, if the NN says a nominal region is faulty on one layer, it will likely not do so on the next layer. For the sake of brevity, a single large fault that demonstrates all other cases is examined here. The fault was found in ORNL HeX-B using the old format of data. Presented will be information regarding the fault on layer 22, visualized in Figure 5.8. Further information regarding the fault can be found in Table 5.9. Roughness values for individual and combined regions of fault visualized in Figure 5.8. through Table 5.14. Summary of NN scores and roughness values for cells that were identified as faulty using the post-build CT data and stitched using the min-max stitching method. In other words, these values are representative of the NN working correctly to identify faults. These cells are colored red in Figure 5.8 and Table 5.10. NN Scores, using both NNs, for individual cells of the fault visualized in Figure 5.8.examining roughness values and NN output scores. In this example, the color scheme is the same as prior: grey represents CT identified data (faulty cells, blue represents data stitched by the min-max algorithm and identified by the NN (the algorithm/NN misidentified a nominal cell as faulty), and red represents where the CT and stitching algorithm overlapped (the NN correctly identified a fault).

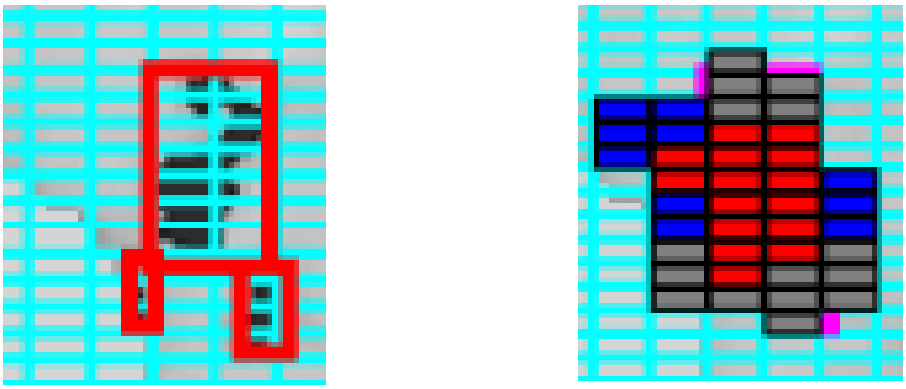


Figure 5.8. Visualization of fault. (a) raw CT scan (b) Effect of Algorithms and NN; grey cells are identified by the CT scan as ground truth, blue represents data stitched by the algorithm and identified by the NN, red is both.

Table 5.9. Roughness values for individual and combined regions of fault visualized in Figure 5.8.

Individual Cell Values						**Combined Values**
		30.72073 43.69672 1.785637 4.075366 -52.2906 108.1594				
		36.15437 37.91246 -0.84109 1.472818 -57.4406 90.55937	67.69547 75.16708 0.733759 1.613794 -153.891 110.5594			
45.91169 53.04222 -1.58039 3.124675 -154.391 -16.1906	38.801 39.28243 -1.03661 1.098605 -53.9906 -28.2406	60.24339 69.13869 0.876637 5.230375 -67.0406 248.8094	48.06633 55.78559 -1.39221 3.260429 -178.291 57.15937			65.38439 81.75634 0.253535 3.249876 -215.591 248.8094
41.84174 52.20428 -1.5866 2.933615 -146.291 10.25937	39.60227 47.19407 -1.67701 4.834622 -196.441 93.05937	61.39853 62.91057 -1.12412 1.479008 -167.891 -47.7406	74.93627 82.1781 -1.32502 2.117239 -215.591 1.809367			
16.7528 29.27341 -2.48989 7.113336 -111.941 18.10937	54.53534 77.33603 0.205124 3.603388 -209.491 185.9094	61.6859 63.56951 -0.5853 1.21305 -112.291 100.1094	100.5482 108.4013 -1.21647 1.612133 -210.091 -48.7406			

Table 5.9 Continued						
Individual Cell Values						**Combined Values**
	43.51764 64.21162 0.337896 5.524855 -179.241 215.8594	63.90107 70.64682 -0.6373 1.891982 -184.991 113.0094	102.9218 124.1532 -1.36609 2.002359 -215.091 55.25937	44.24006 56.73305 -1.59681 2.846427 -143.191 18.95937		
	71.84539 99.51938 1.953368 4.289926 -63.6406 245.6094	94.18992 117.7961 1.478895 2.704027 -110.491 244.5094	74.37039 89.96591 0.523533 2.534663 -196.391 191.7594	68.75287 88.68204 -1.51492 2.996078 -211.591 128.3594		
	49.06206 53.59942 0.790568 1.526795 -102.241 111.5594	114.9785 129.1893 1.184624 1.785235 -174.841 248.0594	62.73285 78.63158 1.601618 2.973594 -34.3906 182.6594	124.5928 137.4959 -0.03416 1.50358 -210.191 206.5094		
		90.17458 97.56502 0.80727 1.564205 -196.641 188.8594	86.64892 101.7027 1.365004 2.023111 -27.8406 181.2094			
		89.84991 93.72903 1.091189 1.305847 -106.341 155.8594				

Table 5.10. NN Scores, using both NNs, for individual cells of the fault visualized in Figure 5.8. The first number represents the score for how nominal the NN thinks the cell is; the second for how faulty.

NN-PS Applied					NN-ORNL Applied				
		1 1.14E-05					1 2.06E-52		
		5.09E-07 1.00	0.9968 3.32E-02				1 4.51E-76	1 6.88E-67	
9.35E-08 1	3.20E-07 1	0.999982 1.83E-05	2.19E-07 1		1 6.23E-41	1 9.45E-44	1 1.67E-80	1 1.24E-41	
3.44E-08 1	1.53E-07 1	5.91E-07 1	2.99E-07 1		1 2.55E-24	1 4.33E-27	1 9.64E-50	1 1.55E-91	
2.32E-08 1	0.729318 2.71E-01	7.79E-06 1	2.28E-06 1		1 1.89E-51	1 3.41E-38	1 2.09E-51	1 1.20E-123	
	0.997838 0.002162	1.49E-05 1	0.011792 0.988208	2.95E-08 1		1 4.02E-48	1 1.76E-31	1 8.62E-81	1 5.06E-30
	1 1.27E-05	1 2.90E-05	0.897608 0.102392	0.543832 0.456168		1 8.97E-67	1 1.62E-83	1 1.21E-28	1 2.84E-25
	0.959412 0.040588	0.999336 0.000664	0.99997 3.05E-05	0.010395 0.989605		1 1.91E-14	1 9.19E-93	1 3.55E-84	1 6.28E-129
		0.953748 0.046252	0.999924 7.63E-05				1 1.39E-32	1 6.43E-95	
		0.997979 0.002021					1 6.35E-76		

Of note in Table 5.10 is that NN-PS seemed to more readily classify individual cells as faulty. This is likely to differences in the parts. The PS part was nearly perfect, with few defects. As such, a NN trained on it likely overestimates what is a fault. That is, when NN-PS, trained on a near flawless part, is applied to a much rougher part, the ORNL build, everything appears to be faulty. This is something that should be considered for the future, as having to retrain the NN for every build defeats the purpose of using the technology for in-situ monitoring and fault correction.

Table 5.11. Summary of NN scores and roughness values when all highlighted cells are grouped and considered a single region

NN-PS Applied			**NN-ORNL Applied**		
NN Scores		**Roughness Values**	**NN Scores**		**Roughness Values**
Nominal	0.355787	65.38439	**Nominal**	6.21E-21	65.38439
Faulty	0.644213	81.75634	**Faulty**	1	81.75634
		0.253535			0.253535
		3.249876			3.249876
		-215.591			-215.591
		248.8094			248.8094

Table 5.12. Summary of NN scores and roughness values for cells that were grouped by the min-max algorithm. These cells are colored blue and red in Figure 5.8 and Table 5.10. NN Scores, using both NNs, for individual cells of the fault visualized in Figure 5.8.

NN-PS Applied			**NN-ORNL Applied**		
NN Scores		**Roughness Values**	**NN Scores**		**Roughness Values**
Nominal	0.299437	54.16713	**Nominal**	0.152606	54.16713
Faulty	0.700563	72.84189	**Faulty**	0.847394	72.84189
		-0.04147			-0.04147
		4.510899			4.510899
		-211.591			-211.591
		245.6094			245.6094

Table 5.13. Summary of NN scores and roughness values for cells that were identified as faulty using the post-build CT data. As such, these cells, combined, are the best representation of how the NN scores an actual fault. These cells are colored grey and red in Figure 5.8 and Table 5.10. NN Scores, using both NNs, for individual cells of the fault visualized in Figure 5.8.

NN-PS Applied

NN Scores		Roughness Values
Nominal	0.306637	69.96949
Faulty	0.693363	85.41104
		0.288051
		2.951297
		-215.591
		248.8094

NN-ORNL Applied

NN Scores		Roughness Values
Nominal	1.03E-18	69.96949
Faulty	1	85.41104
		0.288051
		2.951297
		-215.591
		248.8094

This table clearly demonstrates that the NN does produce a positive response to truly faulty regions. The difficulty, as noted, comes in that if which regions are faulty is known, the NN is effectively pointless. It is also of note that NN-PS identified the truly faulty region as slightly less likely of being a fault than the region identifying by our min-max grouping algorithm, seen in Table 5.12.

Table 5.14. Summary of NN scores and roughness values for cells that were identified as faulty using the post-build CT data and stitched using the min-max stitching method. In other words, these values are representative of the NN working correctly to identify faults. These cells are colored red in Figure 5.8 and Table 5.10. NN Scores, using both NNs, for individual cells of the fault visualized in Figure 5.8.

NN-PS Applied

NN Scores		**Roughness Values**
Nominal	0.834598	81.33578
Faulty	0.165402	97.55245
		0.554452
		2.579257
		-215.091
		248.0594

NN-ORNL Applied

NN Scores		**Roughness Values**
Nominal	0.001526	81.33578
Faulty	0.998474	97.55245
		0.554452
		2.579257
		-215.091
		248.0594

CHAPTER SIX

DISCUSSION AND FUTURE WORK

It has been demonstrated that despite promising initial results, the application results of a NN trained on surface height data may be ill-suited for direct in-situ fault detection. Having said that, the algorithms and insights found through this process may prove useful in the future. The characterization of faults by more extreme peaks and valleys, corroborated by manually found faults, while expected, is a valuable find. Further, the codes written for this project, specifically in regards to extracting and visualizing information about faults, may yet prove useful. However, future attempts to use this data to characterize faults using a Machine Learning approach will likely have to be considered from the ground up, preferably with a less skewed approach. Last, at the time of writing, there a few more avenues that may be investigated, such as a parametric study of the stitching algorithm to see if some “magical combination” of parameters exists that we failed to notice or reconsidering the base approach, but restricting the number of nominal examples to roughly match the number of faulty examples.

LIST OF REFERENCES

[1] D. R. Ian Gibson, Brent Stucker, *Additive Manufacturing Technologies: 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing*. Springer-Verlag New York, 2015, p. 498.
[2] I. Gibson, D. Rosen, and B. Stucker, *Additive Manufacturing Technologies: 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing*, 2nd ed.: Springer, 2015.
[3] S. M. Kelly, "Thermal and Microstructure Modeling of Metal Deposition Processes with Application to Ti-6Al-4V," Doctor of Philosophy, Virginia Tech, 2004.
[4] A. Plotkowski, M. M. Kirka, and S. S. Babu, "Verification and validation of a rapid heat transfer calculation methodology for transient melt pool solidification conditions in powder bed metal additive manufacturing," *Additive Manufacturing,* vol. 18, pp. 256-268, 2017/12/01/ 2017, doi: https://doi.org/10.1016/j.addma.2017.10.017.
[5] J. Raplee *et al.*, "Thermographic Microstructure Monitoring in Electron Beam Additive Manufacturing," *Scientific Reports,* vol. 7, p. 43554, 03/01 2017, doi: 10.1038/srep43554.
[6] T. Debroy and S. David, "Physical Processes in Fusion-Welding," *Reviews of Modern Physics - REV MOD PHYS,* vol. 67, pp. 85-112, 01/01 1995, doi: 10.1103/RevModPhys.67.85.
[7] S. S. Babu, "Introduction to Integrated Weld Modeling[1]," *vol. 6A*, *Welding Fundamentals and Processes*: ASM International, 2011, p. 0. [Online]. Available: https://doi.org/10.31399/asm.hb.v06a.a0005561
[8] Y. S. Lee and W. Zhang, "Modeling of heat transfer, fluid flow and solidification microstructure of nickel-base superalloy fabricated by laser powder bed fusion," *Additive Manufacturing,* vol. 12, pp. 178-188, 2016/10/01/ 2016, doi: https://doi.org/10.1016/j.addma.2016.05.003.
[9] "Weld Solidification," in *Weld Integrity and Performance*, S. Lampman Ed.: ASM International, 1997.
[10] S. David and J. Vitek, "Correlation between solidification parameters and weld microstructures," *International Materials Reviews,* vol. 34, pp. 213-245, 01/01 1989, doi: 10.1179/imr.1989.34.1.213.
[11] R. R. Dehoff *et al.*, "Site specific control of crystallographic grain orientation through electron beam additive manufacturing," *Materials Science and Technology,* vol. 31, no. 8, pp. 931-938, 2015/06/01 2015, doi: 10.1179/1743284714Y.0000000734.
[12] "Phase diagrams and solidification." University of Cambridge. https://www.doitpoms.ac.uk/tlplib/phase-diagrams/index.php (accessed.
[13] K. Keil, J. L. Berkley, and L. H. Fuchs, "Suessite, Fe3Si: a new mineral in the North Haig ureilite," *American Mineralogist,* vol. 67, no. 1-2, pp. 126-131, 1982.
[14] T. V. H. S. Cottenier;. "The Fe-Si phase diagram: from electrical steel to the planet Mercury." Center for Molecular Modeling.

https://molmod.ugent.be/subject/fe-si-phase-diagram-electrical-steel-planet-mercury-0 (accessed.
[15] B. K. D. G. R. E. B. B. RATH, "Investigation of Subsolidus Weld Cracking in Alpha-Beta Titanium Alloys," 1989.
[16] B. R. S. G. S. Babu, "Large Scale Phase Field Simulations of Microstructure Evolution During Thermal Cycling of Ti-6Al-4V," presented at the International Conference on Solid-Solid Phase Transformations in Inorganic Materials, 2015.
[17] S. Babu, "Rationalization of Microstructural Heterogeneity in Components Produced by Welding and Additive Manufacturing," presented at the International Conference on Solid-Solid Phase Transformations in Inorganic Materials, 2015.
[18] D. Rosenthal, "Mathematical Theory of Heat Distribution During Welding and Cutting," *Welding Journal,* vol. 20, no. 5, pp. 220-234, 1941.
[19] R. Komanduri and Z. B. Hou, "Thermal analysis of the arc welding process: Part I. General solutions," *Metallurgical and Materials Transactions B,* vol. 31, no. 6, pp. 1353-1370, 2000/12/01 2000, doi: 10.1007/s11663-000-0022-2.
[20] Y.-C. Wu *et al.*, "Numerical modeling of melt-pool behavior in selective laser melting with random powder distribution and experimental validation," *Journal of Materials Processing Technology,* vol. 254, pp. 72-78, 2018/04/01/ 2018, doi: https://doi.org/10.1016/j.jmatprotec.2017.11.032.
[21] G. Tapia and A. Elwany, "A Review on Process Monitoring and Control in Metal-Based Additive Manufacturing," *Journal of Manufacturing Science and Engineering,* vol. 136, no. 6, pp. 060801-060801-10, 2014, doi: 10.1115/1.4028540.
[22] S. K. Everton, M. Hirsch, P. Stravroulakis, R. K. Leach, and A. T. Clare, "Review of in-situ process monitoring and in-situ metrology for metal additive manufacturing," *Materials & Design,* vol. 95, pp. 431-445, 2016/04/05/ 2016, doi: https://doi.org/10.1016/j.matdes.2016.01.099.
[23] C. Z. J. T. W. S. M. M. Diagne, "Modeling and Cooling Rate Control in Laser Additive Manufacturing: 1-D PDE Formulation," presented at the Annual Conference on Decision and Control, Melbourne, Australia, December 12-15, 2017.
[24] W. S. J. D. K. J. D. J.-P. Kruth, "Arrangement and method for the on-line monitoring of the quality of a laser process exerted on a workpiece," 2011.
[25] S. Kleszczynski, J. zur Jacobsmühlen, J. Sehrt, and G. Witt, *Error detection in laser beam melting systems by high resolution imaging*. 2012, pp. 975-987.
[26] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: John Wiley & Sons, 2000.

[27] "What is the Role of Bias in Neural Networks." StackOverflow. https://stackoverflow.com/questions/2480650/what-is-the-role-of-the-bias-in-neural-networks (accessed.
[28] "How to add bias and weight to neural network diagram?" Stack Exchange. https://tex.stackexchange.com/questions/401681/how-to-add-bias-and-weight-to-neural-network-diagram (accessed.
[29] D. Powers, *Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation*. 2008.
[30] "How to analysis this ROC curve and improve the performance? [closed]." https://stats.stackexchange.com/questions/263732/how-to-analysis-this-roc-curve-and-improve-the-performance (accessed.
[31] C. Gobert, E. W. Reutzel, J. Petrich, A. R. Nassar, and S. Phoha, "Application of supervised machine learning for defect detection during metallic powder bed fusion additive manufacturing using high resolution imaging," *Additive Manufacturing,* vol. 21, pp. 517-528, 2018/05/01/ 2018, doi: https://doi.org/10.1016/j.addma.2018.04.005.
[32] L. Scime and J. Beuth, "Anomaly detection and classification in a laser powder bed additive manufacturing process using a trained computer vision algorithm," *Additive Manufacturing,* vol. 19, pp. 114-126, 2018/01/01/ 2018, doi: https://doi.org/10.1016/j.addma.2017.11.009.
[33] L. Scime and J. Beuth, "Using machine learning to identify in-situ melt pool signatures indicative of flaw formation in a laser powder bed fusion additive manufacturing process," *Additive Manufacturing,* vol. 25, pp. 151-165, 2019/01/01/ 2019, doi: https://doi.org/10.1016/j.addma.2018.11.010.
[34] T. Purtonen, A. Kalliosaari, and A. Salminen, "Monitoring and Adaptive Control of Laser Processes," *Physics Procedia,* vol. 56, pp. 1218-1231, 2014/01/01/ 2014, doi: https://doi.org/10.1016/j.phpro.2014.08.038.
[35] M. Abdelrahman, E. W. Reutzel, A. R. Nassar, and T. L. Starr, "Flaw detection in powder bed fusion using optical imaging," *Additive Manufacturing,* vol. 15, pp. 1-11, 2017/05/01/ 2017, doi: https://doi.org/10.1016/j.addma.2017.02.001.
[36] A. H. Chern *et al.*, "A review on the fatigue behavior of Ti-6Al-4V fabricated by electron beam melting additive manufacturing," *International Journal of Fatigue,* vol. 119, pp. 173-184, 2019/02/01/ 2019, doi: https://doi.org/10.1016/j.ijfatigue.2018.09.022.
[37] B. Baucher, A. B. Chaudhary, S. S. Babu, and S. Chakraborty, "Defect Characterization Through Automated Laser Track Trace Identification in SLM Processes Using Laser Profilometer Data," *Journal of Materials Engineering and Performance,* vol. 28, no. 2, pp. 717-727, 2019/02/01 2019, doi: 10.1007/s11665-018-3842-4.
[38] "Affine Transformation: Linear mapping method using affine transformation." MathWorks. https://www.mathworks.com/discovery/affine-transformation.html (accessed.

[39] W. Burger and M. J. Burge, *Principles of Digital Image Processing: Core Algorithms*, 1 ed. (Undergraduate Topics in Computer Science). Springer-Verlag London, 2009.

VITA

Benjamin Terry was born December 12, 1995 in Memphis, TN, where he has lived since. He attended St. Dominic School from kindergarten through 8th grade. He then moved onto to Christian Brothers High School. Benjamin excelled academically, becoming salutatorian of his class.

After graduating, he chose to attend the University of Tennessee as it was the best school that did not cost upwards of $50,000 a year. He studied Mechanical Engineering, though has shifted away from this field towards more data analytics and machine learning, as demonstrated in this thesis. During his undergraduate time at UT, Benjamin Took part in many activities. Outside of his direct schoolwork, he worked as tutor and studied Ju Jitsu as part of the Martial Arts Club on Campus. Benjamin spent one summer studying abroad in England, and another interning as a web developer in Singapore (this internship was acquired independent of the university's career resources). Benjamin ultimately graduated in May of 2018.

From there, he immediately entered the Masters Program at UT. Having been caught up with finishing is undergraduate requirements, Benjamin was fortunate when Dr. Chakraborty, his undergraduate vibrations professor, mentioned that he needed someone to work in his lab on a project over the summer. At the time, Benjamin was interested in Mechatronics, Robotics, and Controls, which Dr. Chakraborty's Lab generally specialized in, and so he offered his services. Benjamin's first project was running experiments for a Heat Exchanger (not detailed in his thesis) for there sponsor, Applied Optimization, located in Fairborn, OH.

Unfortunately, the contract for this work was not finalized until after fees for the fall semester were due. As such, Benjamin changed his primary work to that of Graduate Teaching Assistant, working for the Engineering Fundamentals Program. During this time, Benjamin continued to work for Dr. Chakraborty on the Applied Optimization project, though the project took a back seat to his primary engagement as a GTA and a student.

Moving forward, the project that forms the basis of this thesis was begun, with Benjamin as a key member performing the work described and taking relevant classes. During his first few semesters as a graduate student, Benjamin also began considering his future prospects and ultimately decided to pursue an MBA in addition to his Masters in Engineering. Benjamin has recently finished both and is currently looking for employment, a process largely hampered by the COVID-19 crisis (and hopefully changed by the time this thesis is accepted).