Old Dominion University
ODU Digital Commons

Computer Science Theses & Dissertations | Computer Science

Summer 8-2022

Evaluation of Generative Models for Predicting Microstructure Geometries in Laser Powder Bed Fusion Additive Manufacturing

Andy Ramlatchan
Old Dominion University, araml001@odu.edu

Follow this and additional works at: https://digitalcommons.odu.edu/computerscience_etds

 Part of the Artificial Intelligence and Robotics Commons

Recommended Citation

Ramlatchan, Andy. "Evaluation of Generative Models for Predicting Microstructure Geometries in Laser Powder Bed Fusion Additive Manufacturing" (2022). Doctor of Philosophy (PhD), Dissertation, Computer Science, Old Dominion University, DOI: 10.25777/0jp3-hq36
https://digitalcommons.odu.edu/computerscience_etds/135

This Dissertation is brought to you for free and open access by the Computer Science at ODU Digital Commons. It has been accepted for inclusion in Computer Science Theses & Dissertations by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.

EVALUATION OF GENERATIVE MODELS FOR PREDICTING MICROSTRUCTURE GEOMETRIES IN LASER POWDER BED FUSION ADDITIVE MANUFACTURING

by

Andy Ramlatchan
B.S. May 2008, Old Dominion University
M.B.A. December 2010, Averett University

A Dissertation Submitted to the Faculty of Old Dominion University in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
August 2022

Approved by:

Yaohang Li (Director)

Mohammad Zubair (Member)

Lusi Li (Member)

Yuan Zhang (Member)

ABSTRACT

EVALUATION OF GENERATIVE MODELS FOR PREDICTING MICROSTRUCTURE GEOMETRIES IN LASER POWDER BED FUSION ADDITIVE MANUFACTURING

Andy Ramlatchan
Old Dominion University, 2022
Director: Dr. Yaohang Li

In-situ process monitoring for metals additive manufacturing is paramount to the successful build of an object for application in extreme or high stress environments. In selective laser melting additive manufacturing, the process by which a laser melts metal powder during the build will dictate the internal microstructure of that object once the metal cools and solidifies. The difficulty lies in that obtaining enough variety of data to quantify the internal microstructures for the evaluation of its physical properties is problematic, as the laser passes at high speeds over powder grains at a micrometer scale. Imaging the process in-situ is complex and cost-prohibitive. However, generative modes can provide new artificially generated data. Generative adversarial networks synthesize new computationally derived data through a process that learns the underlying features corresponding to the different laser process parameters in a generator network, then improves upon those artificial renderings by evaluating through the discriminator network. While this technique was effective at delivering high-quality images, modifications to the network through conditions showed improved capabilities at creating these new images. Using multiple evaluation metrics, it has been shown that generative models can be used to create new data for various laser process parameter combinations, thereby allowing a more comprehensive evaluation of ideal laser conditions for any particular build.

The outputs of both generative adversarial networks were compared to results obtained using recommender systems and variational autoencoders. The recommender system approach utilized a matrix completion framework whereby the missing data was computationally approximated, thereby allowing the model to make predictions, or recommendations, on what microstructure object characteristics would be produced. The variational autoencoder model used a deep learning framework to also try and predict the output, leveraging the generative capabilities with a goal similar to the adversarial networks, albeit in a different fashion. However, while informative, neither of these approaches matched the accuracy of the output of either of the generative adversarial networks when compared to holdout datasets. Overall, the conditional adversarial network proved superior in generating new data from experimentally collected x-ray images, which can be implemented into many other applications of image representation.

Copyright, 2022, by Andy Ramlatchan, All Rights Reserved.

This dissertation is dedicated to everyone who works add to
the scientific body of knowledge of humanity.

ACKNOWLEDGEMENTS

I would like to thank Dr. Yaohang Li for all of his support and guidance over the past 5 years as I have worked on this research. His support was invaluable into having the results I have obtained. Next I would like to thank Dr. Lusi Li, Dr. Mohammad Zubair, and Dr. Yuan Zhang for all of their feedback and encouragement into improving the outcomes of this work.

I would also like to thank the faculty and staff of the Department of Computer Science at Old Dominion University, for their administrative and instructional support over these last few years.

Finally, I would like to thank my wife and kids for their non-stop support and encouragement.

TABLE OF CONTENTS

Page

LIST OF TABLES……..x

LIST OF FIGURES……xii

Chapter

1. INTRODUCTION……....1
   - 1.1 PROBLEM BACKGROUND……1
   - 1.2 MAJOR CONTRIBUTIONS…….....3
   - 1.3 DISSERTATION ORGANIZATION……4

2. LITERATURE REIVEW ……...…….……7
   - 2.1 OVERVIEW OF METALS ADDITIVE MANUFACTURING……7
   - 2.2 LASER POWDER BED FUSION……....11
   - 2.3 MATERIAL MICROSTRUCTURES…….....13
   - 2.4 MICROSTRUCTURE DEFECTS……..14
   - 2.5 PHYSICS VALIDATION FOR THERMAL MODELING……15
   - 2.6 DEEP LEARNING WITH CONVOLUTIONAL NEURAL NETWORKS……..19
   - 2.7 IN-SITU CHARACTERIZATION CHALLENGES……..23
   - 2.8 MISSING DATA FOR PROCESS CHARACHTERIZATION…….....24
   - 2.9 MATRIX COMPLETION OVERVIEW…….....25
   - 2.10 VARIATIONAL AUTOENCODERS THEORY……31
   - 2.11 GENERATIVE ADVERSARIAL NETWORKS THEORY……37
   - 2.12 CGAN THEORY…….....41
   - 2.13 ADDITIONAL APPLICATIONS OF GENERATIVE MODELS……..43

3. MACHINE LEARNING METHODOLOGY…….....46
   - 3.1 DATA COLLECTION THROUGH ULTRA-HIGH SPEED IMAGING…….....46
   - 3.2 COMPUTER VISION TO LEARN VAPOR DEPRESSION GEOMETRIES……49
   - 3.3 DEEP LEARNING FOR MICROSTRUCTURE DEFECT DETECTION…….……53
   - 3.4 RECOMMENDER SYSTEMS BASED ON MATRIX COMPLETION…….....54
   - 3.5 VAE TO GENERATE NEW VAPOR DEPRESSION IMAGES…….……56
   - 3.6 GAN TO GENERATE NEW DEFECT REPRESENTATIVE IMAGES……58

4. DEEP LEARNING FOR GEOMETRY CLASSIFICATION……62
   - 4.1 CNN FOR VAPOR DEPRESSION CHARACTERIZATION……62
   - 4.2 GEOMETRY CLASSIFICATION RESULTS…….63
   - 4.3 CNN FOR DEFECT DETECTION…… …….....65
   - 4.4 TRANSFER LEARNING…….67
   - 4.5 DEFECT DETECTION EVALUATION…… ……69

Chapter Page

4.6 CNN SUMMARY……73

5. COMPUTATIONAL GEOMETRIC ANALYSIS…….....75
5.1 CONVEX HULL GEOMETRIES……75
5.2 CONVEX HULL GEOMETRIX APPROACH……76
5.3 CONVEX HULL RESULTS AND ANALYSIS…….....79
5.4 VORONOI DIAGRAM WITH CLUSTERING…….....81
5.5 VORONOI COMPUTATIONAL APPROACH…….....83
5.6 VORONOI DIAGRAM WITH CLUSTERING ANALYSIS…….....85

6. TIME-RESOLVED GEOMETRIC QUANTIFICATION…….....87
6.1 THERMAL EFFECTS ON MORPHOLOGY…….....87
6.2 ADVANCED IMAGE PROCESSING FOR FEATURE EXTRACTION…….....88
6.3 GEOMETRIC FEATURE EXTRACTION…….....95
6.4 NONPARAMETRIC DATA ANALYSIS……98
6.5 KEYHOLE SEGMENTATION……99
6.6 TIME-RESOLVED GEOMETRIC FEATURES……100
6.7 SPEARMAN'S CORRELATION ……..……101
6.8 AGGLOMERATIVE HIERARCHICAL CLUSTERING……..104
6.9 SUMMARY OF GEOMETRIC FEATURE TRACKING……..104

7. IMAGE SYNTHESIS USING GAN AND CGAN……...108
7.1 IMAGE GENERATION OPPORTUNITIES……108
7.2 GAN MODELING…….....109
7.3 CGAN MODELING…….....110
7.4 GENERATED IMAGE ANALYSIS…….....115
7.5 TRAIN ON SYNTHETIC TEST ON REAL…….....116
7.6 INTERSECTION OVER UNION……...118
7.7 HAUSDORFF DISTANCE ...……119
7.8 MAXIMUM MEAN DISCREPANCY…….....121
7.9 AVOIDANCE OF OVERFITTING……123
7.10 GAN RESULTS WITH RMSE…….....125
7.11 EFFECTIVENESS OF GENERATED IMAGES……...127
7.12 NEW PROCESS PARAMETER GENERATION RESULTS……..…..130
7.13 GENERATIVE OUTPUT SUMMARY……134

8. GAN WITH LIMITED DATA FOR IMAGE SEGMENTATION……...138
8.1 MELT POOL DYNAMICS……138
8.2 MICROGRAPH DATASET……..140
8.3 PIX2PIX AND U-NET FOR IMAGE TRANSLATION……..…140
8.4 PIX2PIX AND U-NET PERFORMANCE EVALUATION…….....143
8.5 SUMMARY OF PIX2PIX ANALYSIS…….....149

9. IMAGE SYNTHESIS WITH VAE……..150

Chapter Page

9.1 VARIATIONAL AUTOENCODER IMPLEMENTATION…….....150
9.2 EFFECTIVENESS OF VAE GENERATED IMAGES…….....152
9.3 CONDITIONAL VAE IMPLEMENTATION…….....154
9.4 CONDITIONAL VAE RESULTS…….....155
9.5 VAE RESULTS WITH RMSE…….……..……157

10. RECOMMENDER SYSTEMS FOR MICROSTRUCTURE GENERATION… ……...159
10.1 ALTERNATING LEAST SQUARES……159
10.2 ALS RESULTS WITH RMSE……162
10.3 ACCELERATED PROXIMAL GRADIENT……163
10.4 APG RESULTS AND ANALYSIS…….……...166
10.5 NOISE REDUCTION WITH ALS AND APG……..167
10.6 APG FOR DATA GENERATION……169
10.7 RECOMMENDER SYSTEM SUMMARY…….....171

11. COMPARISON OF GENERATIVE MODELS……...175

12. CONCLUSION AND FUTURE WORK …….....179

13. PUBLICATIONS ……...…… 184

REFERENCES……..... 185

VITA ...…….... 202

LIST OF TABLES

Table Page

1. Classification Performance Summary…….....…….....64

2. CNN Performance Summary……...……..……73

3. Calculations for Convex Hull……..…….....79

4. The Process Parameters of the 14 Samples…….…….....103

5. Spearman's Rank-Order Correlation Coefficients Between Measured and Derived Variables ……...…….……103

6. AUPRC and AUROC from TSTR……..…….118

7. IoU and HD Measurements…….…….……122

8. GAN Evaluation with RMSE and SI Metrics …….…….128

9. CGAN Evaluation with RMSE and SI Metrics…….……..…….128

10. Average Values for Vapor Depression Geometries by CGANS ……135

11. Average Metrics with 95% Confidence Interval as Evaluated on all Training Set Images…..……..……..145

12. Average Area of Real and VAE Images at 350W.…….....…….154

13. IoU and HD of CVAE Generated Images…….....156

14. VAE Evaluation with RMSE and SI Metrics……..……...……157

15. CVAE Evaluation with RMSE and SI Metrics……..……..……...…158

16. ALS Results of Parameters Tested by Removal from Dataset ……..……....163

17. APG Results of Parameters Tested by Removal from Dataset…….……..167

18. ALS and APG Results with Induced Noise……..……..……..…..169

19. Average Values for Vapor Depression Geometries by APG……...……...…173

Page

20. Comparison of Generative Results for Area Calculations……..…….177

LIST OF FIGURES

Figure Page

1. A flow chart depicting the research process……5

2. 3-d printed rocket nozzle made from copper metal……...…….10

3. LPBF process setup……12

4. Dynamix x-ray image of in-situ LPBF process…….....16

5. In-situ defect generation……25

6. Basic autoencoder network……33

7. GAN algorithm example……38

8. CGAN modification to input values……...……42

9. Process parameter combinations experimentally produced……...48

10. Background subtraction on DXR image…….……..52

11. Non-local means denoising……..52

12. Keyholing classes……...……..64

13. CNN model architecture……...64

14. CNN model performance over training cycles…….…66

15. Confusion matrix for NASNetMobile-B after testing……..70

16. Confusion matrix with DenseNet121-A after testing……71

17. Confusion matrix for the custom CNN model after testing……..71

18. Graph of the ROC curves for the DL models implemented…….....72

19. Original image, geometric area in green, and convex hull area in blue……77

20. Convex hull areas versus geometric areas…….80

Page

21. Convex hull areas clustered against time…….....86

22. A flow chart illustrating the image processing pipeline……89

23. (a) Gray-scaled raw image of LPBF with red line indicating reference level, (b) the same image after transforming and cropping……..……...…….90

24. (a) The final image from Figure 5.2b after rigid transformation and cropping; (b) with the resultant pixel intensity histogram and CDF from (a); (c) after local background division; (d) with the resultant pixel intensity histogram and CDF from (c); (e) after clipping; (f) with the resultant pixel intensity histogram and CDF from (e); (g) after normalization; (h) with the resultant pixel intensity histogram and CDF from (g); (i) after adaptive histogram equalization; (j) with the resultant pixel intensity histogram and CDF from (i)……..92

25. The final image from Figure 6.3i (a) after global binary thresholding; (b) after morphological opening transformation; (c) after filling holes in the keyhole; (d) after median blur; (e) after picking up the largest object……96

26. The final image from Figure 6.4e after the geometric feature extraction process with the target features of the keyhole shown with colored lines……96

27. Gray-scaled raw images from laser melting experiments performed on Ti64 with (a) a power of 197 *W*, velocity of 0.6 *m*/*s*, and beam diameter of 65 μm ; (b) after image processing of (a); (c) a power of 426 *W*, velocity of 0.9 *m*/*s*, and beam diameter of 65 μm ; (d) after image processing of (c); (e) a power of 426 *W*, velocity of 1.2 *m*/*s*, and beam diameter of 65 μm ; (f) after image processing of (e); (g) a power of 426 *W*, velocity of 0.6 *m*/*s*, and beam diameter of 65 μm ; (h) after image processing of (g)…….....100

28. Time-resolved plots depicting the static features for the keyhole (a) depth; (b) top width; (c) aspect ratio of the depth over the top width; (d) whole-body area; (e) perimeter; (f) front wall angle. The x-axis represents the frame number which were captured every 20 μs……..102

29. Time-resolved plots depicting the dynamic features for the keyhole (a) depth; (b) top width; (c) aspect ratio of the depth over the top width; (d) whole-body area; (e) perimeter; (f) front wall angle. The x-axis represents the frame number which were captured every 20 μs……...……..102

30. A dendrogram showing the calculated dissimilarities between the extracted keyhole geometric features and processing conditions for 14 laser melting experiments on Ti64……105

31. GAN layers…….....112

Page

32. Real image (left), GAN generated image (center), and CGAN generated image (right)…...……116

33. Area measurements of vapor depression images for real data (top), GAN generated data (center), and CGAN generated (bottom)……..117

34. Intersection over Union approach for overlap……120

35. IoU overlap of predicted (red) and ground truth (blue) images……..120

36. HD between two points located by the end of the arrows……122

37. Sample distributions in comparison of the real data along with both the GAN (top) and CGAN (below) generated data…….125

38. Qualitative comparison of two melt pool types: within the conduction regime (a-d) and keyholing (e-h). Parts a and e are the input micrographs, image 2 and image 6 respectively. Parts b and f are the ground-truth masks. Parts c and g are comparison masks from the U-Net algorithm. Parts d and h are comparison masks from the GAN algorithm. Orange points denote the Hausdorff point on the ground-truth masks; blue points denote the Hausdorff point on the generated masks……...144

39. Quantitative metrics on individual test images for (a) IoU (b) HD in pixels…….....146

40. Correlation of HD to IoU……148

41. Images from real dataset at 350W (top) at 0.2, 0.4, 0.6, 0.8, and 1.0 m/s, with images generated by VAE (bottom) from that training data……..152

42. Images from real dataset at 350W (top) at 0.2, 0.4, 0.6, 0.8, and 1.0 m/s, with images generated by CVAE (bottom) from each class……..155

CHAPTER 1

INTRODUCTION

1.1 Problem Background

Additive manufacturing is a field within materials science research that has seen tremendous growth over the years due to its advantages over traditional metallurgical techniques for aerospace applications. However, these advantages also come with some disadvantages that are inherent to the process. Some of these include the manifestation of defects in the material microstructure, which are directly caused by the mechanism of using a laser to melt metal powder at extremely high temperatures. This is the case with a technique known as laser powder bed fusion – where a laser is used to melt and fuse metal powder in a specified pattern to build an object. And it is during that process of the laser passing over the material that causes defects to form in-situ. Although research is ongoing into methods to characterize the structures built by the additive process, there currently exists very little work done on examining the fusion process in-situ – to examine the process exactly when those defects are generated, and to therefore examine how those defects affect the material microstructure during the build process. The resultant microstructure has known effects on the quality of the build, yet the nature of the defect generating process that affects that microstructure still has many open questions. This is mostly due to limitations in collecting this data, at the scale and speed necessary to adequately capture the build process.

The lack of experimental data to evaluate the in-situ process in laser powder bed fusion additive manufacturing can be alleviated by exploring machine learning methods that can approximate and even generate data representations. The focus of this work is to develop a methodology that can be generalized to learn from the limited data available, to interpolate from it, and ultimately develop new data computationally that can inform research into the

microstructure quality at the time of the build. This effort will therefore lead to process characterization that does not currently exist.

The process of certifying additively manufactured aerospace parts for use in service is costly in both time and money. For laser powder bed fusion, computational modeling can alleviate some expensive experiments, but these simulations must have experimental validation, best performed by comparing the cross-sectional view of the solidified melt pool to the solidus isotherm predicted by the model. However, melt pool shapes are dynamic, and multiple micrographs are needed to ensure a model is properly calibrated. Automatic extraction of melt pools is needed, and machine learning provides a solution.

The goal of this work was to generate new expressions of material microstructures that were not, and cannot be experimentally produced. Initially, numerical data was explored, where combinations of laser parameters that produced known measurements of microstructure objects as identified during a series of computer vision work was modeled. This effort provided a means for determining the optimal process parameters with respect to the size and shape of the microstructure physical features. Next, two deep learning approaches were developed using generative models. The first focused on generating new computationally derived images for microstructures at settings that were not experimentally performed and for which no data was collected. This provided an ability to examine novel microstructures with a wider range of physical features and geometries beyond the limited data available. Next, an alternate approach was developed which focused specifically on generating new images of microstructures with defects. While the data that does exists for in-situ processes are limited, data that depicts the defect generating process is even sparser. Therefore, the development of a capability to generate new defect structures, with respect

to the previously examined laser parameter settings provided a new data driven approach to computationally model the additive manufacturing process.

1.2 Major Contributions

The use of generative models for data creation has been explored in other use cases, however not for image processing problems where images contain very small gradient differences between one feature of interest and another. There also exists very little research into how to modify the general form of these models to accept more inputs, to correspond to the changing image representations that occur with changing experimental conditions. For an agency such as NASA who wishes to use generative models to create artificially rendered systems – the ability to alter those renderings based on differing conditions is essential. In this work, in order to create new representations of material microstructure images, it is not enough to train a model based solely on the images that were experimentally created. In order for the new data to be useful for process certification and thermal model validation, there should be a way to create new images based on altering the experimental conditions that would have led to those new data. This is in stark contrast to some of the general use-cases for generative adversarial networks, where the user creates new artificial human faces, or in the case of NASA extraplanetary research where these models are used to render hypothetical images of new planetary environments.

To compare generative modeling capabilities, two different image based approaches were explored: the previously mentioned generative adversarial network, and the variational autoencoder. While both of these network architectures can produce new images, the underlying mechanisms by which those new images are generated are fundamentally different. From there, altering the general network architectures to include additional inputs over the images are included,

with results that indicate those novel networks achieve stronger performance for both categories of generative deep learning models.

Following the deep learning methodologies, this work provides a new method for generating new data from image data in the form of the non-image based area of machine learning known as recommender systems. These systems input data in a two-dimensional matrix format and provides recommendations of new geometries, which therefore are comparable to the image outputs from the generative models. And as these approaches do not utilize the computationally taxing deep learning frameworks, their implementation can be applied easier in many cases. The final results are then comparable to the outputs of the generative models, thereby providing researchers a new consideration when it comes to generating new hypothetical data.

This work introduces the use of a Hausdorff distance for evaluating the output of a GAN and VAE and comparing those results to one another. In addition to model comparison, there is a presentation of the following advantages of using the Hausdorff distance over exclusively using intersection over union, or IoU, which is a commonly used metric for image representation tasks. These advantages include considerations for the accuracy of an edge metric for the segmentation, interpretability to a user not versed in machine learning, and the propensity to identify outliers in performance beyond IoU.

1.3 Dissertation Organization

This work is organized as follows. Initially the additive manufacturing process will be examined throughout the literature review for technical details as well as the underlying thermophysics that governs the internal microstructure formation. In that chapter, the literature is reviewed for deep learning for image classification, deep learning for generative models, and the background on recommender systems. The methodology begins by describing the work for

organizing, cleaning, and managing experimentally derived data where a limited amount of image data that was collected was disseminated using computer vision techniques such as pattern recognition and image segmentation. Next, those data were analyzed to understand the complex geometries involved in microstructure characterization, which included understanding defect generation and time resolved geometric evaluation. Upon understanding how microstructures can be quantifiably evaluated, the first generative modeling approach is presented using generative adversarial networks, followed by the second approach using variational autoencoders. The next chapter then discusses the implementation of recommender systems based on matrix completion methodologies. Finally, the conclusion is presented where there is a discussion on using machine

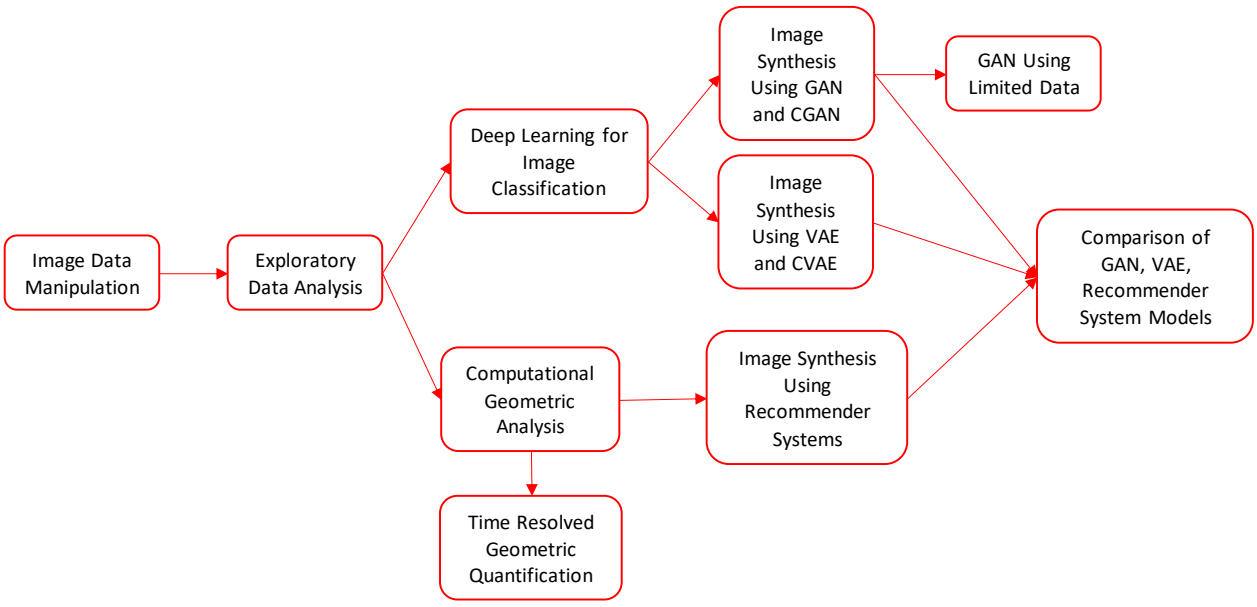

Fig. 1. A flow chart depicting research process

learning to artificially create new data for desired preconditions, where data does not exist for those preconditions. This sequence is depicted in Fig. 1 below. The work described here was performed at NASA Langley Research Center as part of a larger effort in understanding process characterization for the Aeronautics Research Mission Directorate.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview of Metals Additive Manufacturing

Recent advances in metals additive manufacturing (AM) technologies, known more informally as “3-D printing” have allowed superalloys to be developed into complex objects in methods that were previously unavailable. Custom design objects can be tailored by the end user and become available for use after just a few hours. In the healthcare sector, this technology can be especially advantageous for surgeons to be able to make custom tools and have it available for almost immediate use. Work in this area has been progressing such that there are now many kinds of metal based additive manufacturing in use in the healthcare sector [1]. The objects not only have the desired configurations, but also can be built using complex superalloy metals. The fabrication of components using these materials are especially desirable in the aerospace industry as these objects are constantly subjected to extreme conditions when integrated into vehicular applications that may be involved in extreme air-flight, possibly including space flight. For instance, the NASA Aerospace Research Mission Directorate (ARMD) is heavily involved with AM within computational material science research as this technology has allowed metal components to be fabricated in a quicker, and more cost-effective manner than traditional metallurgical processes. This directorate has within its scope the desire to develop technologies for supersonic flight, as well as spacecraft for Moon and Mars exploration missions in the Artemis program. Materials science research is a core component of NASA’s aerospace research mission, which is directly involved with developing structures for space missions. It is especially true in the subfield of additive manufacturing where new complex structures are in constant demand [2]. This is further reinforced under NASA’s Transformational Tools and Technology (TTT) initiative, for

“understanding and development of new types of strong and lightweight materials that are important for aviation” [3], which falls within ARMD. As described by the agency, this project seeks to “develop state-of-the-art computational and experimental tools and technology” that are vital to “NASA’s ability to advance the prediction of future aircraft performance,” and it also “explores technologies that are broadly critical to advancing ARMD strategic outcomes, such as the understanding of new types of strong and lightweight materials, innovative controls techniques, and experimental methods” [4].

Superalloys have traditionally been processed in numerous ways, usually involving metallurgical techniques such as casts and dies, and melting the metal and pouring the liquid material into the required shapes. These objects can then be subjected to machining, which can help shape the superalloy into the desired geometric configuration. However, these techniques can be difficult with some materials that have high tensile strength, or have a set of physical properties that cause it to harden during the machining process. For instance, Inconel-718 (IN718) is a nickel based superalloy that is commonly used for engines, which run at high temperatures, due to its high temperature strength, toughness, and resistance to degradation in corrosive or oxidizing environments [5]. While this set of characteristics makes IN718 a desirable material for such aerospace uses, manipulating and fabricating from it is a costly and time consuming process. output for the complex objects required [6]. Additionally the complexity of developing often cannot be scaled upwards for a high production

Additive manufacturing was established in its earliest forms in the 1980’s, and have now evolved into a variety of tools and techniques [6]. AM is defined by the American Society for Testing and Materials (ASTM) as the process of joining materials to make parts from 3D model data, usually layer upon layer, as opposed to subtractive manufacturing and formative

manufacturing methodologies [7]. Based on this definition that compares AM to subtractive manufacturing, AM has the potential to reduce waste, reduce lead time and cost, and allow the fabrication of complex objects with many intricate design features. For instance, the object shown in Fig. 2 would be impossible to create through any technique other than AM. This object, a rocket nozzle developed by NASA Marshall Space Flight Center, has more than 200 channels built into its wall for regenerative cooling of the nozzle in the extreme environments where this object will be subjected [8]. Developing those small intricate channels would not be possible using a subtractive method. Additionally, a subtractive method would need to start the fabrication with a large solid stock material, which would be cut away to form the desired structure. These cut away portions would be waste; a circumstance that AM does not create.

Additive manufacturing has the ability to achieve lower cost in part development due to net-shape or near net-shape capabilities where an object is built almost exactly as developed. An entire complex part can be built at once, rather than many smaller parts being built and later assembled. Therefore, industries that have a need for complex custom components have begun evaluating AM for their fabrication needs [9]. Since the time and expense of assembling many parts can be mitigated through AM, cost savings scales tremendously when many of the same components need to be built. Cost savings can be even greater when fabricating parts using superalloys, which is often the case for aerospace components. While IN718 is difficult to cut, form, and machine, it is suitable for fabrication through AM processes. Thus, AM can lead to a new way to develop components with superalloy metals, which have certain properties necessary for the physics of flight and intense atmospheric conditions, to be used in aerospace vehicles.

Fig. 2. 3-D printed rocket nozzle made from copper material

2.2 Laser Powder Bed Fusion

Additive manufacturing for metals can be broadly split into two categories of techniques, solid state AM and fusion based AM. The difference is based on the temperatures used to achieve the additive process. Solid state AM does not reach temperatures above the material's melting point, while fusion based AM does cause the material to melt [10]. While solid state AM has become an emerging technology and may offer advantages over fusion based techniques due to the energy requirement being lower – fusion based techniques are the standard in industry.

One of the most commonly used methods for metals AM, a fusion based technique, is known as laser powder bed fusion (LPBF), which uses a layer by layer approach [11]. A diagram of the LPBF process can be seen in Fig. 3. Initially, a layer of a metal powder is spread over a rigid build plate. This metal powder, the substrate, sits beneath a laser which when active has a high enough intensity to melt the individual metal powder particles and thereby cause them to fuse to one another. The laser will move in a route that was predetermined based on a computer aided design (CAD), to outline the desired shape with respect to the layer in progress [12]. For instance, if a solid cube were to be built, a layer in the build process will require the laser to move in a pattern than outlines and fills in a square. Once the area of that square has been completely traced by the laser's movement, that layer will be completed and fusion of the next layer will commence. This begins by the build plate being lowered a slight increment. Then a powder spreader will push a new layer of cool, unaltered metal substrate from a powder stock compartment where the metal substrate is stored, over the lowered build plate. At this point the top of the substrate is horizontally even with the powder stock that sits adjacent to the build plate, with that new layer of metal powder covering the previously fused square shape. The laser now commences to trace the square shape again, fusing the metal particles to one another, and to the layer below. After a layer is fused,

another mechanism sweeps off excess material into a waste compartment (where the powder can later be sifted and recovered). If a more complicated shape were to be built, each layer may not be exactly the same, as with the cube example which would consist of many squares sitting atop one another consecutively. The layer depth, that is, the height that the powder bed drops after a layer is built, can be predetermined by the CAD, and set to a desired specification at any desired interval at any portion of the build. This allows structures to be built with intricate details.

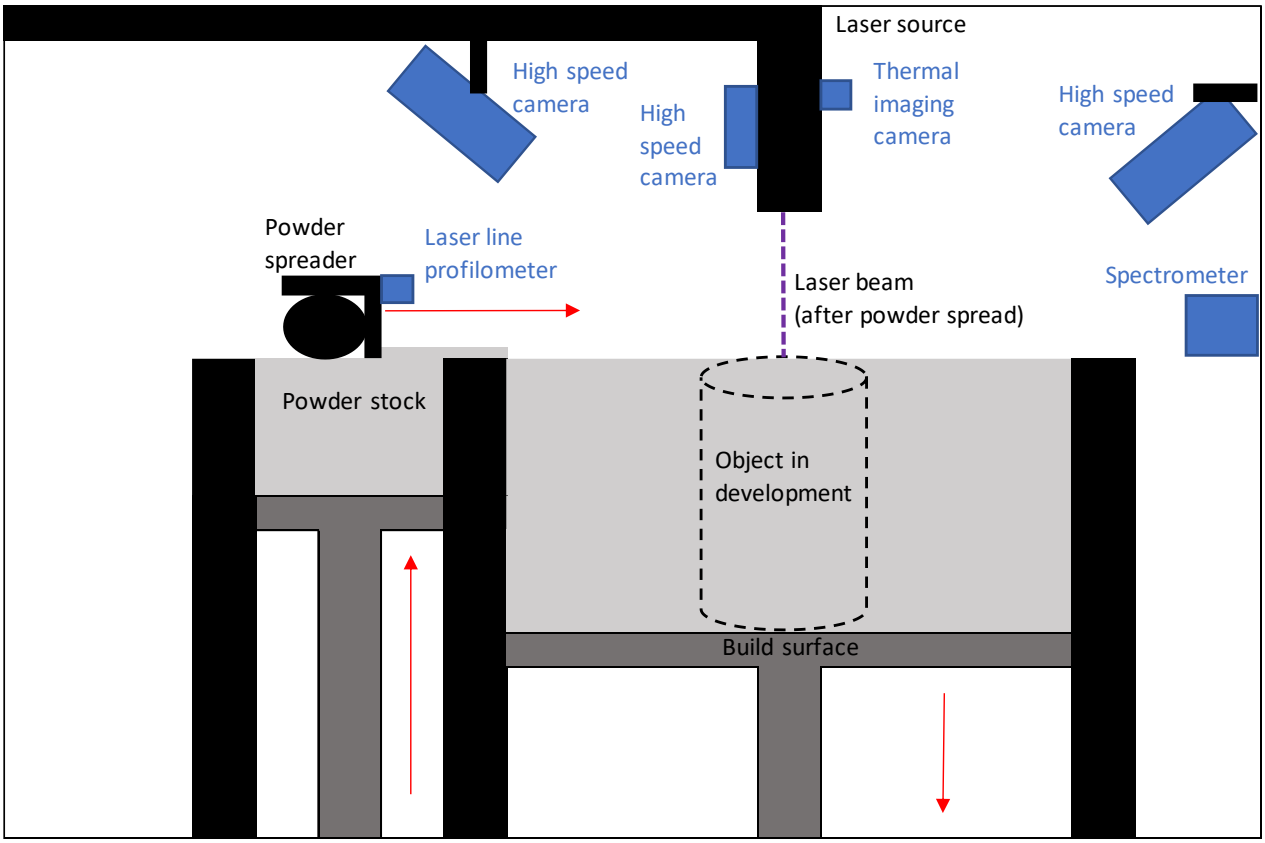

Fig. 3. LPBF process setup

2.3 Material Microstructures

The LPBF process requires many input parameters, each of which will affect the build quality. Some of the main considerations include the laser intensity or power, the laser velocity, the layer thickness, the laser spot size, hatch spacing (the distance the laser traces two adjacent lines in the build pattern) and the build atmosphere among others. In this section we will explore the main parameters that effect the build quality in terms of the microstructure of the material, which will consist of a variety of physical characteristics in the micrometer scale. As the physical characteristics of the material can affect its physical properties, evaluating the process parameters for builds is paramount to certifying AM parts for their use in real-world functions.

One way of combining multiple parameters is to use a metric known as laser energy density, which mathematically combines the laser power, scan speed, hatch spacing, and layer thickness into a single parameter *E* [13]. This can be seen in the equation:

$$E = \frac{P}{vht} \tag{1}$$

where *E* is the laser energy density in units of J/mm^3, *P* is the laser power in watts, *v* is the laser scan speed in millimeters/second, *h* is the hatch spacing between adjacent laser line tracks in millimeters, and *t* is the thickness between vertical layers. In this expression, the hatch spacing value for *h* can be replaced with the laser spot size. Using the laser energy density, which expresses the energy density delivered per unit volume of powder, many studies have shown that relative energy density affects the structure of the material during the build [14]. For instance, micro-hardness increases as the linear energy increases. And as energy density increases, the physical properties of wear resistance and oxidation resistance increases. This shows that laser energy density, which is a function of many of the underlying build parameters, plays a clear role in the overall quality of the build in terms of the physical characteristics and properties of that build.

2.4 Microstructure Defects

The LPBF process has known qualities in terms of the microstructure of IN718 when that material is used. IN718 commonly exhibits a columnar dendritic pattern in its microstructure where these columns tend to extend through multiple layers of the build [15]. The laser speed can determine how far apart these columns are, as well as how deep they penetrate into the subsequent layers. And while these qualities have been examined previously, there is still a lack of a capability to predict geometries, and an additional unknown quality comes from defects that are formed from the AM process.

While the AM process is relatively quick and scales well when there are many objects to be built, there are some inherent challenges unique to LPBF AM compared to traditional die and casting methods. These challenges can involve defects which effect the material microstructure – which is responsible for the physical qualities of that component. For instance, certain defects that are introduced in the AM process can lead to a lower fatigue profile of that object, which can impact its lifespan, and therefore limit its usability in adoption into an aircraft [16]. Some of the main considerations are described here:

- Too high of a laser power setting that causes the metal powder to instantaneously evaporate into a gas, creating a bubble in the solid structure (known as a keyholing defect)
- Too low of a laser power setting that can cause the particles to not correctly form the required shape (known as a lack of fusion defect)
- Too much heat buildup from the laser in corners or crevices of complex shapes that can cause too much of the metal powder to melt or not fuse properly
- The laser moving too fast, not giving the material enough time to properly fuse

- The laser moving too slowly, causing additional heat buildup

During the AM process, the area directly beneath the laser turns to a gas, also known as the vapor depression or vapor cavity. Meanwhile the area immediately surrounding that turns to a liquid, known as the melt pool (see Fig. 4). These features will solidify and form the material microstructure [17], with those aforementioned defects present in that internal structure. Computational fluid dynamics can provide some insight into how these regions will behave theoretically, such as the Marangoni-Gibbs effect (which describes the motion of two fluid bodies with a gradient with surface tension in between), buoyant forces (which describe the upward force by a fluid where it opposes an immersed object or other fluid of differing density), recoil force (which describes a fluid's ability to revert back to its previous position once an external force is removed), vapor dynamic force (which describes the forces of the molecules in a fluid during a phase change), and hydraulic pressure (which describes the pressure that a fluid exerts in all directions against the outer walls of a vessel) [18]. These factors, along with other physics-based models, can inform vapor depression and melt pool geometries; however, there are limitations in the ability to validate those models without extensive high-performance computational processing – often a resource prohibitive effort. This presents a major limitation in understanding the performance, lifespan, fatigue potential, and physical features of an AM build [16].

2.5 Physics Validation for Thermal Modeling

In the LPBF process, the liquid melt pool is governed by five major forces: Marangoni force, recoil pressure, vapor dynamic force, buoyance force, and the hydraulic pressure. Each of these influences the size and shape of the melt pool during the build, which therefore determines the solidified structure once the material cools [18]. The underlying thermal physics of these forces

can be described through physics-based modeling, however, validating these models for the LPBF builds requires extensive data which is not fully available.

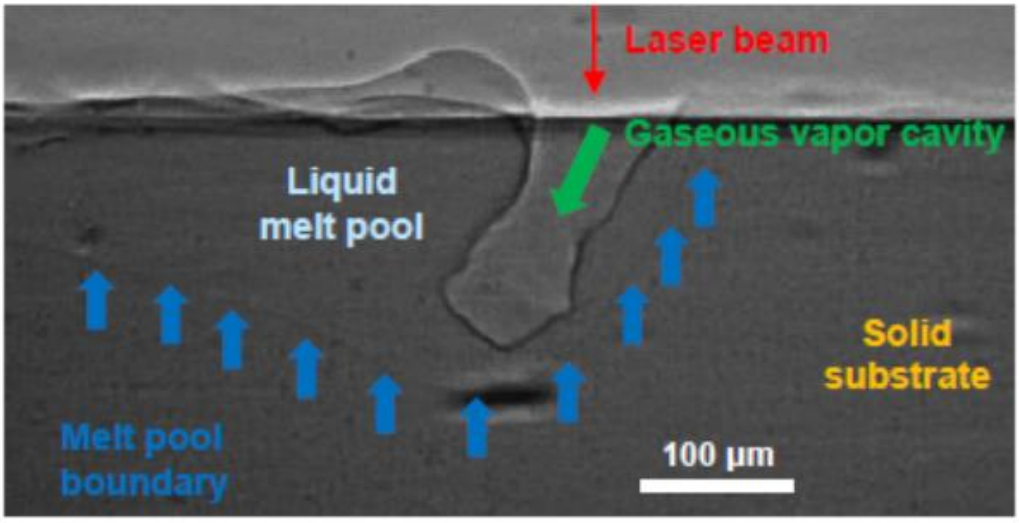

Figure 4. Dynamic x-ray image of in-situ LPBF process

The first force, the Marangoni force, describes the flow from an area of a higher temperature to an area of a lower temperature in a material that has a negative temperature coefficient of surface tension [19]. This can affect two bodies, such as the melt pool and the solid substrate, and determines how the inner surface of the melt pool moves along with the surface of the substrate with respect to the boundary in between those two bodies. The next force, the recoil pressure, describes a force that exerts an inward pressure that is directionally normal to the surface [20]. This effect impacts the melt pool geometry based on the activity of the vapor depression,

formed at the site of the laser during the build. The vapor dynamic force describes the force through friction at the boundary between the liquid melt pool and the gaseous vapor depression [21]. As the vapor depression - the area that instantaneously evaporates – moves, it effects the flow of the liquid melt pool and the boundary between those two bodies has a friction similar to the force of friction between any other body moving against another. The buoyance force describes buoyancy, in this case, the force that drives the liquid melt pool along a density gradient [22]. The density gradient here is determined by the immersion of the melt pool within the solid material, and that liquid's ability to move along the upward thrust of that fluid as it floats within and above the substrate. Finally, the hydraulic pressure describes the energy that is exchanged by hydrostatic pressure. This force influences the liquid body based on the force it exerts on the outward boundaries around it [23].

The extent of the Marangoni force is relevant in the melt pool as this region is dominated by the thermal gradient in the material caused by the laser and the resultant cooling. The liquid moves from the areas of higher temperature to lower temperature, which therefore means that the liquid is moving away from the highest intensity area, the boundary of the melt pool and the vapor depression – which itself is the area closest to the heat source, the laser [24]. What then transpires is that while the liquid areas ahead of the material moves forward, the area behind the laser moves backward, creating a cyclical pattern of motion. Then, when the flows reach their respective edges of the melt pool, they will flow downwards, and the effect of the hydraulic pressure will influence those flows to return back to the area closest to the laser heat. The first flow that originally started ahead of the laser, moves in a clockwise direction, while the liquid behind the laser moves in a counterclockwise direction [24].

The above described forces that govern the motion of the melt pool, which are associated with the heat and therefore the kinetic energy of the system, can be described mathematically. This takes into consideration the extent of the damping of the flow of the liquid, which is influenced by the Marangoni force. This can be expressed by Weber number (*We*) as:

$$We = \frac{\rho v^2 L}{\sigma} \tag{2}$$

where ρ is the liquid density, v is the velocity of the liquid flow, L is the length, and σ is the surface tension [25]. As seen in this expression, among the multiple variables involved, the larger the length of the material, the greater the Weber number will be. Depending on how high this value is, for both the leading area and the trailing area separately, it can be determined whether or not these two regions will dampen out and create smoother areas along the melt pool surface rather than ripples or waves in the material [14].

In addition to the effects described above, heat convection must also be included, based on the effects of thermal energy on liquid motion. This has significance for the area ahead of the laser intensity, the forward region of the melt pool, according to heat transfer theory [26]. The relative effect of heat convection over the effect of heat conduction can be calculated as the Péclet number (*Pe*), which itself is the product of the Reynolds number (*ReL*) and Prandtl number (*Pr*), and is expressed as:

$$Pe = Re_L * Pr = \frac{\rho v L}{\mu} * \frac{c_p \mu}{\alpha} = \frac{Lv}{\alpha} \tag{3}$$

where L is the length, v is the velocity of the liquid motion, and α is the thermal diffusivity. Note that in both equations above, the value of L is determined by the diameter of the melt pool.

Discussed above are some of the major factors that influence the melt pool in the LPBF build, which is associated with the vapor depression. The liquid thermodynamics discussed influence the motion of these bodies, which upon cooling, will determine the internal structure

based on the location of the boundaries of these bodies when they cool. The flow pattern therefore directly impacts the internal microstructure of the final build, which determines its strength among other physical qualities. While there are vast amounts of images for each build there is a limitation on the variety of the data available. Measurements of the size of the vapor depression, and melt pool, provides the *L* value above. Yet, this measurement is only available when a human manually examines an image taken post-processing, which only gives a value at one specific instant in time. Additionally, the manual measurement is a laborious task.

2.6 Deep Learning with Convolutional Neural Networks

An analysis of the types of microstructures should begin with understanding the types of vapor depression geometries that influence those microstructures. If provided an abundant set of image data similar to the image in Fig. 2.3, image classification could be employed to better characterize the various in-situ objects, particularly the size and shape of the vapor depression. Within the field of machine learning, one particular algorithm that has been developed and has been able to advance computer vision capabilities is the convolutional neural network (CNN). CNNs are a class of deep learning, that is, artificial neural networks with an input layer, an output layer, and many hidden layers [27]. The approach is to take an input image, learn features from it, assign importance through weights and biases, and identify different objects and features in the image or across multiple images. As the name implies, the hidden layers perform a convolution operation on the input data, which mathematically is a sliding dot product. Traditionally, CNNs have a more complex network architecture compared to the multilayered perceptron model of artificial neural networks [28]. Multilayered perceptrons generally are fully connected networks, which means that each neuron or node in a single layer is connected to all of the nodes in the next layer, and so on [29]. However, a drawback to this integration of every node to every node is that

it can cause model overfitting, which refers to a situation where the output of a model matches the input data too closely and therefore cannot make additional predictions reliably from future data. While regularization is a technique that can incorporate a loss function and add weights accordingly to limit over-fitting – CNNs resolve this by learning from patterns in the data by partitioning the data into smaller and simpler sets, from which it can extrapolate to larger and more complex patterns [30]. The method of learning and the mathematics behind CNNs allow them to function on image data with little pre-processing, unlike other pattern recognition or image segmentation approaches where data processing can be manually taxing.

The CNN architecture is designed such that many layers are essentially stacked to transform an input to an output through a differentiable function. As there are many layers in a CNN, hence being commonly categorized into the group of machine learning called deep learning, these different layers can have different effects on the learning itself. The main operation of the CNN is the convolution that occurs within the network's hidden layers [28]. The first convolution layer takes the input data, an image, and applies a kernel or filter over that image. The kernel is defined by a predetermined length and width, which therefore gives it a square shape known as the receptive field – or, how much area that this kernel covers on the input. The kernel moves across the pixels of the input image, which are read by the convolutional layer in a tensor of shape equal to the number of images multiplied by the image heights and image widths. The kernel passes along the input image with a predetermined stride, which signifies how many pixels the kernel will move at a time. Thus, if the stride is set to 2, the kernel will move two pixels at a time. This can cause overlapping, where some pixels are involved in one stage of the convolution with the kernel at one location followed by also being incorporated into the next area of pixels to be read when the kernel moves. The kernel moves to the right of the image at the defined stride until it has passed

through the full width of the image. At this point it then moves to the beginning at the far left of the image in the next row of pixels and moves to the right again with the same stride value. Repeating this maneuver, the kernel will eventually scan every pixel/area of the image. During the forward pass of the kernel over the input image, a 2-dimensional feature map is created where the network learns specific features at each location in the image [31]. This allows it to learn what features are most important at each area, and therefore allows the model to learn the relevance of spatial positions of values or features in the input.

Each node in the convolutional layer processes data for its receptive field, which contrasts slightly with other feed forward artificial neural networks [27]. While the function that is applied to the input data is determined by weight and biases in both CNNs and other feed forward networks, in a CNN many nodes can share the same filter and therefore can share the same weights and biases. A single weight and a single bias factor will be used for many receptive fields that share that filter, instead of each receptive field having to calculate its own bias and weight factor. If not processed this way, since each pixel would represent an individual variable, an image could yield thousands or even millions of variables to be processed [32]. For instance, if an image of one megapixel were used as an input, there would be one million variables and therefore weights to be applied to the model. However, the convolution operation described above makes this simpler by reducing the number of parameters and applying the same shared weight, allowing the network to be deeper (i.e., have more hidden layers) with far fewer learning parameters. This helps the CNN both reduce the images into forms that are easy to process without losing significant features in the underlying data, and also be able to make reliable predictions that can be scaled to large datasets with many images [33]. The first convolutional layer will generally learn the most important or most generally applicable features in the image – such as object edges. Subsequent layers then

essentially learn more details such as the orientation of edges from one area of an object to another. Through more and more layers the model can learn more specific features such as pixel gradients at certain locations until finally the model can have enough information to eventually make predictions from those features [34].

Convolutional neural networks also involve layers known as pooling layers which reduce the size and dimensionality of a layer prior to the next convolution layer. Using pooling layers in CNNs helps the model to learn the rough location of features relative to other features of importance. The pooling layer reduces the overall size of the input data fed at that layer which helps reduce the computation resources, and also helps to reduce model overfitting [32]. This operation reduces the dimensions by combining the outputs of node clusters into a single new node, which can be done both locally (in small segments), or globally (to every node in that layer). Max pooling is commonly used, where the maximum value from a group of neurons is used as the input for the next node in the subsequent layer. Alternately, average pooling can also be used where the average value from a group of nodes is used as the input value for the new node in the next layer. However, max pooling performs as a better noise reduction mechanism since it takes the highest value in the region it scanned and discards the other values. Since a noise value would be included in the average, average pooling merely reduces or suppresses noise while max pooling can eliminate it [30].

Similar to multilayered perceptrons where every layer is fully connected, this operation is involved in the CNN architecture, albeit at a smaller scale since not every layer is fully connected. Fully connected layers can be implemented, where for that layer, all nodes are connected to all nodes in the next layer. Therefore, in a fully connected layer, the receptive field is the entire previous layer.

The CNN's classification mechanism whereby the CNN makes a prediction for a class or label is accomplished through a Softmax function [28]:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \quad (4)$$

where $\sigma(z)_i$ is the output class probability, *N* is the number of potential classes, and *z* is the input vector. This function computes the exponential value from the input such that is can normalize the data into a set of probabilistic distributions with values that will sum to 1, that is, $\sum_i^N p_i = 1$. The individual probabilities, p_i will be between 0 and 1. This allows the network to make predictions for multiple possible classes rather than a binary classification scheme in which there will only be two possible classes.

Convolutional neural networks have been used in many industries to perform some sort of image recognition task, among many other uses. For instance, in the healthcare field CNNs have been used to detect certain forms of cancer, with precision rivaling that of qualified medical health providers [35]. They can not only detect the presence of a tumor from medical imaging scans, but also classify it by type. They are also one of the main functions for self-driving cars that must process a continuous feed of images through real time video recording in order to autonomously drive a vehicle through situations with potentially hundreds of objects to recognize at any given time [36]. Other uses for CNNs can include virtually any application where there is more data than is feasible for a human to process manually, with the goal of identifying, categorizing, or otherwise gaining some kind of understanding from that large data set.

2.7 In-Situ Characterization Challenges

Without a thorough way to evaluate the internal structure of those AM builds, while also considering the characteristics of the defects embedded in the structure, these components will not be suitable for application into aerospace systems [12]. Material microstructures can be evaluated

using microscopy tools, but this can only be done after the build. There has been limited research into how the process is affecting, or generating, the microstructure of the material while the build is in process. This is mainly due to practical limitations in any potential experimental setup. For instance, in order to examine the LPBF process in-situ, an imaging device would have to monitor the build in real time, capturing the motion of the laser moving at high speeds (across a surface in a scale of centimeters at a velocity scale of meters per second), and at very small distance scales (in the micrometer range). Data can be collected in this fashion, using advanced imaging techniques, but the cost is prohibitive and therefore it is not feasible to routinely collect this data, nor is it feasible to build a large enough library of various in-situ builds for every possible combination of process parameters. Additionally, many of the manifestations of microstructures and defects from the AM process is material dependent, and therefore a catalogue of images for builds would also have to include many different materials to be useful. This presents a major challenge for materials science research into AM. The lack of data for defect characterization prevents a thorough understanding of the effects of laser settings on a build.

2.8 Missing Data for Process Characterization

The proposed research will begin by examining the problem of gaps in the data in the experimental datasets. There were 35 experimental builds from which in-situ data was collected. While this yielded a large volume of image data, this is not an adequate sample size of process parameters to determine optimal build settings in the LPBF process. Research is ongoing into the desired vapor depression and melt pool sizes that yield builds with the optimal fatigue and lifespan profiles, and the data collected does not give enough insight to make that determination. For instance, at a laser intensity of 350 watts and a velocity of 0.2 meters per second, it can clearly be seen that the vapor depression forms bulbous regions in its bottom tail area, which in turn leads to

keyhole defects. Yet at the same intensity, 350 watts, but at a velocity slightly higher at 0.4 meters per second, it can clearly be seen that the vapor depression remains stable and leaves behind no defects (see Fig. 5). As described previously, it is infeasible to run experiments for every interval between 0.2 and 0.4 meters per second at 350 watts. To then be able to collect data for every power setting further compounds the problem. Therefore, this results in a missing data problem.

Fig. 5. In-situ defect generation

2.9 Matrix Completion Overview

Matrix completion can be used to approximate or generate lost or missing data. This can be done in conjunction with computational approaches for taking past events and making recommendations via computational algorithms. These techniques, known as recommendation or recommender systems (a sub-field of machine learning) have attracted a lot of attention in both research and practice, since they are able to narrow complex, difficult decisions into a few recommendations, which makes this approach particularly attractive in e-commerce applications where a retailer or service provider seeks to match potential items to its users [37]. The following discussion will focus on a commercial use-case, particularly matching users to items, but can be

generalized to other applications. While the most common application might be recommending items for people to purchase, the recommender system has been used in other domains such as bioinformatics [38], systems modeling [39], and engineering [40] among many others.

Generally, recommendation systems are a subset of the information filtering systems, whose goal is to predict the rating a user would give to an item of commodity. The recommendations are typically made through either content-based filtering or collaborative filtering approaches. The content-based filtering approaches utilize a set of discrete features characterizing a commodity and build a user profile indicating the items this user likes in the past. Then, items with similar properties as those the user likes in the past are recommended. Instead of using item features and user profiles, the collaborative filtering approaches produce recommendations based on a user as well as the other users' past behaviors. The fundamental assumption under collaborative filtering is that if the users share similar ratings in the past on the same set of items, then they will likely rate the other items similarly. Content-based filtering and collaborative filtering can be combined to build hybrid recommendation systems, which often demonstrate better recommendation precision than pure recommendation approaches [41].

Typically, a collaborative filtering scenario in recommendation system can be modeled as a matrix completion problem. Given a list of m users $\{u_1, u_2, \dots, u_m\}$ and n items $\{i_1, i_2, \dots, i_n\}$, the preferences of users toward items can be represented as an incomplete $m \times n$ matrix A, where each entry either represents a certain rating or is unknown. The ratings in A can be explicit indications, such as scores given by the users in scale 1-5 or ordinal favorability (e.g., strongly agree, agree, neutral, disagree, strongly disagree). These ratings can also be implicit indications, e.g., item purchases, website/store visits, or link click-throughs. It is generally assumed that no more than one rating can be given by a user for a specific item. As a result, recommendations can

be made by filling out the unknown entries and then ranking them according to the predicted values.

Denoting Λ as the complete set of N entries in A with known ratings, the general matrix completion problem is defined as finding a matrix R such that

$$R_{ui} = A_{ui}, \tag{5}$$

for all entries $(u, i) \in \Lambda$. In addition, we denote $\bar{\Lambda}$ as the complement set to Λ and $P_{\Lambda}(A)$ as an orthogonal projector onto Λ which is an $m \times n$ matrix with the known elements of A preserved and the unknown elements as 0 [42]. However, since the number of known entries is less than the overall number of entries, there exist infinitely many solutions. Nevertheless, it is commonly believed that there exist only a few factors as latent factors [43] influencing how much a user likes an item. For example, studies show that the attributes of actor/actress, director, and decade contribute most to a user's preference to a movie. These relatively small number of influence factors compared to the total number of users or items in the rating matrix A provides a guiding framework to fill in the missing values and to select the correct complete matrix. This corresponds to the low-rank assumption in matrix completion, i.e., the rating matrix A is low rank or approximately low rank. The low-rank assumption in matrix completion also agrees with the well-known Occam's razor principle in machine learning, whose goal is to find the "simplest" complete matrix X that is consistent with the known ratings in A.

The fundamental assumption is that there exists a low-dimensional representation, although probably unknown, of users and items, which can be taken advantage to model user-item association accurately. Such low-dimensional representation is often characterized by a low-rank matrix. We also study models employing various regularization methods and incorporating various

constraints in the completed matrix. Denoting μ as the average rating among all known ratings in the rating matrix A, the baseline model [44] fills out a missing element R_{ui} by

$$R_{ui} = \mu + b_u + b_i, \tag{6}$$

where b_u and b_i represent the observed deviations of user u and item i from μ, respectively. The training parameters b_u and b_i can be estimated by solving the following least squares problem

$$\min_b \|P_\Lambda(R) - P_\Lambda(A)\|_F^2 + \lambda(\sum_u {b_u}^2 + \sum_c {b_i}^2), \tag{7}$$

where λ is the regularization parameter. The first term $\|P_\Lambda(R) - P_\Lambda(A)\|_F^2 = \sum_{(u,i)\in\Lambda}(R_{ui} - A_{ui})^2$ attempts to minimize the training error while the second term $\lambda(\sum_u {b_u}^2 + \sum_i {b_i}^2)$ serves as the regularizing term to avoid overfitting by penalizing the magnitude of b_u and b_i.

The fundamental idea of the SVD model is to decompose the rating matrix A into a user feature matrix, a singular value matrix, and an item feature matrix of low-rank [45]. Starting from a normalized matrix A_{norm} by filling out the missing elements with preliminary, simple predictions, the SVD model carries out a Singular Value Decomposition (SVD) operation on A_{norm} such that

$$A_{norm} = U\Sigma V^T, \tag{8}$$

where Σ is a diagonal matrix with descendently sorted singular values deposited in its diagonal entries and the columns of U and V contains the corresponding left and right singular vectors, respectively. Truncating the diagonal matrix Σ to a top-r rank Σ_r, then $U_r\Sigma_r^{\frac{1}{2}}$ and $\Sigma_r^{\frac{1}{2}}V_r$ represent the latent factor vectors for users and items, respectively. The dot product of the uth row of $U_r\Sigma_r^{\frac{1}{2}}$ and the ith row of $\Sigma_r^{\frac{1}{2}}V_r$ yields the prediction of the rating that the the uth user will give to the ith item.

The matrix factorization model is a generalization of the SVD model, which intends to find a low-rank matrix factorization to approximate A. Assuming an r-dimensional vector x_u associated with each user u measuring the latent factors of u has in items and an r-dimensional vector y_i associated with each item i representing the latent factors of i, the matrix factorization model uses the dot product ${y_i}^T x_u$ to capture the correlation between user u and item i. The predicted rating then becomes

$$R_{ui} = {y_i}^T x_u. \tag{9}$$

Assuming the columns of X and Y contains all x_u and y_i vectors, respectively, the goal of matrix completion is to estimate:

$$R = Y^T X. \tag{10}$$

The parameters to be learned are the user feature vectors x_u and the item feature vectors y_i, which can be done by minimizing the Frobenius norm error as follows:

$$\min_{x_*, y_*} \|P_\Lambda(R) - P_\Lambda(A)\|_F^2. \tag{11}$$

In order to avoid overfitting the observed user-item ratings, the regularized matrix factorization method uses *l*2-norm to regularize the learning parameters by penalizing their magnitudes. Based on the matrix factorization model this can be done by minimizing the regularized *l*2 norm error of x_u and y_i in addition to the Frobenious norm error term as follows:

$$\min_{x_*, y_*} \|P_\Lambda(R) - P_\Lambda(A)\|_F^2 + \lambda_1(\sum_i \|y_i\|^2 + \sum_u \|x_u\|^2)), \tag{12}$$

where λ_1 is a constant controlling the extent of regularization.

A more sophisticated *l*2-regularized matrix factorization model can be built on top of the baseline model by considering the user deviation b_u and the item deviation b_i. Then, each predicted rating $\hat{R}_{ui}$ in $\hat{R}$ then becomes:

$$\hat{R}_{ui} = \mu + b_u + b_i + {y_i}^T x_u. \tag{13}$$

The parameters to be learned become b_u, b_i, x_u, and y_i, which can be done by minimizing the regularized *l*2 norm error as follows:

$$\min_{b_*,x_*,y_*} \left\| P_\Lambda\left(\hat{R}\right) - P_\Lambda(A) \right\|_F^2 + \lambda_2 \sum_{(u,i)\in\Lambda} (b_u^2 + b_i^2 + \|y_i\|^2 + \|x_u\|^2), \tag{14}$$

where λ_2 is the regularization parameter. Due to fact that there are more training parameters, this model often yields prediction accuracy improvement.

Upon achieving a completed matrix, this procedure can be incorporated into the recommender system, such as by using the Alternative Least Square (ALS) algorithm which is designed for the *l*2-regularized matrix factorization model. However, due to the term ${y_i}^T x_u$ for calculating R_{ui}, the objective function is non-convex and optimizing it is NP-hard. Nevertheless, if x_u can be fixed by treating its variables as constants and then the minimization objective becomes a convex function of y_i [43]. Alternately, y_i can then be fixed by treating its variables as constants and then the objective becomes a convex function of x_u. Therefore, in ALS, when one is fixed, the other is calculated, and this process will repeat until convergence is reached. This derivation process for the user vectors x_u for all u can be expressed as:

$${x_u}^{(j+1)} = {Y^{(j)}}^T \left(Y^{(j)} {Y^{(j)}}^T + \lambda_1 I_r \right)^{-1} {x_u}^{(j)} \tag{15}$$

and similarly, the process for calculating the item vectors y_i for all i is:

$${y_i}^{(j+1)} = {X^{(j)}}^T \left(X^{(j)} {X^{(j)}}^T + \lambda_1 I_r \right)^{-1} {y_i}^{(j)} \tag{16}$$

where I_r is an $r \times r$ identity matrix.

Computationally, the ALS algorithm is desirable due to its scalability to large datasets. The process of using two loss functions alternatively allows it to run its computation in parallel. For some datasets where the combination of users and items can reach into the billions, using an optimization technique such as stochastic gradient descent would not be feasible. Rather, through

ALS re-computing the user-factors and item-factors, each step is guaranteed to lower the value of the cost function [46]. Outside of the e-commerce applications previously noted, recommender systems have been employed for computational drug repositioning research; an approach to take advantage of known drugs to identify new treatments [38]. This work modeled the drug repositioning problem as a recommendation system to discover new disease indications for drugs. The related data sources and validated information of drugs and diseases were integrated to construct a heterogeneous drug-disease interaction network. Then, the heterogeneous network was represented as a large adjacency matrix where the unknown drug-disease associations were presented as blank entries. Finally, the recommender system algorithm was used to complete the drug-disease adjacency matrix with predicted scores for unknown drug-disease pairs.

The results of this work described above suggests that the recommender system approach can be applied to the experimental datasets to numerically recommend, or predict, new vapor depression geometries for depth, width, area, or any other physical measurement – based on the laser process parameters.

2.10 Variational Autoencoders Theory

While the recommender system can yield numerical results for vapor depression geometries for process parameter settings not experimentally performed – generative modeling has also been shown to yield new image based representations from experimental data. An alternate approach to computational modeling based on latent feature derivations is to use a deep learning approach known as autoencoders. As previously discussed, the recommender system is adept at finding the latent features in a dataset by which future predictions can be made. Whether its product features that influence a consumer's buying habits, or the underlying latent features that impact a LPBF build – which will correspond to the underlying thermophysics discussed in

Section 2.4. Autoencoders take a different approach to learning however, which will be explored in this section. The ultimate goal will be to use a specific type of autoencoder, the Variational Autoencoder (VAE) as a generative model to not only predict vapor depression geometries, but generate new vapor depression expressions visually in new images [27].

Autoencoders can broadly be categorized into a few types. Traditional autoencoders are relatively simple neural networks (based on their architecture), with the exception that there are two major segments, the encoder and the decoder. In CNNs, the convolution step takes in an image; for example a rank 3 tensor of size 300 x 300 x 3 for the dimensions of the image and the 3 color channels. It then converts this to a much more compact, denser representation, such as a rank 1 tensor of size 1000. This resultant dense representation is then used by the fully connected layer to make a classification for the image based on the features learned throughout the previous hidden layers. Similarly, the autoencoder takes in an input and produces a smaller, denser representation [47]. This is the goal of the encoder portion of the network, the original data is encoded in this smaller representation (similar to compression). The encoder and the decoder can be expressed as:

$$\theta : X \rightarrow F \tag{17}$$

$$\varphi : F \rightarrow X \tag{18}$$

where the encoder function, θ maps the original data, X to a latent space F, and the decoder function φ, maps the latent space F to the output (see Fig. 6). In this regard, the output is expressed as the same as the input as the goal is to reconstruct the original input after non-linear compressions. The encoding portion of the network can be represented similarly to a basic neural net's activation function such as:

$$z = \sigma(Wx + b) \tag{19}$$

and the decoder portion of the network can be expressed similarly, just with different weights and biases being used [48]:

$$x' = \sigma'(W'x + b') \tag{20}$$

The loss function is going to be used to train the autoencoder neural network through backpropagation similar to standard neural networks [27]:

$$L(x, x') = \|x - x'\|^2 = \|x - \sigma'(W'(\sigma(Wx + b)) + b'\|^2 \tag{21}$$

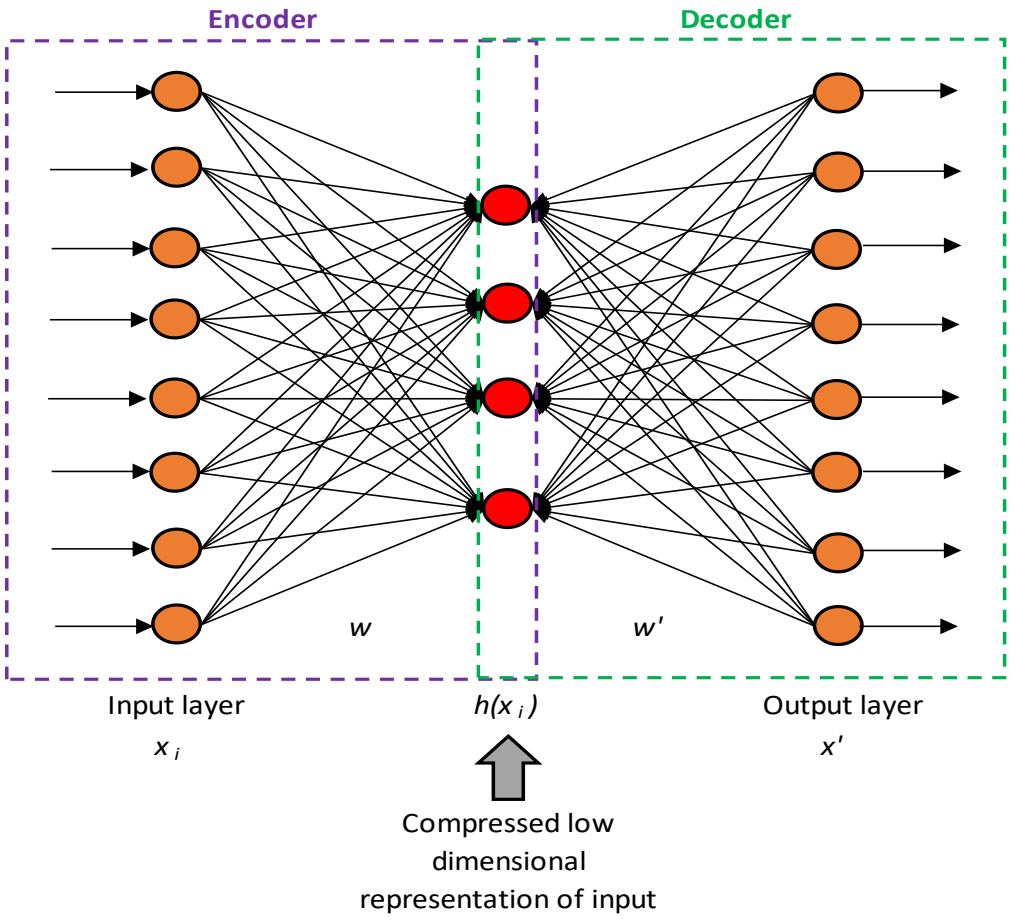

Fig. 6. Basic autoencoder network

While CNNs make classification predictions from the dense representations, autoencoders use a portion of the network called the decoder to attempt to reconstruct the original data, its input. When the network fails to appropriately reconstruct the original data, this is known as reconstruction loss, for which mean squared error or cross entropy between the output and the input is calculated. The network is then penalized for deviations from the output to the input. As the encoding layer (i.e., the output of the hidden layer in the middle of the network) has less information than the original input data, the encoder must have discarded irrelevant information and learned relevant information such that the decoder can learn to take that encoding and properly reconstruct the original data. This could come in the form of an image reconstruction if the original input was an image. Thus, the aim of the autoencoder network is to find an optimal solution whereby the most minimal amount of information is used to encode the image such that it can be reformed on the other end of the network as close to the original image as possible.

The standard autoencoder can be a useful tool for data denoising, as noisy elements would likely not be reconstructed by the decoder. Other applications for experimental data are somewhat limited. An improvement upon this standard model comes in the form of the Variational Autoencoder (VAE), which are a class of autoencoders that improve upon the model's ability to sample from a latent feature space and can then be used to generate new representations of the input data from that latent feature space. The standard autoencoder neural network is capable of compressing images and reconstructing them effectively, but has weaknesses in terms of what reconstructions are capable of being produced from the latent space. For instance, when images are encoded, there would likely be clusters in a feature space representing objects that were encoded of similar type. But if there are gaps in the latent space then the network does not know

what image representations would look like at that location in the latent space, similar to a neural network not having training data for a certain type of observation [28].

VAEs are said to have a continuous latent feature space, from which random sampling can produce variations of the encoded input data [49]. Its output from the encoder is not just a vector of size *n*, rather it produces outputs of two vectors of size *n*, a vector of means, and a vector of standard deviations. Together these factors, μ and σ are essentially the parameters of a new vector representation of random variables that have the length of *n*, with the *i*th element being the mean and standard deviation of the *i*th random variable which is sampled in order to receive the encoding that was manipulated by the decoder. The mean vector represents a central point in the latent space where the sample should be taken around in that latent space [27]. Meanwhile the standard deviation vector represents the area, a circular region around that central point in the latent space. Encodings can then be generated by the network anywhere inside this region of space defined by the mean and standard deviation vectors, thus the decoder learns the features not from a single point (as might be the case in the standard autoencoder), rather it learns from all the nearby points in that space. As a result, the decoder is then able to decode encodings that vary, not just specific encodings, as it has been able to learn from a range of variations of the encoding of an input.

In general, the goal for the encodings is to have them be as close together as possible while still being distinct enough to tell them apart from the groupings in another region. Having this amount of close distinction is what allows for the model to construct new samples. This is accomplished by incorporating information theory, to quantify how much information is in the data, or entropy, which is expressed as [50]:

$$H = -\sum_{i=1}^{N} p(x_i) * \log p(x_i) \tag{22}$$

The entropy helps to understand how much information is contained in the data, but this can be further modified to also quantify how much information is lost when an observed distribution is substituted for a parameterized approximation. Essentially, the goal is to measure the divergence between two probability distributions. The calculation for entropy can be modified to calculate the relative entropy, also known as the Kullback-Leibler (KL) divergence [51]:

$$D_{KL\,(p||q)} = \sum_{i=1}^{N} p(x_i) * \log \frac{p(x_i)}{q(x_i)} \tag{23}$$

where the KL divergence is the expectation of the log difference between the probability of data in the original distribution with the approximating distribution. For instance, if p(x_i) = q(x_i) then the ratio of the two values is always equal to 1, and the log(1) = 0. Therefore, the goal of the VAE here is to minimize the image loss while simultaneously minimizing $D_{KL}(N(\mu(I), \sigma(I)) || N(0,1))$.

The KL loss is effective at creating a latent space where the encodings will be packed in a dense region, randomly, near the center of the latent space. However, using just the KL loss will be problematic for the decoder because it cannot decode anything meaningful from sampling in this region. This is overcome by optimizing both the KL loss and the reconstruction loss together, which thus results in a latent space that maintains the similarity of the encodings locally through the clustering, and globally at the same time through the dense packing near the latent space origin [51].

In a VAE the values for μ and σ can come from a wide range of values; there are no limits on what these vectors can be. This allows the encoder to compute a value for μ that may be very different for different clusters of observations in the latent space. Values that vary greatly can represent clusters far apart, yet the value of σ can be minimized such that the encodings do not very greatly from the same. This lowered uncertainty for the decoder operation is what helps it to accurately reconstruct the training data [52]. The clustering formed by the reconstruction loss and the dense packing formed by the KL loss results in distinct areas that the decoder can successfully

decode. If a new sample is to be taken halfway between two samples, the algorithm finds the difference between their mean vectors, adds half of that to the original, and then decodes.

Prior work with VAEs suggest that these models can learn the latent features of images to then predict new images, or otherwise depict complex structures. While no work currently exists in developing VAEs for material science research, there has been research into applying this model towards structures such as molecular structures [53]. Given the ability of these models to learn latent features, and the likelihood that such latent features are governing the dynamic representations of the vapor depression geometries – VAE should be capable of computationally predicting new geometries that were not experimentally derived.

2.11 Generative Adversarial Networks Theory

The Generative Adversarial Network (GAN) is a class of generative models that can produce a new output, similar to the VAE model, albeit in a much different fashion. As described in the previous sections, VAEs are generative models that encode input data with a regularization component such that the hidden representations are normalized. The decoder function then samples from the latent feature space and constructs a new image. A GAN essentially consists of two neural networks that are both accomplishing different objectives, with the overall goal to be to produce valid new images different from those upon which the GAN was trained [47]. The two components of a GAN are the discriminator, $D(x)$ and the generator, $G(x)$. The generator network works to generate new images while the discriminator works to judge whether or not the images produced by the generator are valid for the label assigned. Thus, the generator models the distribution of classes while the discriminator learns the boundary between those classes [54]. The images generated by the generator are completely artificial, and through the learning process and the discriminator accepting or rejecting those images, the generator can make better outputs that more

closely match the desired label. This implies two simultaneous feedback loops for both portions of the network: one for the discriminator and the known labels for images and one for the generator and the discriminator itself (see Fig. 7).

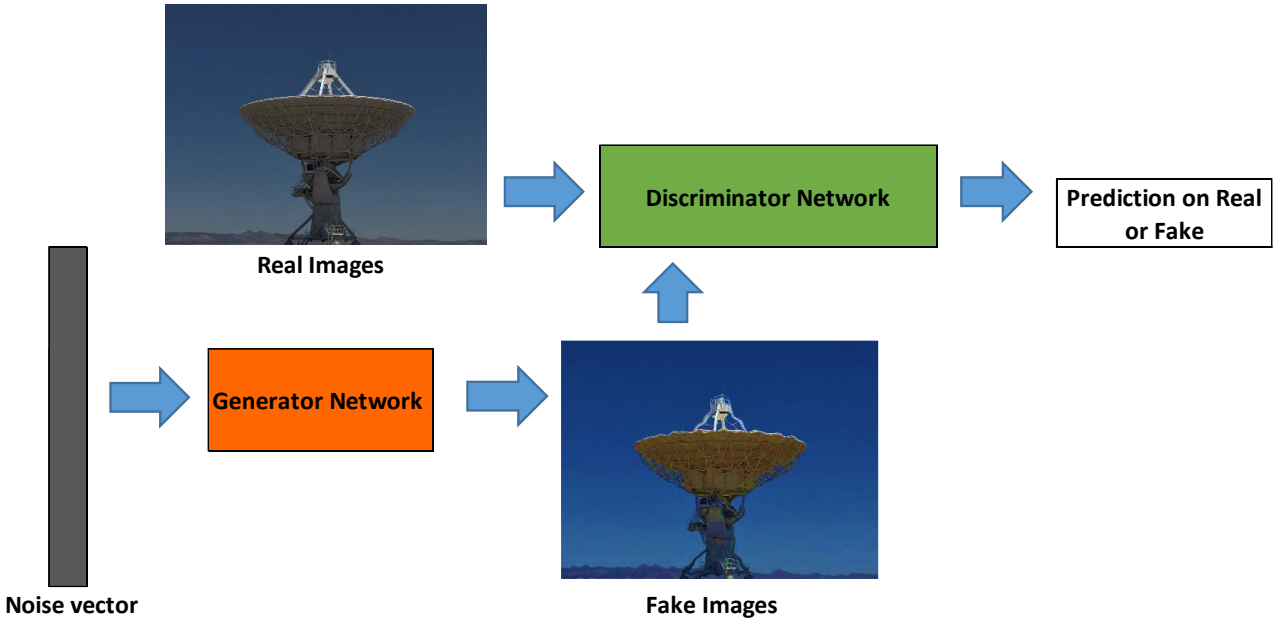

Fig. 7. GAN algorithm example

The generator begins by sampling from random noise *(z)* from a distribution, which it then uses to make images [55]. The generator output is taken as input by the discriminator which has been trained to differentiate real from fake images as a binary classification problem. The discriminator then delivers its own output in terms of the probability that the input is real, such as $D(x) = 1$ if it is real and if it is fake then $D(x) = 0$. The generator tries to minimize while the

discriminator tries to maximize loss, resulting in the following minimax loss function with the value function *V(G,D):*

$$min_G max_D V(D,G) = E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))] \quad (24)$$

where *D(x)* is the discriminator's probability that the generator's output *x* is real, E_x is the expected value over all real data observations, *G(z)* is the generator's output when given noise *z*, *D(G(z))* is the discriminator's estimate of the probability that a fake instance is real, and E_z is the expected value over all random inputs to the generator [55]. The above formula is partially derived from the cross-entropy between two probability distributions; here evaluating the difference between the real and generated distributions. In that function, the generator does not have a direct effect on the *log(D(x))* term, rather, it is trying to minimize the *log(1 – D(G(z))* term. Therefore, when the value of *D(G(z))* is high then *D* will assume that *G(z)* is the same as *x*, which makes *1-D(G(z))* a low value. Alternately, the discriminator tries to maximize the terms *D(X)* and *(1-D(G(z)))*. This will result in an optimal state for *D* as *P(x)* = 0.5 as this is a binary classification operation [27]. Yet, ultimately the generator should be trained such that its outputs taken as input by the discriminator will not be able to differentiate *x* and *z*.

The purpose of the minimax function serves for the discriminator to maximize the objective, V, while the generator minimizes it. As such, both of these functions are learned by an alternating gradient descent. An iteration of the gradient descent on the discriminator will use the real and generated images produced by fixing the generator function. Then the discriminator will be fixed and the generator will be trained to generate an output to deliver to the discriminator in hopes that it will be accepted as appropriately valid [56]. This alternating approach works similarly to the approach described in Section 2.9 for recommender systems, albeit with a much different goal. Here, optimizing the minimax function by iterating between the discriminator, D, and the generator,

G, will be an attempt to achieve better and better quality images from the generator until the discriminator cannot tell the difference between its output and the initial training data. The pseudocode for this operation is as follows [54]:

for number of training iterations do:

for k steps do:

sample minibatch of m noise samples $\{z^{(1)},\ldots\ldots, z^{(m)}\}$ from noise prior $p(z)$

sample minibatch of m examples$\{x^{(1)},\ldots\ldots, x^{(m)}\}$ from data generating distribution $p(x)$

update the discriminator by ascending its gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]$$

end for

sample minibatch of m noise samples $\{z^{(1)},\ldots\ldots, z^{(m)}\}$ from noise prior $p(z)$

update the generator by ascending its gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \left[\log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]$$

end for

While there currently exists no research into developing GANs for material science, or microstructures at all, as it has been noted previously that there is a lack in adapting machine learning in general to these areas – GANs have been used sparingly in other domains with promising results. For instance, in medical science research to generate new images of blood cells – to then have enough data to train classification models to predict cell types [57]. In this work, the underlying problem was a lack of image data, a problem similar to the experimental data discussed throughout this work. Having a computational methodology that can generate new data representations, in this case, new images, was shown to be advantageous towards then having

enough data to properly train an image recognition model. For the LPBF data, a similar approach will be presented in subsequent sections where GANs can be developed to generate new vapor depression generations, similar to the output of the VAEs albeit through a different learning mechanism.

2.12 CGAN Theory

The Conditional GAN is a modification to the standard generative adversarial network where both the generator and discriminator are prepared, or conditioned, during training with some kind of additional information [58]. For a conditioning label *y*, the generator uses the noise vector *z* and the label *y* to create an artificial observation:

$$G(z, y) = x^{*|y} \tag{25}$$

Meanwhile the discriminator will take in as an input the real observations with the labels *x* and *y*, and the artificial observations with the labels that were used to generate those observations, *x*|y* and *y* [59]. The discriminator can then attempt to learn both the real data and the labels, and output a probability value based on its calculation on whether the label-pair is a real, and appropriately mated label-pair. Likewise, its output will attempt to determine whether the observation is either a fake observation or an incorrectly matched label-observation pair [60]. Fig. 8 shows the network modifications for both the generator and discriminator to the general form of the GAN network architecture.

Including this new input parameter helps the CGAN to potentially have two advantages over the general form of the GAN. First, the model should have an improvement in performance as it learns the correct labels for which to generate images. Second, the model has the ability for targeted data generation, that is, producing specific images of interest. This is because the GAN creates images from the latent space which it has mapped, however, interpreting the association

between the points in that latent space and the resultant images is difficult to accomplish. Generating a specific type of image, that is, an image for a given label is therefore difficult in the GAN model. The CGAN however, can overcome this limitation based on its conditional inputs.

The fundamental component of the Conditional GAN is that the added information helps the model to match images to labels during training, and the generator can then use that learned label-image pair to generate new images that correspond to a particular label. This approach can therefore allow for more informed images to be generated, that is, images that are created with a particular end-goal output in mind. For the work described later, that purpose will include the ability to generate new images that correspond to LPBF process parameters – to generate new images based specifically on the inputs to the model for those parameters.

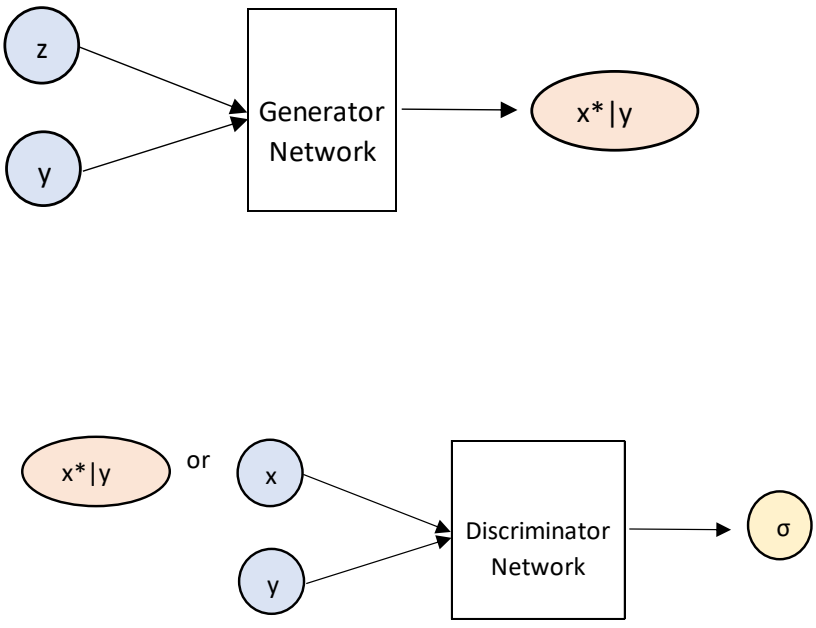

Fig. 8. CGAN modifications to input values

2.13 Additional Applications of Generative Models

The previous section described a potentially advanced modification to the GAN architecture such that images can be created for a specific purpose, to show generate images that correspond to desired labels. Outputs of GAN described in the literature show promising results for being able to generate new images based on larges sets of training data. For instance, human faces, for which random generations are made after the system learned from thousands of samples. Refinements for these purposes can include hyper realistically generated images, which can be accomplished by such models as BigGAN [61], and StyleGAN [62]. Both of these implementations were shown to produce photo-quality images in their generated outputs. In the case of BigGAN, the system replies on the application of a orthogonal regularization in the generator network, which allows the network to alleviate the variance in the generator's output and thereby make better quality images. For StyleGAN, which is based on a Progressive GAN and a neural style transfer design, the output at each layer during training passes through what's known as a style latent vector. Yet while these implementations do deliver high quality results, just like with the general GAN, it is very difficult to control the output of the generators to create something with specific desired attributes.

One method that has been shown to deliver images based on a predetermined characterization is the text-to-image GAN [63]. These methods, typically employing a StackGAN can generate images based on a text input description. Thus, if a text string contained a statement such as "red bird on a tree", the generator would attempt to create an image to match that description. The architectures are described as stacked because they network consists of stacks of images along with texts and image pairs. The underlying mechanism is actually the conditional GANs mechanism, albeit with many more potential labels based on what amounts to a text

dictionary for training. Therefore, these methods show the most similarity to the work that will be described throughout this dissertation, in regards to developing a GAN architecture to take in as input laser processing parameters and output images that should fit those parameters.

While not utilizing a text string as in the StackGAN, the goal is to be able to tell the model to generate an image under the characteristics that govern the different types of vapor depression geometries, based on the laser parameters for the builds that are expressed in the experimental training data. Work in the field is limited in developing and applying generative models for specific classes of images.

Literature is sparse even on real-world examples of text-to-image applications. Historically, many of the implementations of GANs are purposeful at creating new images, but rarely if ever are these images used for scientific purposes, such as model validation. One of the few examples of work such as this resides at NASA, for an undertaking whereby researchers attempted to use a GAN to generate galaxy images [64]. This work was novel in its attempt to create new images, and while successful in their aims, this work again focuses on creating random environments. That is, the researchers did not attempt to use the generative model to create galaxies under specific conditions. And likewise, while the images created give the users new interpretations of how galaxies may look, that work was not used to validate physics-based models. Similarly, in medical imaging synthesis GANs have been employed with encouraging results for generating hypothetical expressions of organs [65]. However, again, this work was not aimed at generating specific organs with specific conditions. Therefore, the development of a novel application of generative models for creating images under specific criteria can be of extreme value to the global machine learning community as the work to be described in later chapters will follow that mission. Furthermore, generative models will be compared to one another, while also

including the outputs of semi-supervised learning approaches in the recommender systems to also produce a generator-like output. It will be shown that the GAN is not the only way to produce a valid and accurate output. In fact, a GAN may not even be the best way to produce a generated output, in some circumstances.

CHAPTER 3

MACHINE LEARNING METHODOLOGY

3.1 Data Collection through Ultra-High Speed Imaging

This chapter discussed the data used for this work, as well as the steps for data preparation. With the exception of Chapters 6 and 8, these methodologies apply to all image-driven work to be discussed. The data and data preparation for Chapters 6 and 8 are discussed separately in those sections, as the work described there had specific aims that required additional data beyond the overall dataset.

The data used for this work was collected from a LPBF experimental build using IN718 that was conducted in Argonne National Laboratory's Advanced Photon Source Synchrotron. This large scale imaging facility captured ultra-high speed (50,000 frames per second) x-ray images of the internal cross sectional area of the LPBF process, at a micrometer scale. This process, known as dynamic x-ray radiography (or DXR) can allow an experimenter to clearly see features of interest in the footage, as the laser (also clearly visible) passes over the metal substrate layer by layer to create the build (see Fig. 4 on page 13). The experimental setup involved the x-ray detection system to be aimed at the powder bed at a 90 degree angle, relative to the direction of the laser movement. For the experiments in which this dataset under evaluation was collected, the laser moved in a single direction. This single track set of experiments developed simple structures, which can serve as a standard before later building complex shapes for production worthy objects. The metal substrate area was approximately 30 millimeters long, 5 millimeters tall, and 500 micrometers wide. The powder layer (which sits on top of the metal substrate and) was between 50 and 100 micrometers tall. After a layer is fused by the passing of the laser, the LPBF system sweeps a new layer of powder over the substrate on the build platform to commence the next

layer's fusion. The laser spot size was approximately 50 micrometers wide, which provided ample space on either side of the laser for the material underneath to melt or vaporize and ultimately fuse in all three dimensions.

A single dataset shows one pass of the laser over the material. The speed of the laser, and therefore the amount of frames where the laser is visible in the field of view, is dependent on the process parameter settings – experimenters have a wide range of laser velocity settings from which they can use. In general, experimental settings for the laser speed range from 0.2 meters per second to 1.4 meters per second. At the scale of the DXR capture ability, the field of view for the laser movement is capable of roughly 2000 images, or frames, separated at 0.01024 seconds apart. As a single build can have thousands of layers in total, these 2000 images that can be captured for each track can easily yield total datasets with hundreds of thousands of images which creates a big data problem. Yet while the volume of data collected on a single build can be large – there is a lack of variety in the data, which presents another problem. Each build, which can take many hours to complete, depending on the size and complexity of the build, must be conducted while in the Advanced Synchrotron facility. The entire LPBF system must be physically moved to the synchrotron's location, and kept there for the duration of the experimental builds. The cost of this procedure, combined with the time required to produce each build, makes it extremely difficult to collect the appropriate in-situ data. Additionally, the time and resources required to procure time at Argonne National Laboratory can take many months or even years due to the high international demand for the facility for purposes across all domains of science.

The work described below constitutes data collected from 35 experiments where the laser power and laser velocity were adjusted (see Fig. 9). Each of these builds used IN718 and were all conducted using a fresh stock of powder material.

Experiment	Laser Power (W)	Laser Velocity (m/s)
1	150	0.2
2	150	0.4
3	150	0.6
4	150	0.8
5	150	1.0
6	150	1.2
7	150	1.4
8	200	0.2
9	200	0.4
10	200	0.6
11	200	0.8
12	200	1.0
13	200	1.2
14	200	1.4
15	250	0.2
16	250	0.4
17	250	0.6
18	250	0.8
19	250	1.0
20	250	1.2
21	250	1.4
22	350	0.2
23	350	0.4
24	350	0.6
25	350	0.8
26	350	1.0
27	350	1.2
28	350	1.4
29	400	0.2
30	400	0.4
31	400	0.6
32	400	0.8
33	400	1.0
34	400	1.2
35	400	1.4

Fig. 9. Process parameter combinations experimentally produced

3.2 Computer Vision to Learn Vapor Depression Geometries

Raw data from the DXR output was initially in the form of a video file, which depicted the in-situ LPBF process in grayscale. The laser is shown moving across the field of view from left to right as it passes over the substrate material in a cross sectional view. From this, the gaseous vapor depression can clearly be seen generated under the laser, and moving left to right across the field of view along with the laser. While this video provides rich imagery of the process, there were some issues with using the raw data for analysis. First, the laser melting process produced small metal particles to be ejected from the surface. Where this ejected material leaves the surface and where it lands could have an effect on measurements taken of the vapor depression as computational techniques might include those objects with the vapor depression depending on how close they appear [66]. Additionally, the vapor depression causes a wave to form on the top surface of the metal, which can distort measurements of the vapor depression's width and depth as the wave fluctuates at a rate faster than the laser moves in the video.

The characteristics of the vapor depression can fluctuate depending on the experimental laser parameter settings. At a slower laser speed the heat intensity from the laser can build up and cause both a larger vapor depression and a bulbous cavity that will solidify and trap the gaseous material, thereby causing keyholing defect. This can also happen if the laser is moving faster, but at a higher intensity where that heat can build up beneath the surface of the material. While it is not yet known what combination of settings will yield an optimal final product (optimal defined as having a predetermined fatigue profile), it is generally accepted that the presence of more defects will lead to a weaker build [16]. Additionally, a long and narrow penetrating vapor depression, even one that does not cause defects, can still have a deleterious effect on the material

microstructure as the solidified structure will have an internal texture that reflects deep and narrow striations.

As described previously, the amount of data for a single build can be large. Therefore, taking measurements of the vapor depression geometries must be automated as human led manipulation and examination of every frame would not be feasible. Pattern recognition techniques can accomplish this, but not without several steps of data cleansing to ensure that only the object/area of interest is targeted by the chosen algorithm. The first step in preprocessing the raw data was to isolate everything in the images except the vapor depression. This would allow measurements to be taken of that vapor depression without allowing any erroneous calculations made where ejected particles, excess surface material, or any other non-relevant region/object being included in the subsequent calculations for that vapor depression. An effective technique for this task was to use foreground extraction and background reduction. In this process, the background of the image is identified as the areas not constantly changing, which reflects movement in the foreground. The images are essentially compared frame to frame to see which pixels are changing the most dynamically, and close together, which therefore implies that those pixels represent an object that is moving. Thus, every area of an image, and therefore the video as a whole, can be categorized into two regions, the foreground and the background. For this analysis, a Gaussian based approach was used, known as Mixture of Gaussians (MOG). In this method in particular, a mixture of *k* Gaussians distributions are used to make a model for each image pixel [67]. The different distributions then represent each of the different image pixel values, which in turn represent the pixel color and intensity. The weight of each one of the distributions used in the models is proportional to the amount of time each pixel stays at the same value for color and

intensity. When the weight values of a pixel distribution are low, that pixel is classified as a foreground pixel.

The results of the background subtraction approach created a defined foreground based on the vapor depression, which was then applied to a thresholding technique to binarize the image into 2 pixel values for black and white (see Fig. 10). Adaptive thresholding was used where the threshold value, the value that determines if a pixel should be converted to black or white, is adjusted throughout the image. This technique is effective over other techniques because of its ability to be generalized better to large datasets, or a variety of similar datasets [67]. For instance, in some images there could be more light and therefore a different contrast, even if the image is capturing the same general image. Adaptive thresholding can overcome such limitations where a single threshold value is applied to all pixels in all images [68]. This method functions by finding the local threshold value in certain areas, or neighborhoods in the image (see Fig. 11). The intensity values at each neighborhood are statistically examined to determine a value for that region. The statistical measures to identify the threshold value of that neighborhood *T*, can be the mean value where *T* can be the mean, the median, or the mean of the range of the values where $T = (max\ value + min\ value) / 2$. A filter (or block) for the neighborhood size of 5, 7, and 9 was tested, with results that did not yield a statistically significant difference in the final results. This was likely due in part to the lack of objects in the image as a result of the background reduction approach previously applied. The vapor depression was highlighted in white in each frame of the video. However, every other pixel was not automatically turned to black as those previously discussed particles in motion were also highlighted in white due to the adaptive threshold approach. Therefore, cropping was applied to all frames in the video which served to both reduce the size of the overall dataset and to ensure that areas above the vapor depression on the surface of the material were not considered in

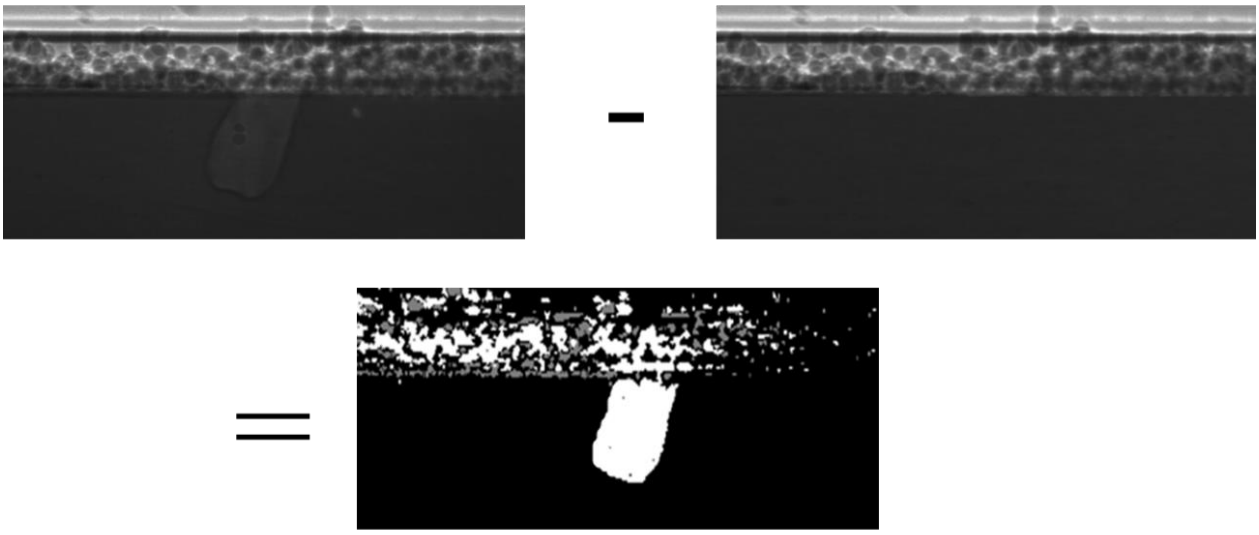

Fig. 10. Background subtraction on DXR image

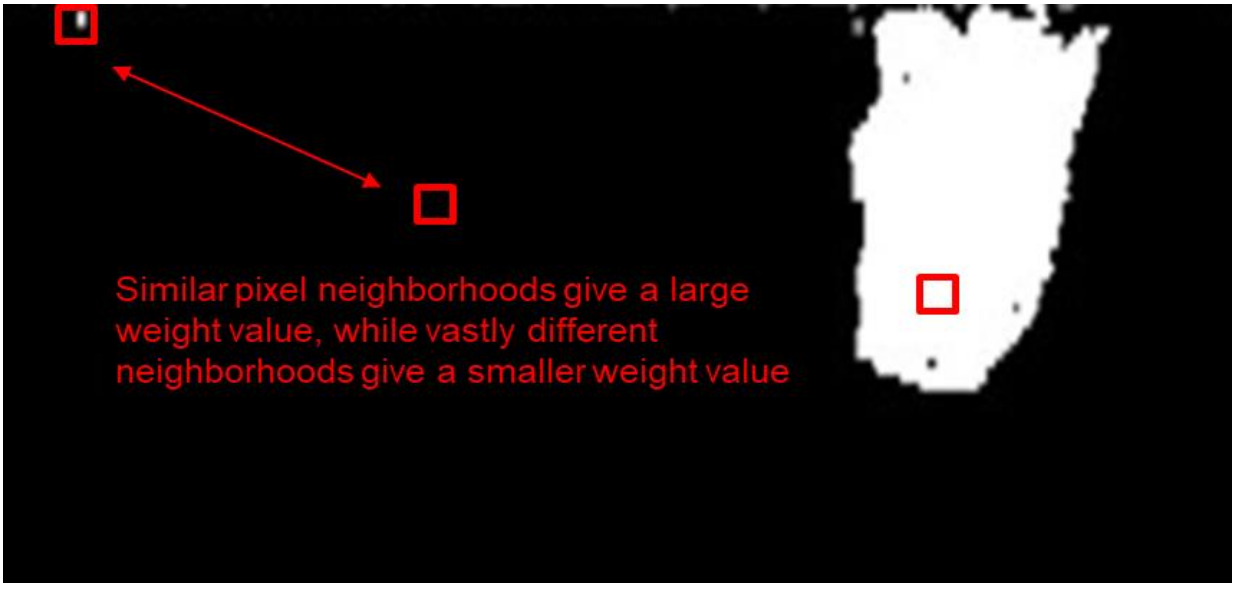

Fig. 11. Non-local means denoising

calculations on the vapor depression geometries. The specific area to be cropped off was the area above the substrate layer, which was easily determined visually as the area above the top pixels in the white highlighted vapor depression region.

The frames in the video were left with pixel noise, or small distributions of pixels of white in the black background or black in the white foreground, which could lead to a higher rate of error in subsequent calculations of pixel values. Therefore, a denoising approach was applied known as non-local means denoising. This is an algorithm commonly used in image processing where the mean value of all pixels in the image is calculated and weighted based on how statistically similar those pixels are to a target pixel. This approach is in contrast to local means denoising where regions or neighborhoods are used, in a fashion similar to the adaptive thresholding technique [69]. While a localized pixel approach was appropriate for thresholding the values, in this dataset where the result was only one object of interest in the vapor depression, a localized approach would be computationally more resource intensive than taking the mean of the entire image. Finally, an image contouring approach was used where the location of each pixel on the boundary of the white foreground and the black background was used, thus providing the perimeter of the vapor depression in each frame. This combined data manipulation approach allowed the analysis to commence on the measurements of the vapor depression for each frame.

3.3 Deep Learning for Microstructure Defect Detection

Multiple deep learning models were developed to test the ability of deep learning networks to learn the latent features of vapor depression and melt pool geometries based on images collected from the DXR technology during the build process. The objectives of this work were multifold:

1. Determine feasibility of the multilayered artificial neural network approach to learning prior to the development of deep learning based generative models

2. Determine whether a deep learning framework is capable of disseminating image based objects based on class of geometry
3. Determine if a deep learning framework is capable of learning to identify images that contain defects, which itself is indicative of the type of geometry

Prior to developing a deep learning architecture using variational autoencoders and generative adversarial networks, it was necessary to ascertain whether or not these techniques are even feasible, if an artificial neural network is able to model from these data. The complexities of the generative models necessitate a baseline using models that while still complex in their architecture, have easier to interpret outputs. Yet these models also have practical applications for this work. The ability to predict the class of a vapor depression geometry leads into the ability to predict both the resultant material microstructure after cooling and has the potential to inform and predict defect generation as certain geometries are more likely to induce defects than others. The four types of geometries or classes for the convolutional neural network (CNN) to learn were: conduction keyholing, penetration with defects, penetration with no defects, and no keyholing. Upon establishing the CNN's ability to learn the different geometries, an additional model was developed to identify defects from the images. This has the potential to inform researchers specifically at the time and location where these defects occur, which is a non-trivial task given the vast amount of data, and images collected for each build. Human identification of defect generation from the tens of thousands of images would not be feasible, which is a direct benefit of using a machine learning approach for data mining.

3.4 Recommender Systems Based on Matrix Completion for Depression Geometries

The missing data problem for characterization of LPBF microstructures can be mitigated through the use of the recommender system with matrix completion. The in-situ experimental data

was obtained from 35 builds that provide vapor depression images, which as discussed in section 3.2, can be used to obtain physical measurements of the vapor depression's size and shape. However, this data is not enough to provide researchers a framework for knowing optimal build settings to induce a vapor depression at the appropriate depth, width, and area to sufficiently penetrate the substrate material yet prevent the buildup of keyholing defects during the build. For instance, data was collected for a build with the laser set at 350 watts and 0.2 meters per second, which depicted an unstable vapor depression that constantly collapses and forms defects. Yet at the same intensity but a slightly faster velocity of 0.4 meters per second, the vapor depression is stable with no defects, but the overall depth of that vapor depression is shallower in comparison. This shows there somewhere between 0.2 and 0.4 meters per second there is an optimal speed that is deep, but not too deep where the vapor depression loses its stability.

The data from all 35 experimental builds can be analyzed frame by frame to produce a dataset of its geometric properties at each 0.01024 second time interval during the build (matching the frame capture rate from the DXR sensor). At 514 frames per experimental build (where the vapor depression is in the field of view), this dataset would have 17,990 rows, or observations of vapor depressions. From this data, a matrix can be constructed where instead of users and items represented in the rows and columns, the laser power and laser velocity could be represented. And in place of user ratings for an item to populate the values in the matrix, the measured value for the depth of the vapor depression could be used. As data was collected at velocity settings of 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, and 1.4 meters per second, this matrix would have 7 columns. From this, new empty columns could be created between each measure interval, which would notionally represent 0.3, 0.5, 0.7, 0.9, 1.1, and 1.3 meters per second. In this new matrix of 13 columns, there would be enough missing data that this could be characterized as a sparse matrix. Upon performing the

matrix completion methodology discussed previously, these values could potentially be approximated, which would represent vapor depression measurements that were not experimentally collected. The completed matrix could then be used by the recommender system to make recommendations, or predictions, on what combination of power and velocity should be used to achieve a vapor depression at a specified depth.

Using the recommender system approach, data can be approximated to mathematically recreate vapor depression geometries that were not experimentally derived. Additionally, it could provide a computational approach to determine optimal process parameters that could lead to industry certification of laser power bed fusion additive manufacturing components. Since new experiments at the Advanced Photon Synchrotron are not able to be performed, having a computational method to achieve new vapor depression measurements entirely using machine learning approaches would be of great benefit to this subfield of materials science research, which can be applied to any different material suitable to LPBF AM. While this methodology would provide a wealth of numerical data for LPBF process characterization, the following sections will describe how deep learning can also be used to derive new data, albeit image data to supplement the raw DXR images experimentally collected from the limited experimental runs captured at the Argonne National Laboratory.

3.5 Variational Autoencoders to Generate New Vapor Depression Images

Sampling from the latent space using VAE should generate new images based on the features encoded and decoded by the model. Unlike the standard form of the autoencoder which returns the input image, the VAE utilizes the encoder to produce a distribution over the entire latent space rather than a single point by incorporating the reconstruction loss and the regularizing nature of the KL loss [53]. This allows sampling in that space that can generate new images. For instance,

to generate a new image based on a region in the latent space in between two different samples, with the expectation that the new image will be an expression of some mixed-hybrid image of those two different samples – the procedure will be to find the difference between the mean vector from those two samples, add half of the difference to the original, then decode the result.

The generative ability of the VAE will be used to generate new images of vapor depressions based on experimental parameters that were not experimentally conducted. Data was captured for builds with a laser velocity setting at 0.2, 0.4, 0.6, 0.8, 0.9, 1.0, 1.2, and 1.4 meters per second. The velocity settings in those gaps, such as for 0.3, 0.5, 0.7, 0.9, 1.1, and 1.3 therefore yield unknown vapor depression geometries. As discussed in previous sections, there is a narrow range of velocity values for which a major change can occur in the vapor depression geometry. We have seen that at 0.2 meters per second there are vapor depression shapes that induce defects forming, while at 0.4 meters per second there are none (at 350 watts). Therefore between 0.2 and 0.4 there should be a velocity setting value that will yield an optimal vapor depression that is deep yet stable and not defect inducing. Sampling from images in the latent space between 0.2 and 0.4 will help provide information into what vapor depression geometries in that gap would look like. The VAE will find the encoded vapor depressions at 0.2 meters per second and 0.4 meters per second, obtain their encoded vectors, compute their difference, and decode it as described in the previous section to computationally produce an image of a vapor depression at 0.3 meters per second.

By developing a VAE model that encodes images at 0.2, 0.4, 0.6, 0.8, 0.9, 1.0, 1.2, and 1.4 meters per second, and sampling halfway between them, this work will achieve images of the vapor depression that will represent experimental runs at 0.3, 0.5, 0.7, 0.9, 1.1, and 1.3 meters per second. This will be repeated for each laser intensity setting, such as 150, 200, 250, 350, and 400 watts. The result will be a dataset of new images for 30 different combinations of laser parameters that were

not, and will not be experimentally produced. From these images, materials science researchers can have almost double the information produced experimentally to visually identify and quantifiably measure and verify, the appropriate laser settings to produce vapor depressions at the desired depth, width, area, and convex hull area for optimal build quality.

3.6 GAN to Generate New Keyhole Defect Representative Images

The fundamental idea of a GAN is to have what are essentially two CNNs working in conjunction. One, known as the generator, generates new data. Meanwhile, another, known as the discriminator, evaluates the output of the generator to determine if the image created appropriately captured the features necessary to be classified in the appropriate category [57]. This technique is effective due to the combined nature of these algorithms. As described previously, the CNN is a discriminative algorithm; if trained with images of a certain class, it can learn the features of those images and predict future images into the appropriate class. This makes the use of a GAN appropriate to explore as a CNN was previously evaluated for its ability to effectively learn the features of vapor depressions. Given the effectiveness of the CNN at appropriately learning the features of the vapor depression geometries, the discriminator will be just as affective at learning those features and determining if an output from a generator is real or false. Likewise the generator should be able to iteratively craft a better output to achieve an accurate representation of the training data.

The following work involved developing a GAN that was trained on vapor depression images which were generated under known process parameters that led to the generation of defects – to then produce additional images of defect generating depressions. It has been established that a deep and narrow vapor depression can sufficiently penetrate the substrate material deep enough to fuse the material adequately. But when this happens, sometimes the vapor depression geometry

leaves behind portions of the material where gas is trapped and the defect forms in the microstructure. Materials can be tested post-processing using a variety of stress-based techniques to determine if a certain amount of porosity is acceptable, given the tradeoff between vapor depressions that are deep enough to melt the material and the likelihood that that deepness will cause defects to generate.

From the in-situ data for the 35 experimental builds, only 8 of those builds had data that depicted vapor depressions that left keyhole defects. While the size and location of those defects vary somewhat, there is not enough data to make conclusive determinations on the quantity of pores that are acceptable in a build. Therefore, the use of a GAN will generate additional representations of vapor depressions with defects. For each experiment that yielded defects, the GAN will be trained on those images and then the generation of new data of novel microstructures will then be used to create a porosity profile for each of those 8 experimental configurations. These will result in 8 sets of new microstructure images. The vapor depression in these images will then be measured in terms of depth, width, area, and convex hull area to examine if there is a significant difference in the sizes and distribution of those defects, in a manner supplementing the work discussed in previous sections.

The use of the GAN will contribute to ongoing research into process parameter characterization by producing more images that could not be experimentally derived. Similarly, the lack of data (only 8 experiments that depict defects) represents a missing data problem as there is not enough data to adequately evaluate the dispersion of defects in these builds. As demonstrated in the previous section describing the theory behind the GAN algorithm, this approach is a valid technique for generating new images. Therefore, it is an appropriate technique to apply to this lack

of image data problem, thereby providing a new methodology for computational materials science research in porosity profiling.

Modifications are possible to the general form of the GAN network architecture such that performance and accuracy can be improved upon [70]. By incorporating a continuous feature representation into the training, the model can potentially learn the distributional relationships of that feature with regard to the underlying principals governing that representation. For instance, the previously described thermodynamics and physics that inform the fluid nature of the gaseous vapor depression and the liquid melt pool formations. While the laser intensity for each build is relatively constant, the heat at the surface of the build is changing over time, due to buildup in energy as the laser moves across an area. The heat intensity is recorded in the thermal sensors in the LPBF experimental setup, and therefore provides a means of engineering a new feature to describe heat per unit area. Generating by incorporating this continuous representation would also have a direct impact on LPBF applications as complex builds will be a constant focus; to create objects that can be utilized in aerospace projects. As described previously, the heat buildup in the corners and crevices of these builds as the process develops from layer to layer can have a deleterious effect on the microstructural stability. Thus, involving a feature representation for the thermal variations in the build can improve the GAN's ability to accurately model and generate those new microstructure representations.

This body of work represents an advancement in the field of computer science by developing a first of its kind evaluation of multiple machine learning frameworks for image characterization, including by methods typically not employed for image-based learning tasks, such as the recommender system approach. These recommender system algorithms have historically been researched for incorporation into commercial systems but historically have not

been evaluated for image prediction tasks, nor for prescribing potential image characteristics, thereby demonstrating some generative capabilities. Their development here improves upon the body of knowledge around how recommender systems can be adapted for image processing; computationally deriving object boundaries and making predictions on future object boundaries in hypothetical images for uncollected experimental data.

Further this work seeks to improve upon the state-of-the-art in generative models by developing a methodology for incorporating a continuous feature representation into the training such that the model can potentially learn the distributional relationships of that feature with regard to the underlying principals governing that representation. While work exists in this regard in conditional GAN, the process that will be developed here will use thermophysics based equations to characterize and predict fluid properties that are encoded in the deep learning framework as latent features. Additionally, the work described in this dissertation demonstrates an attempt to train a generative model in and end-to-end fashion which can be generalized to many other image representation, generation, and restoration problems in the machine learning subfield of computer vision.

CHAPTER 4

DEEP LEARNING FOR GEOMETRY CLASSIFICATION

4.1 CNN for Vapor Depression Characterization

While advancements have been made in studying the vapor depression of LPBF builds using advanced imagery, there has been no research into the application of deep learning to these datasets. This could aid researchers in two ways: 1) to reduce the manual labor required to examine all images in a large dataset where thousands or potentially millions of images could be taken, and 2) allow researchers to have a comprehensive catalogue of expected vapor depression geometries based on the 35 combinations of build parameters previously used. A convolutional neural network (CNN) was developed and utilized in order to learn the features of the DXR images and to make predictions from the thousands of input images from the 35 different experiments to determine the general shape of the vapor depression. The data was manually curated whereby images that depict the following vapor depression shapes were labeled: conduction keyholing (2560 images), penetration keyholing with defects (2750), penetration keyholing with no defects (3480), and no keyholing (3060). Fig. 12 provides examples for each class. The inclusion of a class for penetration with defects was to allow for the model to determine the likelihood for keyholing defects to be left behind. The data was then divided into a training set at 80% of the data and a testing set at 20% of the data.

The objective was to see if this deep learning approach can accurately classify the type of keyholing based on learning from input images of each class. This model could be used to accurately label future image datasets without manual inspection or curation, to determine what type of keyholing would be present in a future build with different build parameters. While it would be infeasible to collect DXR data in-situ for every future build, having the results of this

model could serve as a ground truth for establishing future experimental build parameters. Additionally, if the model is able to accurately identify a defect in an image, this can be interpolated to all layers in a build to then give a porosity profile for that build whereby the aggregated defects are quantified.

4.2 Geometry Classification Results

An outline of the CNN model architecture is shown in Fig. 13. The rectified linear unit (ReLU) function was used as the activation function, as this is the most commonly used activation function in deep learning with numerous research studies showing this activation function achieves faster training and better performance over other approaches [71]. This function also avoids saturation, where an activation function can squeeze the input – meaning they have upper and lower bounds that compress the neural response into a bounded set of values. For instance, the TanH activation function compresses the values to a range of -1 and 1. ReLU avoids saturation by providing a non-linear function with no limit. Applied to each pixel in the input image, the function returns a value of 0 if a negative value is provided as input and returns the actual value if a non-negative value is used as an input. As such, this expression can be mathematically described as:

$$f(x) = \max(0, x) \tag{26}$$

This gives it the qualities of a linear function for values greater than 0, yet acts as a non-linear function for negative values at the same time. While somewhat simple in its approach, it is effective at handling non-linearity in the underlying data.

The CNN was able to correctly label images from the testing set 92% of the time for the 4 classes of vapor depressions, with a true positive rate of 93%, false negative rate at 7%, true

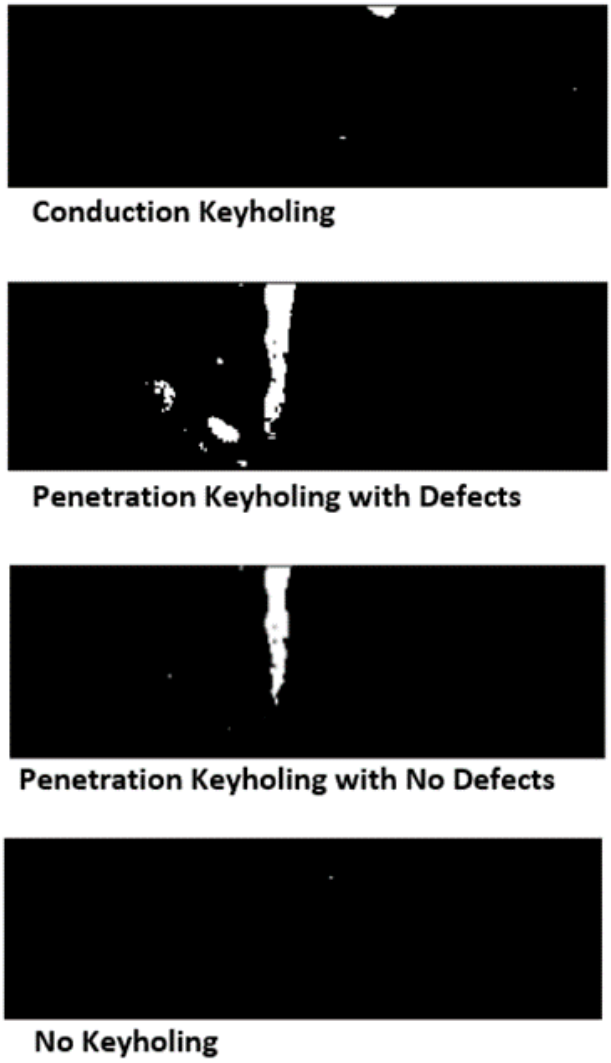

Fig. 12. Keyholing classes

TABLE 1
CNN Performance Summary

	Accuracy
Overall	97.21
True Positive	97.83
False Negative	2.24
True Negative	93.41
False Positive	6.64

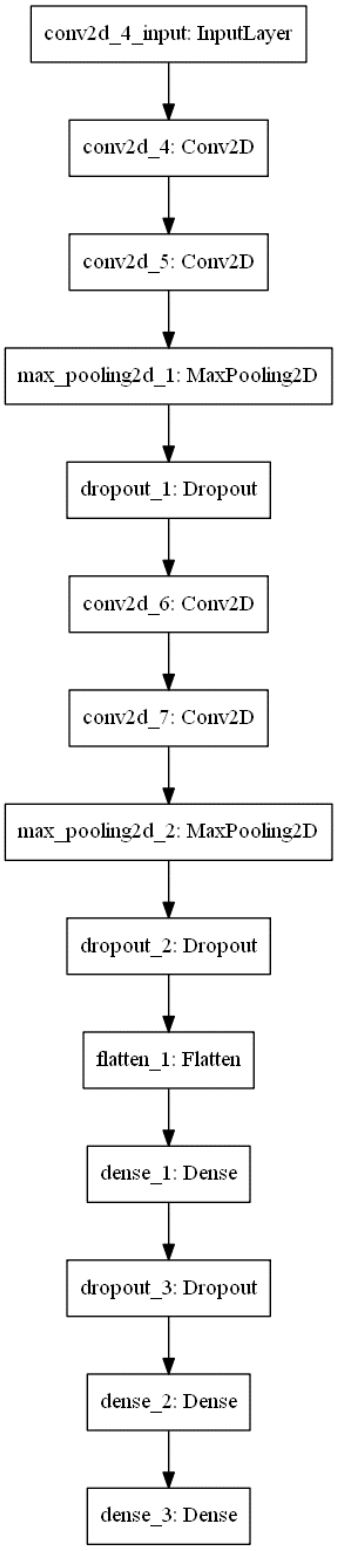

Fig. 13. CNN model architecture

negative rate at 90%, and false positive rate at 10% (see Table 1). This implied that there was enough information learned from the geometries of the vapor depression and size of the vapor depression to accurately determine how the vapor depression would act given the experimental settings of the laser (the process parameters) as well as determine whether or not that combination of settings will induce keyholing defects in the substrate material. The model was then used for a training set containing only three classes of vapor depression geometries; removing the penetration with defects class. This resulted in an overall accuracy of 97% for the predictions with a true positive rate at 98%, false negative rate at 2%, true negative rate at 93%, and false negative rate at 7%. While this smaller set of labeled classes does not predict the presence of defects in a build, it does imply that the number of input images in the training set was sufficient for the model to correctly learn the features in the images of each class. A summary of model performance can be seen in Fig. 14. This model could be advantageous to use for determining if the laser settings were sufficient to penetrate the substrate with enough depth to create a deep enough melt pool to fuse the metal powder particles. Ultimately, the results from both models showed that deep learning is successful in identifying the features in an experimental build from the LPBF process and characterize its in-situ process in a manner that has never been applied previously.

4.3 CNN for Defect Detection

In this portion of the work, the CNN approach was used to evaluate images for defect detection. Here the previously discussed model was used in addition to pre-trained models, thereby using transfer learning on images of builds made by LPBF. Four models based on NASNetMobile [72] and DenseNet121 [73], and the custom designed CNN were trained and evaluated, and further comparison was conducted. The objective was to demonstrate that CNN models can be feasible

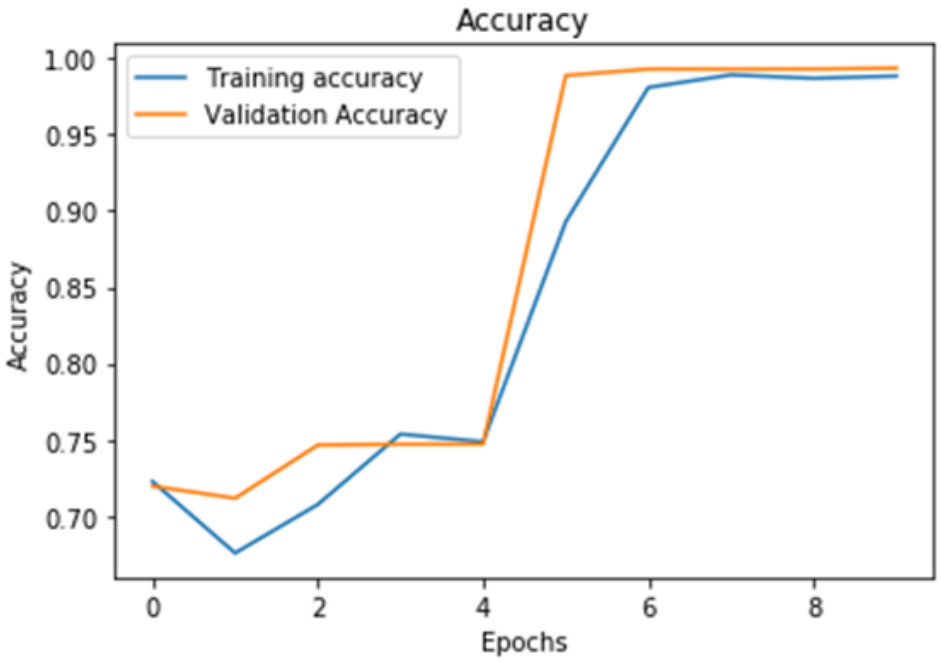

Fig. 14. CNN model performance over training cycles

tools for LPBF process monitoring and to provide an automated and rapid classification of images, which display challenging defects in shape and size.

NASNet architectures are based on the neural architecture search (NAS) framework [74]. NAS is performed to find the best architecture based on a smaller dataset (CIFAR-10) and then transfer the learned architecture to a larger dataset (ImageNet). In the NASNet search space, the convolutional networks are comprised of convolutional layers or cells, which possess identical structure but different weights. Once the best cell structure is found using the CIFAR-10 dataset, several copies of that cell are stacked to build the convolutional architecture that can be applied to the larger ImageNet dataset. Searching for a cell is computationally more affordable than searching

for a complete network architecture and the best cell found with one dataset can generalize to other datasets.

A dense convolutional network (DenseNet) is comprised of dense blocks connected by transition layers. Each dense block has several dense layers and the feature map of each layer is concatenated to the feature maps of the preceding layers [73]. A dense layer performs batch normalization rectified linear unit (ReLU) activation, convolution, and concatenation [75]. A transition layer performs batch normalization, ReLU activation, convolution, and average pooling [76]. The predictions layer, which is a fully connected layer, is connected by global average pooling. The equation below shows the H_L function whose output x_L in the L^{th} layer is the concatenation of the outputs (concatenated feature maps from preceding layers: $[x_0, x_1, \ldots, x_{L\text{-}1}]$) of the previous dense layers 0, 1, …, L-1 [77]:

$$x_L = H_L([x_0, x_1, \ldots, x_L]) \tag{27}$$

4.4 Transfer learning

Transfer learning is a technique that allows transferability of knowledge to a modified neural network. It relies on weights learned from other datasets to train the deep neural network with a different dataset. A mathematical definition for transfer learning was described in [78] and is also formulated here with the same notation and definitions for consistency. A domain $\mathcal{D}$ consists of $\mathcal{X}$, which is a feature space of all data instances, and $P(X)$, which is a marginal distribution of the data used during the learning (training) process, such that X is a sample ($X \subset \mathcal{X}$), where $x_i \in X$ is the i^{th} data instance (feature input), $\therefore \mathcal{D} = \{\mathcal{X}, P(X)\}$. A task $\mathcal{T}$ consists of a label (or class) space $\mathcal{Y}$, where $y_i \in \mathcal{Y}$ is the i^{th} output or label, and an objective predictive function $f(.)$ that learns during the training process and can be expressed as $f(x) = P(\mathcal{Y}|X)$, such that $\{x_i, y_i\}$ is the training data, $\therefore \mathcal{T} = \{\mathcal{Y}, f(.)\}$. During testing, $f(.)$ can be used to predict the label of a new data

instance in $\mathcal{X}$. Now, let us define a source domain $\mathcal{D}_S$, where $(x_{S_i}, y_{S_i}) \in \mathcal{D}_S$, such that $x_{S_i} \in \mathcal{X}_S \wedge y_{S_i} \in \mathcal{Y}_S$, and a target domain $\mathcal{D}_T$, where $(x_{T_i}, y_{T_i}) \in \mathcal{D}_T$, such that $x_{T_i} \in \mathcal{X}_T \wedge y_{T_i} \in \mathcal{Y}_T$. The source task is $\mathcal{T}_S$ and the target task is $\mathcal{T}_T$. Therefore, transfer learning can be defined as [78]: $(\mathcal{D}_S, \mathcal{T}_S)$ enables knowledge transferability to $(\mathcal{D}_T, \mathcal{T}_T)$ by improving the learning process of the target prediction function $f_T(.)$, where $\mathcal{D}_S \neq \mathcal{D}_T \vee \mathcal{T}_S \neq \mathcal{T}_T$.

Transfer learning accelerates the training of neural networks and allows adaptability to other classification cases. In the TensorFlow Keras API [79], different deep neural network models have been implemented that can be initialized with weights learned using the ImageNet dataset. For example, by replacing the top layer of a base model (e.g., DenseNet) that has been trained with ImageNet is possible to add a new classifier layer for the new output labels and only train the added layer with the new input data. Further fine-tuning can be achieved by enabling the model entirely or partially for training.

TensorFlow was used to build the models based on NASNetMobile and DenseNet121 using transfer learning [80]. The weights on ImageNet were loaded on the base models. Transfer learning was applied by replacing the top layer of NASNetMobile with a new classification layer that implements the softmax function with two outputs for binary classification.

For the models based on NASNetMobile and DenseNet121, batch renormalization was enabled in the added classification layer to overcome some shortcomings of batch normalization with mini-batches [81]. In this work, a batch size of 64 was used for the NASNETMobile and DenseNet121 based models and a batch size of 16 for the custom CNN model. The training dataset had 4352 images, the validation dataset had 640 images, and the test dataset had 640 images. All images were divided equally into two categories (defect and non-defect).

4.5 Defect Detection Evaluation

The two models based on NASNetMobile were trained and validated after replacing the top layer in the base model. The optimizer used was Adam with a learning rate of 0.0001 and AMSGrad enabled. For model training, the batch normalization layers in the base models were set to work in inference mode. Two blocks and a classification layer were added to the base model to implement NASNetMobile-A. Each block included batch normalization (with renormalization), dropout, and a fully connected layer (ReLU activation). The top layer was a two-output fully connected layer with softmax activation. The new layers were enabled for training whereas the layers of the base models were frozen. For NASNetMobile-B, the top layer of the base model was replaced with a two-output top layer with softmax activation. The entire network was set to trainable but still maintaining their batch normalization layers in inference mode. After testing, the accuracy achieved with NASNetMobile-A was 0.9359 and NASNetMobile-B was 0.9719. The confusion matrix for NASNetMobile-B is shown in Fig. 15.

Two models based on DenseNet121 were next implemented. The top layer of the base model was replaced with new layers and a new classification layer. The Adam optimizer was used with a learning rate of 0.0001 and AMSGrad enabled. This is a variant to the Adam optimizer that uses the maximum of past squared gradients to update the parameters, rather than using the exponential moving average, thereby achieving convergence, where the Adam optimizer alone may often fail to achieve. During fine-tuning, the conv5 dense block, which is the last dense block in the DenseNet121 model, was enabled alongside the added layers, and the model was retrained. In the case of DenseNet121-A, two blocks were added and each block included batch normalization (with renormalization), dropout, and a fully connected layer (ReLU activation). The top layer was a two-output fully connected layer, with softmax activation, for binary classification.

For the first time training, the added layers were trained whereas the base model was frozen, and its batch normalization layers were set to inference mode. After fine-tuning and testing, DenseNet121-A achieved an accuracy of 0.9875. For DenseNet121-B, the top layer of the base model was replaced with the following layers: a fully connected layer (ReLU activation), dropout, and a two-output fully connected layer with softmax activation. Similar to DenseNet121-A, initially the base model was frozen and the added layers were trained. After fine-tuning and testing, DenseNet121-B achieved an accuracy of 0.9641. The best performance after testing between the two DenseNet based models was of DenseNet121-A, whose confusion matrix is shown in Fig. 16. As described previously, the custom CNN model was trained in its entirety using the Adam optimizer, with a learning rate of 0.001 and AMSGrad enabled. After testing, the accuracy achieved was 0.9766. Fig. 17 shows the confusion matrix for the custom CNN model.

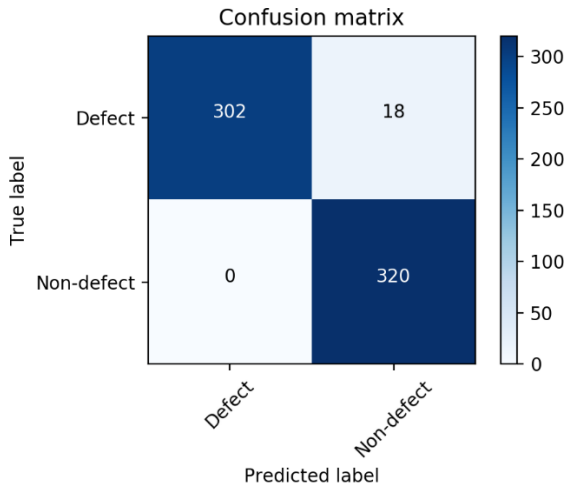

Fig. 15. Confusion matrix for NASNetMobile-B after testing

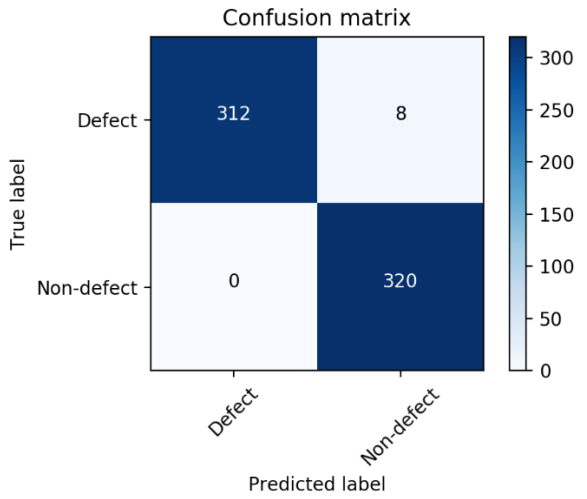

Fig. 16. Confusion matrix for DenseNet121-A after testing

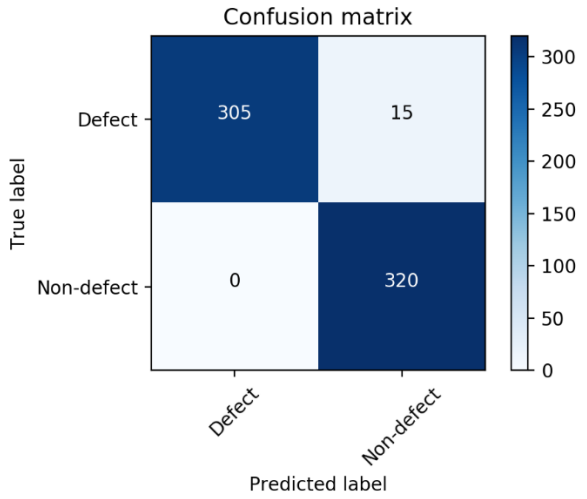

Fig. 17. Confusion matrix for the custom CNN model after testing

Fig. 18 shows the receiver operating characteristic (ROC) curves for all the models implemented in this section. The black dashed line represents the random assignment case. Owing to the characteristics of the ROC curves and the fact that the base rates of true positives and true negatives are equal, the area under the ROC curve and the accuracy are numerically equal. DenseNet121-A exhibited the best results.

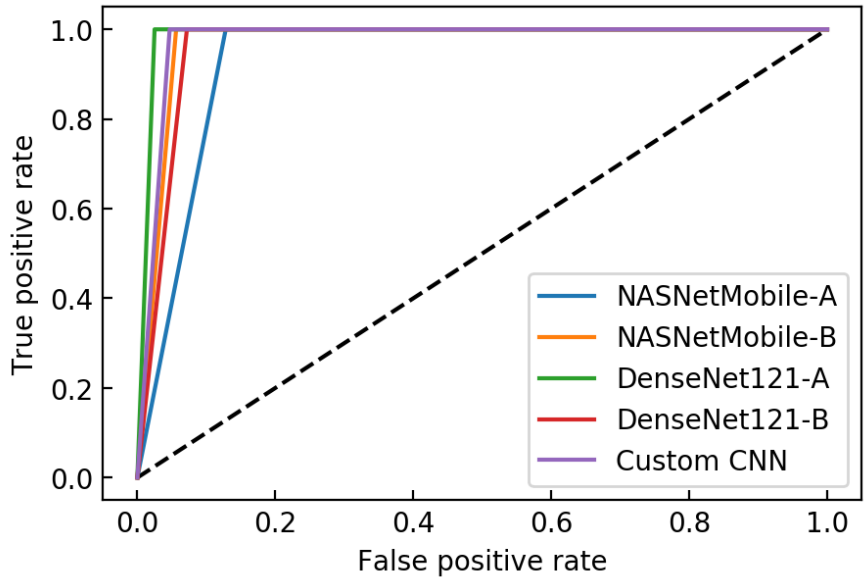

Fig. 18. Graph of the ROC curves for the DL models implemented and tested

Table 2 summarizes the performances of the implemented models showing the accuracy, precision, recall, and F-1 score. With the employed test dataset, all of the deep learning models were able to identify correctly the non-defect label, i.e., no false positives. Therefore, the precision was equal to 1.0 for each of these.

TABLE 2
CNN Performance Summary

MODEL	ACCURACY	PRECISION	RECALL	F-1 SCORE
NASNetMobile-A	0.9359	1.000	0.8719	0.9316
NASNetMobile-B	0.9719	1.000	0.9438	0.9711
DenseNet121-A	0.9875	1.000	0.9750	0.9873
DenseNet121-B	0.9641	1.000	0.9281	0.9627
Custom CNN	0.9766	1.000	0.9531	0.9760

4.6 CNN Summary

This chapter provided an overview of several CNN models to evaluate if deep learning methodologies are appropriate for the data at hand. As discussed, the CNN is capable of correctly classifying images from the DXR data to determine what type of vapor depression geometry is captured in each image. Further, the CNN is capable of detecting the presence of defects in the images, thereby establishing that the models are capable of learning not only the features that constitute the vapor depression objects, but other objects in the images as well. This indicates that these deep learning methods are a valid approach for further investigation of LPBF images from a machine learning standpoint. As has been discussed in the literature review, many generative models are complex adaptations to the CNN model utilized in this chapter, in the algorithmic processes that converge on a solution. These results therefore justified further investigation into

more complex deep learning architectures, not just for data classification, but for data generation as well.

Following the results of data generation, presented in later chapters, revisiting this work with transfer learning could find additional benefits. It was established throughout the literature of the CNN that model performance tends to improve with more training data. Therefore, having more images of the in-situ process could allow investigators to perform even more finely tuned transfer learning or deep learning methodologies to detect the presence of defects. While such a system cannot be deployed in real-time for future builds due to the complexity of the DXR imaging process, it is possible to match those builds that create the most defects and suggest which combinations of parameters are most likely to induce those defects. The work presented here in this chapter can therefore serve as a precursor to a more thorough investigation, in conjunction with the results to be discussed in later chapters on generating new data.

CHAPTER 5

COMPUTATIONAL GEOMETRIC ANALYSIS

5.1 Convex Hull Geometries

The shape of the gaseous vapor depression gives insight into how it will solidify, thereby impacting the microstructure of the material. For defects to occur in the material, the vapor depression should have a rounded shape such that the top portion can close off and thereby result in an enclosed void after the laser has passed that region and the material cools. This can easily be visualized by inspection of the vapor depression movement through the material; there are instances where the bottommost area of that depression forms a bulbous shape, which after several frames seems to break off and is left behind (see Fig. 2.4). This activity strongly suggests there is a link between the shapes of the vapor depression at any given time, and the likelihood that a defect will occur in the next few hundredths of a second.

Understanding the geometry of the vapor depression is integral into understanding the characteristics of the final material. But just analyzing the size may not be conclusive enough to generate expectations on those properties. For instance, the vapor depression could be wide and shallow, known as conduction keyholing, or deep but narrow, known as penetration keyholing. These are essentially opposite representations yet have the same area. Additionally, while penetration keyholing is more likely to have an effect on the material properties as it leaves behind deep striations in the material, as previously noted the exact shape that penetration takes is relevant.

While drawing conclusions from the area of the vapor depression could give insight, there can be other methodologies that might be more impactful on the analysis, such as examining the convex hull area of the vapor depression at any given time. The convex hull of an object is defined as the set of points that enclose all of the points in a given set of points [82]. The name convex

implies that this shape will have no regions bent inwards, as would be expected if only the outermost points in a set were used to outline its shape. For the vapor depression geometries, this would yield a shape that encompassed a region larger than the vapor depression itself. The area of the region enclosed by the convex hull should be larger than the geometric area of the vapor depression – if that vapor depression had a geometry where a portion of it was larger and more bulbous in nature than the rest of the depression (see Fig. 19). Since the vapor depression only has one area (near the bottom) that when shaped into a round bulb will break off to form a defect - a larger convex hull area would be indicative of this effect.

For the vapor depression analyses, the convex hull was calculated for every frame in the dataset. This technique was applied to the dataset previously manipulated, to yield a set of convex hull measurements for the vapor depression in every frame. The number of outermost points that were used to determine the convex hull was also derived. Additionally, the area of the convex hull in each frame was calculated to compare against the geometric area. The feasibility of this approach was also analyzed, by looking at how the convex hull algorithm converges to a solution, for each frame – and how taxing that computation would be for each frame individually. This is additionally relevant as these shapes are complex polygons, where a single pixel can potentially form an edge and therefore impact the computational complexity of this approach.

5.2 Convex Hull Computational Approach

The general problem to be solved by the convex hull calculation can be summarized by, if given a set of points on a two-dimensional plane, how can the points be connected such that they would not form a concave angle? In the case of this analysis, the points necessary for consideration were the boundary points, all of the points that constituted the perimeter of the vapor depression.

Fig. 19. Original image, geometric area in green, and convex hull area in blue

While there are many algorithms that can be used to solve this problem, with varying results and computation time, one such approach that has been firmly established as an efficient technique is known as the Jarvis March, or otherwise known as the "gift wrapping algorithm" [83]. It has this name as it is similar to how gift wrapping paper is wrapped around an object, in this case a set of predetermined points. Its performance has been documented as being favorable for datasets where the number of points is small, or the number of points that constitute the convex hull is expected to be small in comparison to the total number of points [83]. The algorithm begins by selecting a starting point *p*, which will be the minimum value on the x-coordinate, which is done in $p - O(n)$ where *p* is the initial point. The total number of points that are selected as pivot points, or *h*, are then determined against all available points, *n*. The point *p* then becomes a pivot point from which in a counterclockwise direction the next point is located by checking the orientation of the other points from point *p*. The point that has the largest angle is the point that is the most counterclockwise from *p*, which is done in $O(n)$. After comparing against all points, this point becomes the new point *p*, therefore again is $p - O(n)$. This computation is completed in $O(nh)$ time complexity. The pseudocode for this algorithm can be demonstrated as [83]:

```
convex hull algorithm (S)
    # S = set of points
    # P = set of points which form the convex hull.
    # I = final size of the set of points
    xmost = minimum x coordinate point
    i := 0
    repeat
        P[i] := xmost
        endpoint := S[0]      // initial endpoint for next pivot point possibility
        for j from 0 to |S| do:
            // endpoint == xmost is rare case and is only when j == 1 no endpoint
                has not yet been set for the loop
            if (endpoint == xmost) or (S[j] is on left of line from P[i] to
                endpoint) then
                    endpoint := S[j]   // if there is a larger left turn, change
                    endpoint
        i := i + 1
        xmost = endpoint
    until endpoint = P[0]      // continued until returning to first point
```

The algorithm essentially runs with two loops comparing the chosen point to all of the other points in the set, *S*. One loop checks every point in *S* while another repeats this for each point on the convex hull. This is why the complexity time is *O(nh)* as it is dependent on both the number of *n* and *h*. This is also why it is generally faster than other approaches to convex hull computations

that can be done in *O(n log n)* such as Graham's algorithm, when the number of vertices, or *h*, is smaller than the log of *n* [84].

5.3 Convex Hull Results and Analysis

The application of the convex hull algorithm to a single frame in the dataset took 0.000998 seconds on a system with a 7th generation Pentium i5 CPU processor at 2.60 GHz. For all frames in a single dataset, 2,000 images, this took 1.996 seconds.

On average, there were 13 points that were found as vertices for the convex hull. While the average number of hull vertices was 13, the average number of points in total was 522. Therefore, the *h* value in these calculations were significantly smaller than the *n* value, thereby supporting the Jarvis algorithm approach. From these 13 points, the area of the resultant polygon's convex hull was calculated, which took 0.000997 seconds for a single frame. A summary of the processing time is depicted in Table 3. The second column of the table shows the calculation time applied to every frame in a single experimental run. The final column is an approximation based on a hypothetical build with 1,000 layers/datasets.

TABLE 3
Calculations for Convex Hull

Calculation	**Single Image**	**2000 Images 1 Dataset**	**1000 Datasets 1 Complete Build**
Convex Hull Points	0.000998 sec	1.996 sec	33 min 16 sec
Convex Hull Area	0.000997 sec	1.994 sec	33 min 14 sec
Total for Points and Area	0.001995 sec	3.990 sec	1 hr 6 min 30 sec
Geometric Area	0.000999 sec	1.998 sec	33 min 18 sec

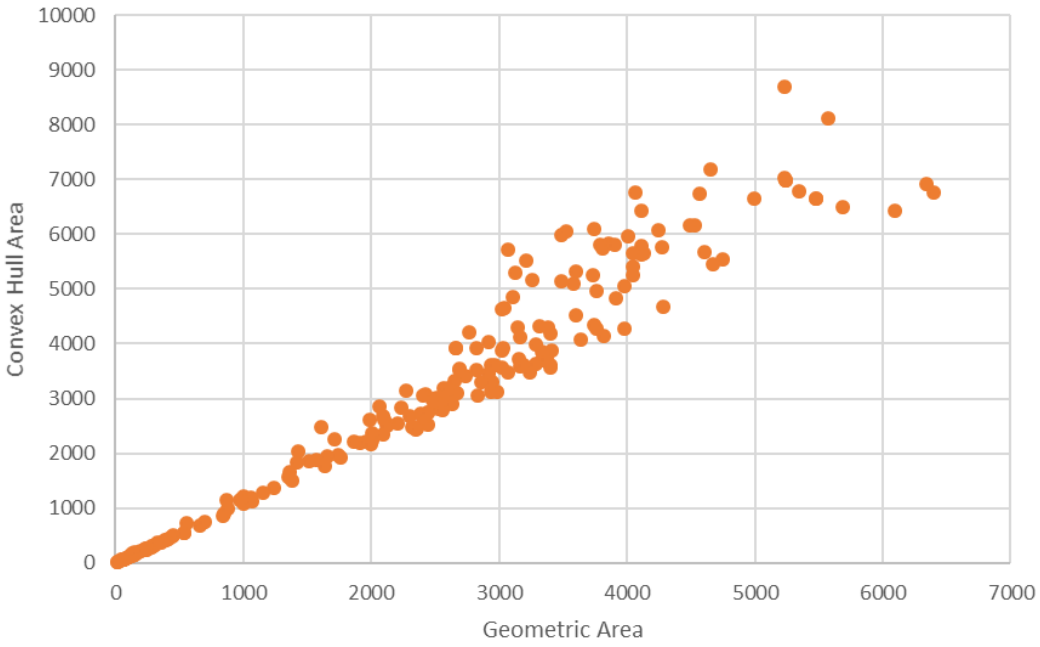

Fig. 20. Convex hull areas versus geometric areas

As shown in the table, the combined approach of finding the convex hull vertices and then calculating the area of that region takes approximately twice as long as calculating the area of the vapor depression. The calculation for the geometric area is relatively simple in comparison as it is simply the total pixels in the vapor depression, multiplied by the conversion of pixel size to micrometers, which was 1 pixel = 1.932 micrometers. Overall, the convex hull areas were larger than the geometric areas, and in many cases significantly more (see Fig. 20). From these results, the average area of the convex hull was 3,099 μm^2. Meanwhile the average geometric area was 2,429 μm^2, a statistically significant difference. The average percent difference among all frames was 19.24%, and the median percent difference among all frames was 16.11%.

Those frames in which the percent difference between the geometric area and the convex hull area was greater than 20% were subsequently extracted from the original dataset for inspection. Observing each of these frames in the dataset confirms that the vapor depression had a large bulbous region near the bottom. Furthermore, for all of those frames with a percent difference value greater than 20%, the analysis showed that within 2 to 4 frames later the percent difference fell to a range of 0.05 to 0.10 percent difference in all but 5 of the frames under inspection. This further indicates that when the vapor depression develops that large bulbous geometry, that area will break off and form a defect. The convex hull calculations indicate when that is likely to occur, as its area will be significantly higher than the area of the vapor depression at that point in time. This data also shows that defect formation happens quickly, as these frames are taken 0.01024 seconds apart, which therefore suggests that the vapor depression can create a defect in 0.01024 to 0.04096 seconds.

5.4 Voronoi Diagram with Clustering

After examining all frames in the dataset with regards to the size of the vapor depression, the next analysis focused on finding a method to classify different vapor depressions from the dataset into groups such that those that are most likely to lead to defects can be categorized. A combined approach was taken where an unsupervised machine learning approach known as K-Means clustering was applied to all measurements of the convex hull area with regards to time through the experimental run. Subsequently, the cluster regions were used to create a Voronoi Diagram based on the centroid points of those clusters in a Euclidean space. This combined approach helped to develop a new data label for each vapor depression in the dataset where the largest could be quantifiably grouped separately. The goal of this approach was to help identify

the combination of settings that yielded the largest measurements, which could be grouped together as a category.

Unsupervised machine learning is an approach where inferences are made from patterns within the data without referencing an outcome [85]. These algorithms collectively can provide insight into large datasets where those patterns may previously be unknown, thus providing a valuable tool to use for exploratory data analysis, or EDA [86]. One technique in particular that can aid in EDA is clustering, where observations from the dataset can be grouped, or clustered, based on some criteria [87]. This grouping can then be used as an additional feature for more targeted analytical approaches [88]. The K-means clustering algorithm is one method that can be useful for grouping all observations using continuous variables from the dataset [89]. K-means iterates between two steps. Initially in the data assignment step, there is a centroid point established which defines the center of each cluster. Every data point is then categorized into one of the clusters based on the Euclidean distance to its nearest centroid point [90]. This can be demonstrated by:

$$\underset{c_i \, \epsilon \, C}{\operatorname{argmin}} \, dist(c_i, x)^2 \tag{28}$$

where:

dist is the Euclidean distance, or L_2 distance

c_i is the collection of centroids in set C

x is each data point

In the second step, each centroid is recomputed by taking the mean of all data points assigned to that centroid's cluster:

$$c_i = \frac{1}{|S_i|} \sum x_i \, \epsilon \; S_i{}^{x_i} \tag{29}$$

where:

S_i is the set of data point assignments for each i^{th} cluster centroid

The Voronoi approach is to partition a plane, where polygons are generated by those partitions such that each polygon contains one generating point and every other point within its boundaries are closer to that generating point than any other [82]. In this case the plane was the Euclidean plane with the results of the clustering analysis plotted, into such regions where the center of the cluster was the generating point for the Voronoi regions. In the Euclidean plane, the distance between two points p and q are determined by:

$$dist(p,q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \tag{30}$$

where:

p_x is the x coordinate of the first point

p_y is the y coordinate of the first point

q_x is the x coordinate of the second point

q_x is the x coordinate of the second point

The Voronoi diagram will then be computed such that the subdivision of the plane P will be into n number of regions where a point q is found in the same region as p_i if and only if $dist(q,p_i) < dist(q,p_j)$ for each $p_j \in P$ with $j \neq 1$.

5.5 Voronoi Computational Approach

The calculation of Voronoi regions can be efficiently produced using Fortune's algorithm, which is a sweep line algorithm, that "sweeps" a surface in a two dimensional Euclidian plane [91]. The Fortune approach combines two lines, a sweep line that is a straight line moving left to right across the plane, and a beach line which is curved like a series of parabolas that move to the left of the sweep line across the plane. While the points that the sweep line has passed points to its left which have been incorporated into the Voronoi diagram, the beach line divides that area based on the information currently known into its subsequent regions. The beach line function can occur

regardless of the information not known, that is, the points to the right of the sweep line that have not yet been passed. This is possible because the parabolic nature of a line can have the properties of having a set of points equidistant from any point left of the sweep line to the sweep line itself. The beach line is therefore the boundary between those resultant parabolas that define that equidistant space. Therefore, as the sweep line progresses through the plane, the beach line will form vertices at points where those lines and the sweep line parabolas cross, resulting in the edges of the Voronoi diagram [91]. As the algorithm progresses, it uses a binary search tree to record the properties of the beach line in memory. This allows fast lookup and insertion or deletion of new items into the memory structure, as constant information is being received through the line sweeping [92]. As new points are integrated, the insertion and deletion of parabolic lines to the left of the sweep line can be ordered such that the x-coordinate of their location is the reference point from which they can be identified. The algorithm then continues adding or removing parabolic structures until the model converges once the entire plane has been swept.

The application of Fortune's algorithm computes in $O(n \log n)$ time as there are $O(n)$ events to consider as features of the Voronoi diagram, and it takes $O(\log n)$ time to process each of these events. Each of these events requires the binary search tree functions of recalling, removing, or adding information. Therefore, this results in the total run time of the algorithm as $O(n \log n)$ time [91]. A proof of this follows [93]:

Theorem: For $n \geq 3$ sites, the Voronoi diagram contains at most $2n-5$ vertices and $3n-6$ edges.

Proof:

1. For points that lay on the line, this is always true. For points that do not lay on the line, let V = number of vertices in Voronoi diagram, E = number of edges, and N = number of inner faces which is also equal to the number of points

2. From Euler's formula: V – E + N = 2. Due to the possibility that the Voronoi diagram contains infinite edges, let's create a new vertex to represent infinity and connect all of the edges to it such that is becomes a planar graph: (V + 1) – E + N = 2.
3. In the Voronoi diagram we know that every vertex has a degree at least 3. An edge is between two vertices, so that equates to $3 * [(V + 1) \leq 2 * E]$. Algebraically, this transforms into less than or equal to 2n – 5 vertices and less than or equal to 3n – 6 edges.

5.6 Voronoi Diagram with Clustering Results and Analysis

The application of the K-means algorithm for the convex hull measurements for every frame in the dataset took 0.1560 seconds on the same system with a 7th generation Pentium i5 CPU processor at 2.60 GHz. To determine the optimal number of clusters, *k*, to model the data, Bayesian Inference Criterion (BIC) was used, as it has shown to be an effective technique by using the information criteria values [94]. This resulted in 6 clusters being chosen as the optimal number of clusters for the observations in this dataset. The centroid of the 6 clusters was then determined, which was then used to construct a Voronoi diagram of these points, of which the values were then normalized, and overlayed on which was a plot of the convex hull area on the y axis with the time on the x axis. Based on the 6 points to be evaluated, the generation of the Voronoi diagram was completed in 0.0603 seconds. Thus, the total combined time for this evaluation on the dataset took 0.2163 seconds to complete.

As shown in the resultant plot (see Fig. 21), the observations from the dataset, which was each recorded vapor depression, can be grouped into 6 classes based on the convex hull geometries. As this value has been previously shown to be illustrative in determining whether or not a vapor depression will generate a defect in the material, having this evaluation measure to group the recorded geometries can be an additional tool to understand how and when these defects may

occur. The Voronoi diagram serves as both a visual inspection tool, one that can be quickly generated for each experimental build, which can then be read to determine what percentage of the build was likely to have defects based on how many frames (with a known time separation between) had displayed a geometry likely to form such a defect. Additionally, cluster and region assignment for each measured geometry can provide an additional data label that could be used for future purposes, such as training a predictive model to learn and predict what cluster (and therefore likelihood) a vapor depression geometry may have for generating a defect in the material.

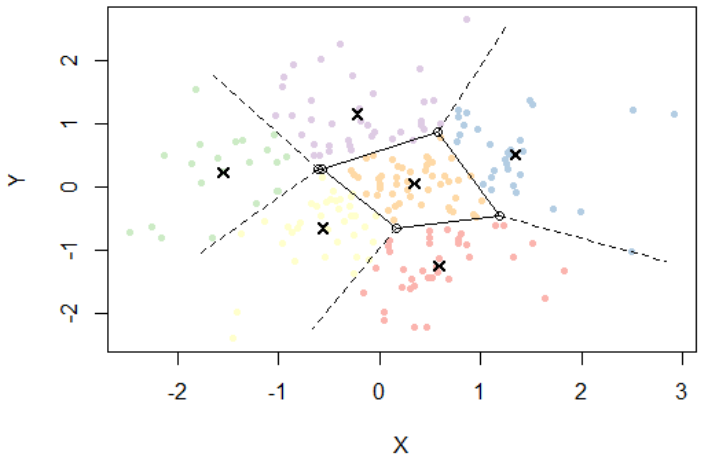

Fig. 21. Convex hull areas clustered against time

CHAPTER 6

TIME-RESOLVED GEOMETRIC QUANTIFICATION

6.1 Thermal Effects on Morphology

As a laser melts the surface of a material, it is possible that too much laser energy can be absorbed and the surface temperature becomes high enough to vaporize the material. This often occurs in LPBF because of tightly focused laser beams used, which leads to a complex interplay of multiple physical phenomena, such as recoil pressure and Marangoni convection [95]. All of these phenomena can cause a vapor depression, or keyhole, that depresses the surface of the liquid metal. The resulting morphologies of these vapor cavities vary widely with process conditions, i.e., different combinations of laser scan power, speed, and beam diameter at the metal surface [96]. More specifically, high power, low speed, and a small spot size promote deep and narrow keyholes.

During keyhole-mode laser melting, the occurrence of a deep and narrow keyhole and its turbulent fluctuations can cause the formation of porosity, which constitutes as a subsurface defect and degrades the mechanical properties, such as the fatigue life, in additively manufactured components [96]. Keyhole fluctuation can also generate spatter which can fall back down on the surface of the powder bed and contribute to defect formation [97]. These realities motivate the necessity of a sophisticated time-resolved quantification technique for capturing keyhole geometry under different process conditions to investigate the variation in keyhole morphologies and determine correlations between the quantified keyhole geometric features and the process parameters.

With the development of real-time synchrotron observation, i.e., high-speed X-ray imaging, which reveals the density contrast between solid material and a vapor cavity, it can be

possible to track the dynamic evolution of the keyhole geometry with high spatio-temporal resolution [97]. To measure geometric features, such as depth, width, and front wall angle, of a keyhole from high-speed X-ray images, computer vision techniques were employed. The application of computer vision techniques in materials science has been rapidly expanding because of their advantages for quantifying digital images. The quantification of microstructural images through computer vision enables numerical feature extraction so that the resulting data can be used for analysis and characterization of microstructures [98]. Similarly, it is also possible to leverage computer vision to extract quantitative information from the high-speed X-ray visualizations for analysis and characterization of keyhole dynamics.

This chapter presents a pipeline that carries out an image processing routine followed by geometric feature extraction to obtain time-resolved geometric information of a moving keyhole. Statistical methods are applied, such as Spearman's rank-order correlations followed by agglomerative hierarchical clustering to correlate keyhole geometric features with the process parameters. The emphasis of this chapter is on the methodology behind the proposed image processing pipeline that performs geometric feature extraction from pixelated data. Thus, the main point of the data analysis is to demonstrate the practicality of semi-automatic extraction of keyhole data as well as how analysis of the resulting data may lead to new insights.

6.2 Advanced Image Processing for Feature Extraction

Before extracting the geometric features from a series of x-ray images, it is necessary to obtain a well defined digitized boundary around each keyhole. To this end, it was necessary to develop a novel image processing pipeline which is documented in this section. An overview of the image processing path is depicted as a flow chart in Fig. 22.

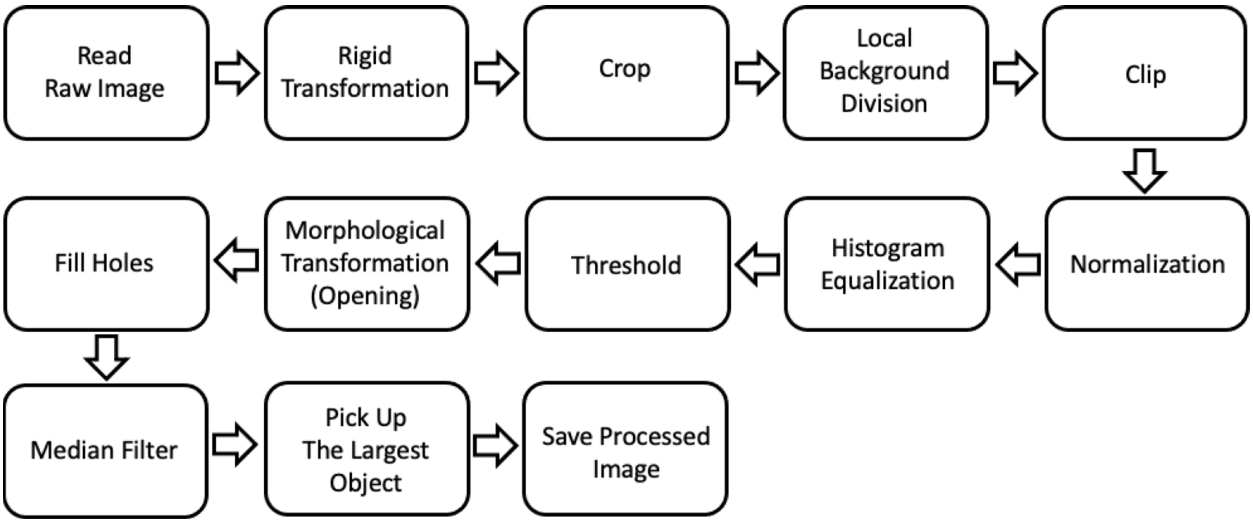

Fig. 22. A flow chart illustrating the image processing pipeline

The operation begins by importing a series of raw x-ray images as 8 bit gray-scaled images. A single representative image from an x-ray experiment performed on titanium alloy Ti64 with power at 426 W, velocity 1.2 m/s, and beam diameter 65 μm is shown in Fig. 23a and will be used in subsequent sections. In some of the high-speed x-ray imaging experiments, the top surface of the sample can appear slightly tilted if the top and bottom of the sample are not completely parallel. Ensuring the sample is parallel to the frame and removing the pixels above the top surface improves the preservation and segmentation of the top of the keyhole. Because this can impact the subsequent image processing steps and skew the geometric data of the keyhole, a rigid (Euclidean) transformation was employed to mitigate this issue. The red line in Fig. 23a runs parallel to the initial sample surface at the beginning of the experiment and serves as the reference line for cropping and transformation. The rigid transformed image in Fig. 23b was rotated and translated about this red line while preserving the rigidity of the sample [99], ensuring that the top surface of

the sample was parallel with the frame of the image. After performing the transformation, a calculation was performed followed by storing the transformation angle of each experiment to correct the front wall angle that is measured in the proceeding feature extraction. In addition to the transformation, cropping was used to remove any unnecessary area (i.e., any part of the image above the sample surface). Whenever the aforementioned "overlap band" hindered the described image processing steps, it was cropped at a level up to 5 pixels beyond the top of the sample surface to remove this artifact. This is especially important for resolving the top portion of the keyhole; thus, it became an important aspect of the cropping that the depth of the keyhole measured in the subsequent geometric feature extraction process was correctly calculated. Fig. 23b displays the result of both the rigid transformation and cropping process applied to the raw image.

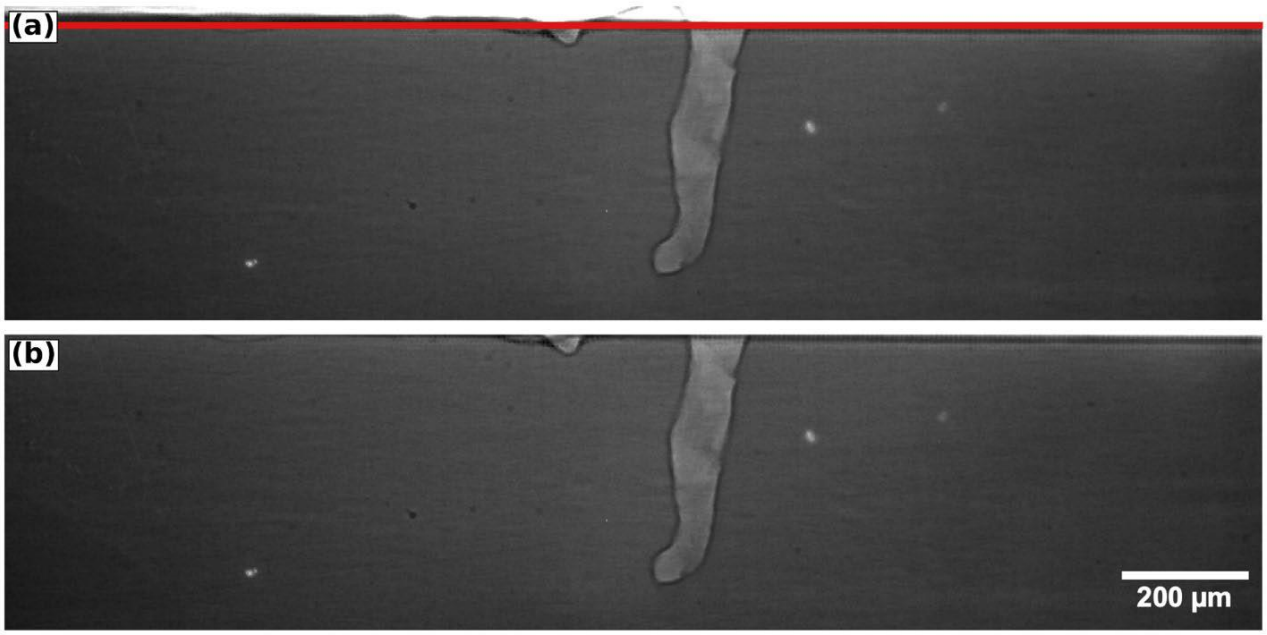

Fig. 23. a) Gray-scaled raw image of LPBF with red line indicating reference level,
b) The same image after transformation and cropping

Following the cropping and transformation, the images underwent a sequence of de-noising and normalization steps to enhance the quality and contrast of the keyhole. To quantify each enhancement, both a histogram and a cumulative distribution function (CDF) were generated of the pixel values for each processing step as shown in Fig. 24. In the raw x-ray image, as shown in Fig. 24a and b, it can be observed that the pixel values were clustered, which made it difficult to segment the keyhole from the image. The first step to achieving a well-defined keyhole was to remove the experimental noise which is apparent in Fig. 24a. Typically, a simple background division step was used for this purpose where each image in the series was divided by the first image; however, when applied to the sequence of x-ray images, this rudimentary method proved inadequate. This was because the background divided images possessed low noise at the beginning of the series, but as the laser travelled across the sample, the experimental noise increased in intensity, perhaps because of thermal expansion of the sample and interactions between the sample and the enclosing holder.

To alleviate the increase in noise with each image, a local background division process was enacted. This algorithm divided a given image, *i*, by an image that appeared earlier in the series by some certain distance, $i - x$, with x being a hyperparameter to be calibrated. This effectively kept the experimental noise at a consistent level across the entire series of images, which greatly benefits the subsequent processing. Any cropped pixel location beyond the surface of the sample was stored so that the following step used integers. Fig. 24c and d shows the resulting image of the local background division and its pixel intensity histogram. Here, there are two keyholes in each image: the leading keyhole from the original image and trailing keyhole from the image used for the division step. This new methodology for instrument denoising in time-resolved image sequences necessitated that a second keyhole appeared which came from the frame that was x

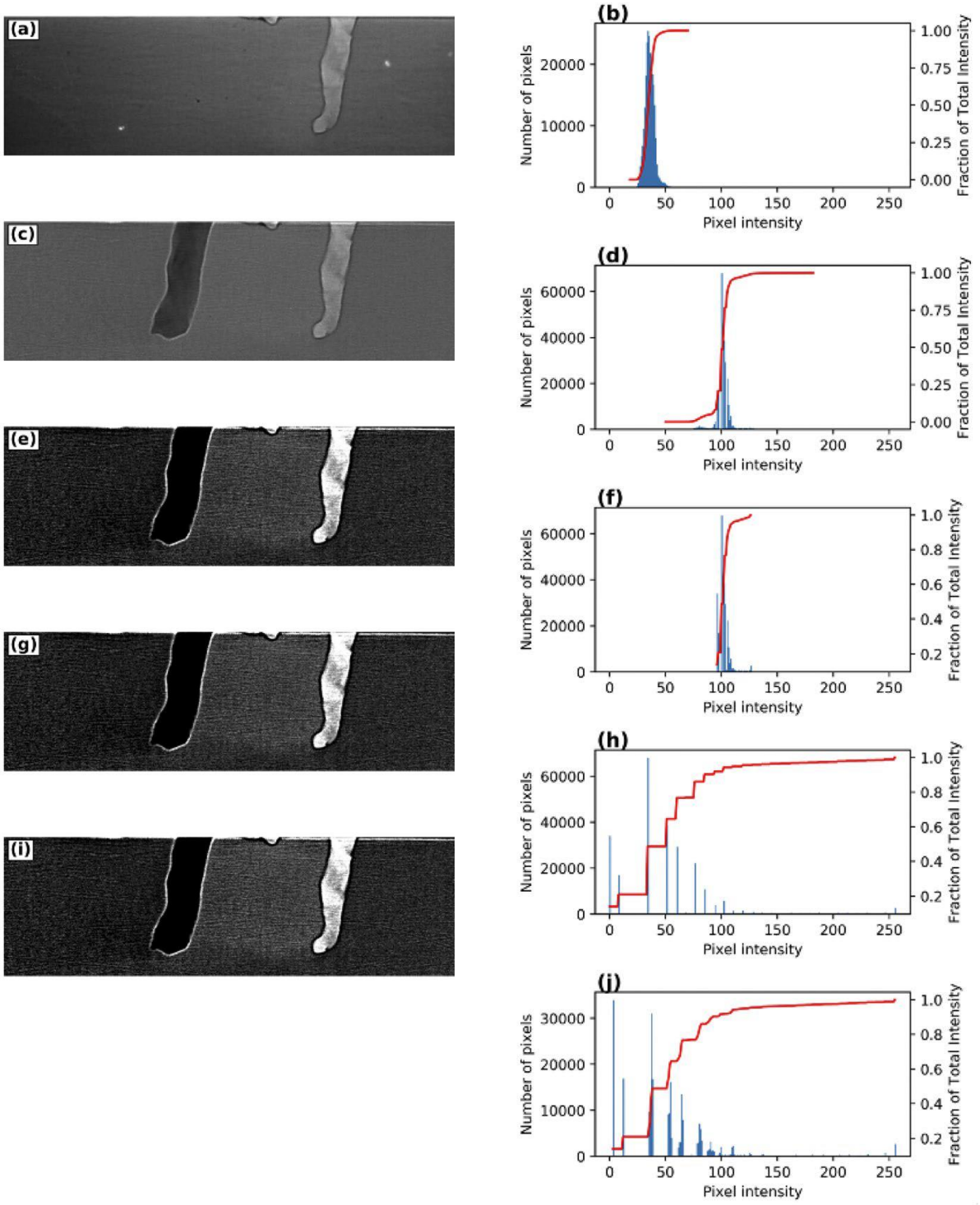

Fig. 24. (a) The final image from Figure 23b after rigid transformation and cropping; (b) with the resultant pixel intensity histogram and CDF from (a); (c) after local background division; (d) with the resultant pixel intensity histogram and CDF from (c); (e) after clipping; (f) with the resultant pixel intensity histogram and CDF from (e); (g) after normalization; (h) with the resultant pixel intensity histogram and CDF from (g); (i) after adaptive histogram equalization; (j) with the resultant pixel intensity histogram and CDF from (i)

behind the frame of interest, *n*. For the remainder of the analysis, only the leading keyhole (which is brighter than the trailing keyhole) was considered for segmentation. This is because, in some cases, pores generated behind the leading keyhole can collapse into the trailing keyhole region thereby hindering the subsequent image processing steps and the geometric feature extraction. Note that setting an optimal value for *x* during this step was crucial to avoid overlap of the two keyholes, which entangles the shapes of both keyholes and interferes with segmentation. After systematically evaluating many values for *x*, typically, setting $x = 50$ provided the best balance between preventing any interference between the keyholes as well as minimizing the image noise.

Although the image in Fig. 24c depicts a well defined keyhole, the local background division still has outlier pixel intensities, which could explain why the CDF has a much larger pixel range in Fig. 24d than in 24b. To resolve this issue, it was necessary to introduce a clipping step to the processing routine. Here, the goal was to remove any outliers and also enhance the contrast by clipping the levels of intensities to a chosen range. The minimum and maximum values were each set to a chosen percentile of the histogram of pixel values which increased the robustness of this routine for each image in the series. This interval edge is another hyperparameter that can be fine-tuned for different experiments, but it was usually set to the 10*th* and 99*th* percentile, respectively. Values greater or smaller than the set interval were clipped to the edges of the interval through an array clipping process [100]. This step eliminated the remainder of the outlying pixels that were unrepresentative of the information contained in the image and resulted in a smaller CDF pixel range as observed when comparing Fig. 24d and 24f.

After going through the previous image processing steps, the pixel values were mostly confined to a range that varied among each frame within the experiment. To address this inconsistency, the pixel values in each frame were normalized by re-scaling with the maximum

and minimum values of the respective frame. This step increased the distance in intensities between the keyhole and its surroundings since the re-scaling stretched the range of values to the end values, i.e., 0 and 255 [101]. The resulting image and its pixel intensity histogram of this step are shown in Fig. 24g and h, where it is apparent that the CDF spans the whole range of the x-axis. Next, we performed adaptive histogram equalization [102] to further enhance the local contrast around the keyhole, as shown in Fig. 24i and 24j. This function equalizes an image locally by dividing the image into small tiles which avoids excessive amplification of noise and enhances edge definition. Although the localized tile-wise nature of this step only made subtle changes to the histogram according to Fig. 24j, performing this routine tended to increase the repeatability of determining a suitable threshold value that outlined the keyhole for the entire image sequence.

After the adaptive histogram equalization, a simple global binary thresholding method was employed on the image to assign any pixel with a value less than the threshold value to zero, and all other pixels to 255. As shown in Fig. 25a, there was too much noise in the image after the initial thresholding step. Thus, a sequence of de-noising steps was performed to remove noisy pixels in the binarized image. To begin, this used an opening morphological operation, which consists of pixel erosion followed by dilation as shown in Fig. 25b. By using a predefined kernel, a single anchor point, size, and shape were dictated for the whole stack of images. The erosion operator then computed the minimum pixel value overlapped by the kernel when it scanned across the image and replaced the image pixel under the anchor point with that minimum value; while the dilation operator did the opposite [101]. After this, any remaining open holes in the keyhole were removed by using a hole filling command from OpenCV, which produced Fig. 25c. For further de-noising, a median blur operation was used. In this operation, the predefined square, which can

be any size, was scanned over the image with each pixel being replaced by the median value in the square neighborhood surrounding the center pixel.

As seen in Fig. 25d, the majority of the small speckles of noise from Fig. 25c were cleaned with this last de-noising step. The final stage in the methodology was developed automatically detect nonzero object in the image, which was the keyhole itself, as shown by the outcome in Fig. 25e. This was achieved by using an algorithm based on a function from the OpenCV library that detects all nonzero objects in the image, sorts them by number of pixels, and selects the largest object. After all of the aforementioned processing steps were complete, the finalized images were ready for geometric feature extraction.

6.3 Geometric Feature Extraction

After successful image segmentation, geometric feature extraction was performed on each image from the array of x-ray images. The key utility of this process was making use of the contour function from OpenCV to identify the object, i.e., the keyhole, in the segmented image. Once a feature was identified, it was characterized by using quantitative metrics based on its geometry. The primary geometric features that were extracted consist of the width, depth, and front wall angle, which are regarded as the key attributes for understanding keyholes [103]. Additional geometric features, such as area and perimeter, were also considered for this analysis. Fig. 26 presents the extracted geometric features of each frame from the same series of images as the experiment shown in the previous figures.

The keyhole depth at each frame was calculated by taking the difference between the maximum and minimum pixel positions perpendicular to the top surface of the sample. The depth

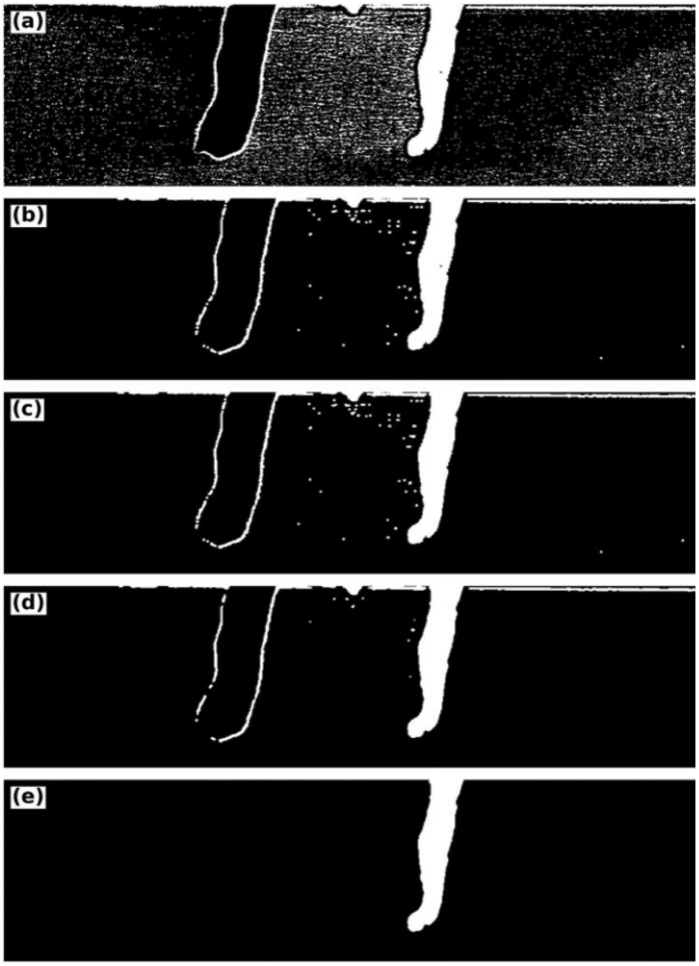

Fig. 25. The final image from Figure 6.3i (a) after global binary thresholding; (b) after morphological opening transformation; (c) after filling holes in the keyhole; (d) after median blur; (e) after picking up the largest object

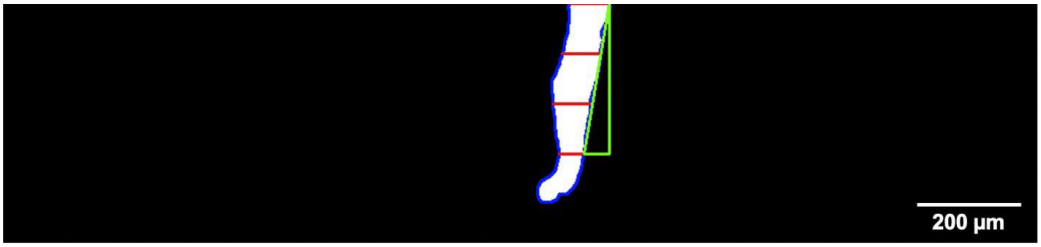

Fig. 26. The final image from Figure 6.4e after the geometric feature extraction process with the target features of the keyhole shown with colored lines

also took into account the additional distance applied to the cropping process to remove the "overlap band" artifact, which was set in the image processing step. For the width, four locations were considered regularly spaced about the keyhole depth indicated by the red lines in Fig. 26. The first, second, third, and fourth red lines correspond, respectively, to the width at the top, quarter, halfway, and three-quarter distance of the keyhole depth. Here, it was assumed that the width at the top of the keyhole in the processed image was the same as the width at the keyhole entrance. This was because any additional cropping to remove the "overlap band", which was limited to a maximum of 5 pixels, will have an insignificant effect on the width at or around the keyhole entrance. The resulting aspect ratio of the keyhole was defined as the ratio of the keyhole depth to the width at the top of the keyhole. In addition, the four red lines used to define the various keyhole widths in Fig. 26 also constituted four different regions within the keyhole to define separate keyhole areas. These areas consisted of one complete keyhole area and three partial areas. The three partial areas corresponded to the areas bounded by the top of the keyhole to the quarter, halfway, and three-quarter length of the keyhole depth. We included the various subdivided keyhole widths and areas to capture more detail on the location dependent fluctuations of the keyhole. Moreover, the front wall angle, shown in green in Fig. 26, was calculated based on the angle that formed from the rightmost pixel at the very top of the keyhole and the rightmost pixel at the three-quarter depth of the keyhole. Lastly, the perimeter of the keyhole was simply retrieved from the contour of the keyhole without further modification.

Since the geometric features are collected for each frame of the series, we are able to quantify the trends that occur with time for each feature. An additional way to investigate the transient nature of keyholes is to numerically differentiate the feature of interest. This method for quantifying of the fluctuation rate was calculated using the equation below for each extracted

geometric feature and allowed data analysis to take place on both static and dynamic geometric features.

$$Fluctuation\ Rate = Feature[i^{th}] / Feature[(i-1)^{th}] \quad (31)$$

6.4 Nonparametric Data Analysis

To elucidate the correlations that may exist between the processing parameters and the extracted keyhole features, two statistical methods were employed. The first method was Spearman's rank-order correlation coefficient, r_s, which is a nonparametric method and measures the rank correlation between two variables based on the equation:

$$r_s = 1 - \frac{6\sum D_i^2}{n(n^2-1)} \quad (32)$$

In this equation, D_i indicates the difference between the ranks (x_i vs y_i) for the i*th* observation, and n is the number of observations. The r_s value ranges from −1.0 to +1.0, where values close to −1.0 or +1.0 indicate a strong relationship and values close to 0 indicate no relationship. A positive value indicates that the increase in one variable is related to the increase in the other variable, while a negative value implies that the increase in one variable is associated with the decrease in the other variable, i.e., the two variables are anti-correlated. The second method used for data analysis wa agglomerative hierarchical clustering coupled with visualization using a dendrogram [104]. As a bottom-up approach, all observations are present as their individual clusters at the start, while clustering happens gradually as it moves up the hierarchy by merging a pair of clusters at each step based on the (dis)similarities between sets of observations. In this chapter, the focus is on using average group clustering, which calculates the (dis)similarity, $d_{GA}(G,H)$, between the two groups based on:

$$d_{GA}(G,H) = \frac{1}{N_G N_H} \sum_{i\in G} \sum_{i'\in H} d_{ii'} \quad (33)$$

Here, *dii* denotes the dissimilarity between element *i* from group *G* and element *i*' from group *H*, where N_G and N_H represent the number of observations in corresponding groups. A cluster merged earlier implied that a pair of clusters were more similar in comparison with others formed later in the hierarchy, where entire hierarchical clustering can be graphically visualized with the aid of a dendrogram.

6.5 Keyhole Segmentation

In order to demonstrate the robustness of the image processing technique built for keyhole segmentation, the code was employed on four different high-speed x-ray experiments representing strongly differing keyholes shapes, as shown in Fig. 27a, 27c, 27e, and 27g. The corresponding processed images of those x-ray images are presented in Fig. 27b, 27d, 27f, and 27h. These segmented images show that the various keyhole shapes were well-preserved after undergoing the aforementioned image processing. Hence, this verified the correct operation of the keyhole segmentation code and its ability to accurately capture the keyhole shape over a wide range of processing parameters. To validate the robustness of the geometric feature extraction, manual measurements were taken of the width and depth in the raw image from each of the four different keyholes in Fig. 27a, 27c, 27e, and 27g, and compared with the width and depth measured by the process described above. This provided confirmation that the values from both the manual and automatic measurements provided a match. The outcome values from the following geometric feature extraction process are trustworthy since the structures of the keyholes are successfully segmented. This method allowed for a keyholing segmentation methodology that did not rely on the large datasets necessary for segmentation with deep learning.

6.6 Time-Resolved Geometric Features

The static and dynamic geometric features obtained from the same experiment shown in Fig. 23, 24, 25 and 26 are plotted in Fig. 27 and 28, respectively. Fig. 27 plots each static geometric feature value versus the corresponding frame number along with a dashed and dotted line representing the mean and twice the standard deviation, respectively. These reveal the time-dependent nature of each geometric feature during laser processing along with a quantitative measure of the variance indicated by the 2σ value. Likewise, the calculated fluctuation rate of each geometric feature for the same experiment is in Fig. 29. These plots show occasional spikes that may be correlated with a processing defect such as pore formation or spatter ejection.

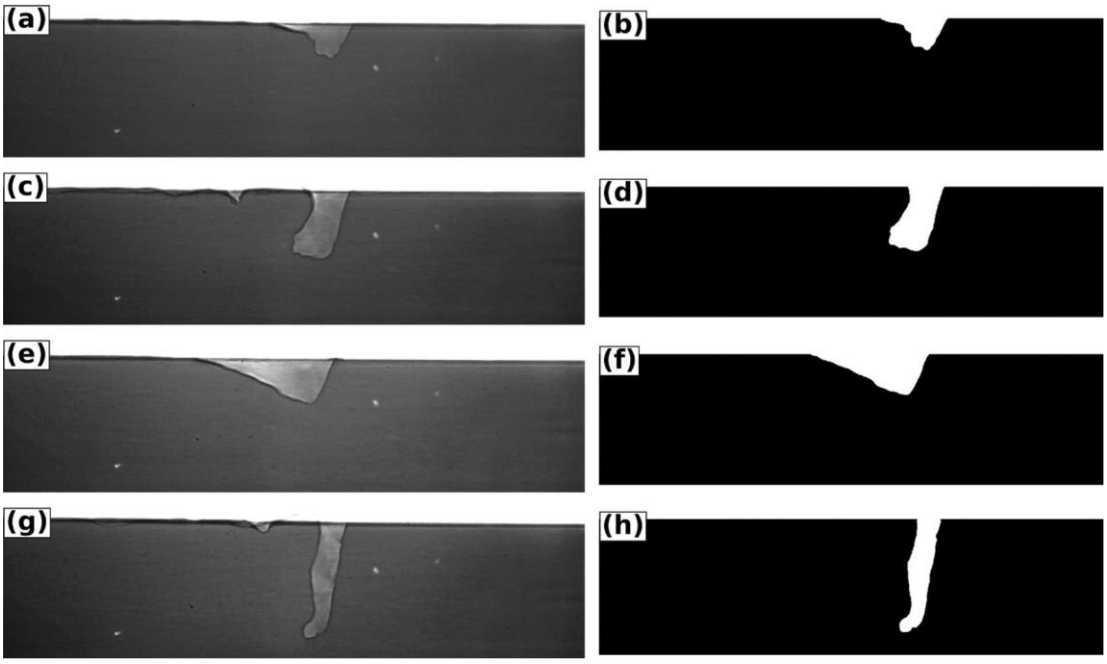

Fig. 27. Gray-scaled raw images from laser melting experiments performed on Ti64 with (a) a power of 197 *W*, velocity of 0.6 *m*/*s*, and beam diameter of 65 *_m* ; (b) after image processing of (a); (c) a power of 426 *W*, velocity of 0.9 *m*/*s*, and beam diameter of 65 *_m* ; (d) after image processing of (c); (e) a power of 426 *W*, velocity of 1.2 *m*/*s*, and beam diameter of 65 *_m* ; (f) after image processing of (e); (g) a power of 426 *W*, velocity of 0.6 *m*/*s*, and beam diameter of 65 *_m* ; (h) after image processing of (g)

6.7 Spearman's Correlation

To map correlations between the extracted geometric features with different process parameters, it was necessary to apply the Spearman's correlation on the obtained data, i.e., the mean value of each static geometric feature. In addition to the typical processing parameters, such as power (P), velocity (V), and beam diameter (D), several combinations of these parameters, such as energy density ($P/VD^2(\pi/4)$) and power density ($P/D^2(\pi/4)$), were also added to the analysis. The process parameters used for all 14 x-ray imaging experiments are given in Table 4. The results are presented in Table 5, where values more/less than +/− 0.85 are highlighted in bold.

It is apparent that there is a strong positive dependency between depth and front wall angle with any combination of parameters that include P/V, regardless of whether the spot size (D) is included or not. However, the aspect ratio (depth divided by the top width) only had a high correlation (0.87) with P/V and not any of the parameters that include spot size (D). The keyhole area and perimeter both showed strong positive dependency with laser power (P). Lastly, the velocity (V) was the only parameter that had a moderate impact on the keyhole width, with all other parameters having poor correlation. Furthermore, the addition of feature $P/\sqrt{VD}$ highlights strong positive dependency with depth, which agrees with the work conducted by Gan *et al.* [105] where they found that keyhole depth scales linearly with $P/\sqrt{VD}$. It should be noted that the size of the dataset only consisted of 14 x-ray imaging experiments, so the trends observed are indicative rather than definitive. Furthermore, the number of geometric features of the keyhole in each frame can be expanded to find more meaningful relationships between the variables. Despite the small data quantity, the potential for a comprehensive quantitative analysis based on this novel image processing and feature tracking method is evident from the initial results.

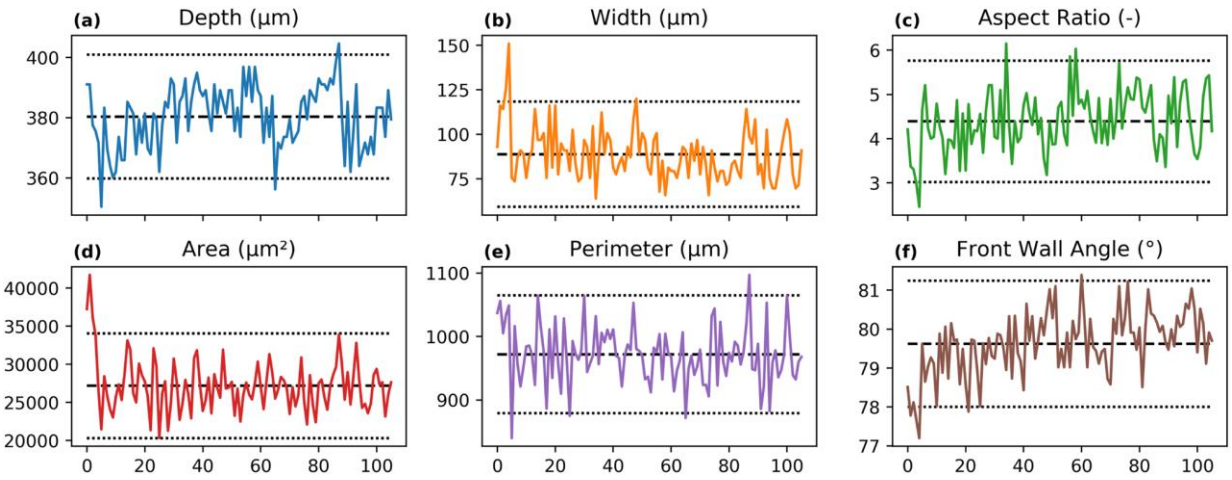

Fig. 28. Time-resolved plots depicting the static features for the keyhole (a) depth; (b) top width; (c) aspect ratio of the depth over the top width; (d) whole-body area; (e) perimeter; (f) front wall angle. The x-axis represents the frame number which were captured every 20 μs

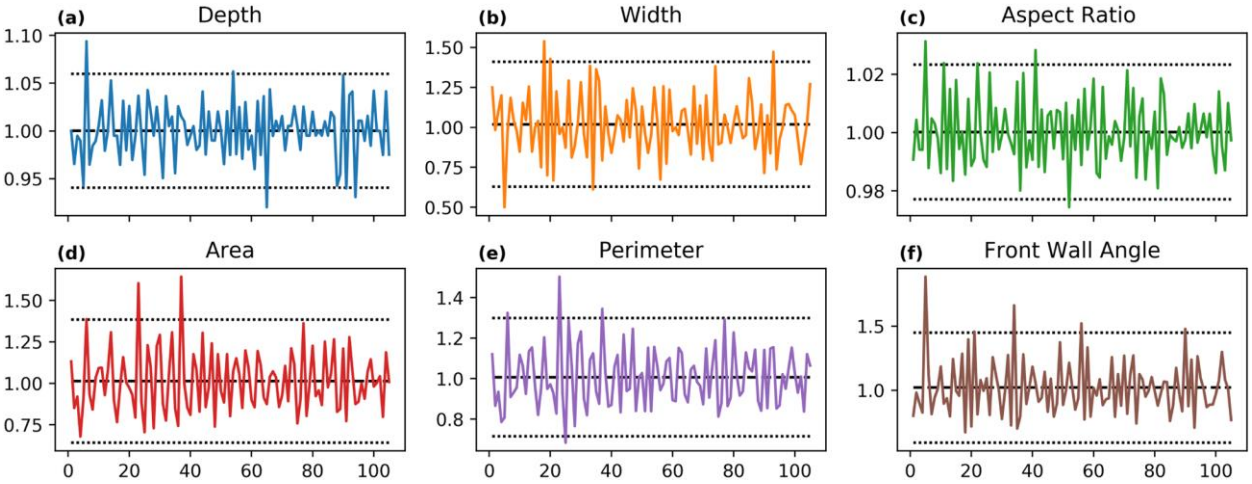

Fig. 29. Time-resolved plots depicting the dynamic features for the keyhole (a) depth; (b) top width; (c) aspect ratio of the depth over the top width; (d) whole-body area; (e) perimeter; (f) front wall angle. The x-axis represents the frame number which were captured every 20 μs

TABLE 4
The Process Parameters of the 14 Samples

Sample	**P(W)**	**V (m/s)**	**D (μm)**
Sample 1	426	1200	74
Sample 2	139	400	74
Sample 3	540	1200	74
Sample 4	311	800	74
Sample 5	426	700	65
Sample 6	426	900	65
Sample 7	426	1200	65
Sample 8	426	600	65
Sample 9	426	800	65
Sample 10	197	700	65
Sample 11	197	900	65
Sample 12	197	500	65
Sample 13	197	600	65
Sample 14	197	800	65

TABLE 5
Spearman's Rank-Order Correlation Coefficients Between Measured and Derived Variables

Metric	**P**	**V**	**D**	**$P/D^2(\pi/4)$**	**$P/VD^2(\pi/4)$**	**P/VD**	**P/V**	**$D^2(\pi/4)$**	**$V/D^2(\pi/4)$**	**$P/\sqrt{VD}$**
Depth	0.706	-0.051	-0.196	0.817	**0.974**	**0.987**	**0.975**	-0.196	-0.035	**0.930**
Top Width	0.209	0.721	0.235	-0.025	-0.464	-0.450	-0.495	0.25	0.721	-0.178
¼ Width	0.340	0.801	0.275	0.117	-0.367	-0.341	-0.363	0.275	0.767	-0.037
½ Width	0.560	**0.865**	0.318	0.426	-0.064	-0.002	-0.011	0.314	0.809	0.253
¾ Width	0.717	0.554	-0.118	0.766	0.498	0.512	0.453	-0.118	0.561	0.626
Front Angle	0.600	-0.171	-0.274	0.748	**0.982**	**0.974**	**0.931**	-0.275	-0.137	**0.868**
Aspect Ratio	0.281	-0.510	-0.078	0.440	0.824	0.846	**0.867**	-0.078	-0.519	0.631
Perimeter	0.849	0.316	-0.235	**0.858**	0.771	0.767	0.691	-0.235	0.393	**0.886**
Area	**0.938**	0.469	0.000	**0.886**	0.714	0.763	0.713	0.000	0.450	**0.900**
¼ Area	**0.856**	0.296	-0.1961	**0.955**	**0.850**	**0.877**	0.847	-0.196	0.320	**0.947**
½ Area	0.839	0.296	-0.1961	**0.950**	**0.850**	**0.877**	0.845	-0.196	0.305	**0.930**
¾ Area	0.764	0.134	-0.235	**0.900**	**0.925**	**0.934**	**0.893**	-0.235	0.153	**0.921**

6.8 Agglomerative Hierarchical Clustering

In addition, to visualizing the results in Table 5, agglomerative hierarchical clustering was applied on the Spearman correlation coefficients. Here, a 22x22 correlation matrix (12 measured features, 10 processing parameters with their derivatives) became the input to hierarchical clustering, where an individual column or row represents dependency of a single parameter on the rest of the variables. This governs the relative position of the variables on a dendrogram as a function of their dependency with all the variables, not just one. A dendrogram (Fig. 30) visualizes the calculated dissimilarities, where a shorter distance (x-axis, i.e., moving from right to left) implies more similarity, resulting in a cluster forming sooner [106]. For instance, in Fig. 30, the depth is more related to P/VD than P/V, which is also conveyed by the Spearman rank-order correlation in Table 5. On the other hand, the position of beam diameter highlights that it is not correlated with keyhole geometry features. Here, because of visualization, the interpretation of data is quick and often provides useful insights. To summarize Fig. 30, (1) the light blue cluster captures width dependence on velocity and depth; (2) next, the top green cluster captures the area dependency on power density; (3) similarly, the next sub green cluster captures the depth and front wall angle dependence on the P/V ratio and their variations; (4) followed by, the perimeter and area dependency on power; and (5) the aspect ratio's weak dependence on the P/V ratio.

6.9 Summary of Geometric Feature Tracking

The quantitative outcome values from the keyhole geometric feature tracking method and the following data analysis enabled discovery of trends between the process parameters and the keyhole geometric features. Separately, referring to the binary correlations in Table 5, both keyhole depth and front wall angle are inversely proportional to laser scan velocity [103].

The calculated trends shown in Table 5 agree with those discovered by Cunningham et al. [104] where they propose that front wall angle depends on laser power density over laser scan speed. However, Table 5 does not show a strong correlation between power density, ($P/D^2(\pi/4)$), and front wall angle. When compared to previous work by Gan *et al.* [105], the current study showed that depth is more strongly correlated with P/VD than with $P/\sqrt{VD}$. One thing to note is that work used the absorbed power in the material which they were able to estimate using absorptivity simulations, rather than the raw laser power, which is what is used in this work. This discrepancy will most likely increase the correlation to the $P/\sqrt{VD}$ combined parameter.

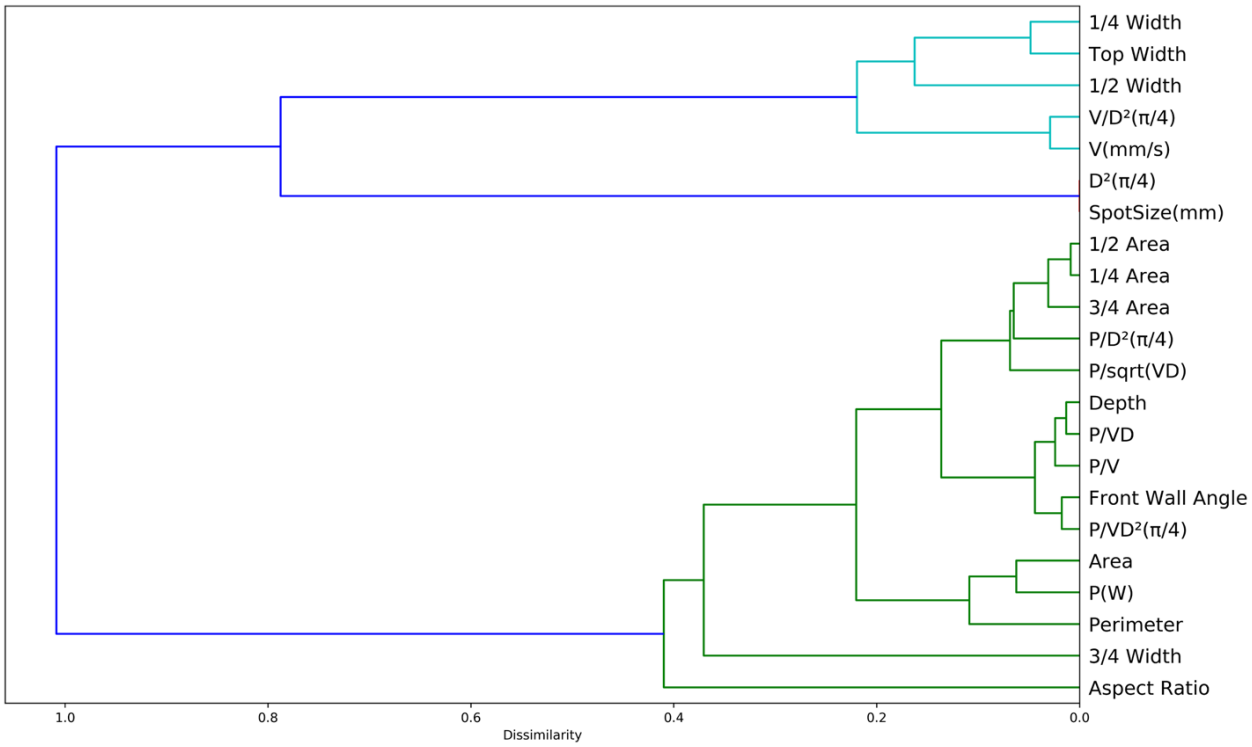

Fig. 30. A dendrogram showing the calculated dissimilarities between the extracted keyhole geometric features and processing conditions for 14 laser melting experiments on Ti64

These results indicate that uncharted trends between process parameters and keyhole geometric features or correlations among keyhole geometric features can be discovered through this proposed method. In particular, this methodology may be able to link the occurrence between geometric fluctuations of the keyhole and localized events such as a pore pinching off at the bottom of the keyhole. In addition, since the laser absorptivity is highly dependent on keyhole geometry [107], this method may be effective for quantifying the changes in the laser absorptivity throughout the fluctuations of the keyhole so that a direct correlation between keyhole geometry and laser energy absorptance can be revealed experimentally.

The following conclusions can be drawn based from this chapter. 1.) A novel image processing routine was developed to quickly and accurately segment keyholes regardless of their morphology. The method relies on noise removal, histogram normalization, and morphological transformations and was developed to be effective on a variety materials/ experiments with minimal changes. In this way, the method is material agnostic because of its focus on the manipulation of pixel intensity distributions in order to preserve keyhole shapes and facilitate thresholding. 2.) Feature extraction was employed in order to measure and quantify a variety of geometric descriptors of the keyhole throughout its evolution. Because these descriptors are solely a function of geometry, several new features were added to the data collection. Feature tracking of a fully segmented keyhole across many frames offers a more complete understanding of the highly transient nature of keyholing thanks to its comprehensiveness. 3.) Twelve different geometric features were tracked for numerous frames in 14 different experiments resulting in a large data set relative to previous manual analysis. Preliminary data analytics was applied with two different statistical tools to discover relationships between the process parameters and geometric features of keyholes. While strong correlations were discovered in the analysis, more data should be collected

over a wider range of process parameters and material systems to increase confidence in the observed trends. 4.) The manipulation of the proposed image processing pipeline followed by feature extraction is the salient point of this chapter. Thus, the presented method for quantifying the keyhole geometry from in situ x-ray videography can be used as an input to a wide variety of data analysis methods to discover meaningful relationships between the variables. Furthermore, the power of this pipeline is that one can add additional user-defined geometric descriptors by manipulating the targets of extraction which adds to its versatility.

CHAPTER 7

IMAGE SYNTHESIS USING GAN AND CGAN

7.1 Image Generation Opportunities

As has been discussed, the effort of LPBF process characterization and subsequently part certification faces a shortfall in the amount of data available for which to thoroughly understand these processes. The DXR image capturing methodology has been extremely beneficial for providing data in-situ, at a high resolution where inspection of the build quality microstructures can be conducted. The ultra-fast speed of the imaging provides a large dataset to investigate, too large for human inspection in a reasonable timeframe. The advancements of artificial neural networks which can process thousands of images quickly has turned this issue into an opportunity, as these machine learning approaches can also mine that data for patterns that would be otherwise imperceptible to humans – even if there was enough time and manpower for a visual inspection of all of the images. But while the volume of data is substantial, the variety of that data is lacking. The limited number of experimental builds under which the DXR apparatus captured imaging was limited. There is a need for more data, both in terms of new combinations of process parameter combinations, and for new images of those parameter combinations that were collected. These additional data can help inform and validate thermal models for the resultant physical characteristics of the build objects, as they relate to the size and dimensions of those microstructure objects visible in the DXR data.

The lack of experimental data to evaluate the in-situ process in LPBF additive manufacturing can be alleviated by exploring machine learning methods that can approximate and even generate data representations. In order to adequately characterize an LPBF build in terms of quality, there must be a comprehensive understanding of the nature of the defects induced during

the manufacturing process. At the scales involved for these defects (in terms of micrometers), humans alone cannot evaluate those defects without some advanced imaging techniques. The focus of this work was therefore to develop a machine learning paradigm that can be generalized to learn from the limited data available, to interpolate from it, and ultimately develop new data computationally that can inform research into the microstructure quality at the time of the build. This involved the usage of a class of generative models known as the conditional generative adversarial network, or CGAN – a modified system architecture from the more general GAN approach. For this work, the goal was to use the generative model to create visual approximations of microstructures for combinations of process parameters that were not experimentally produced under the DXR imaging.

7.2 GAN Modeling

An initial GAN was developed that utilized a generator with one hidden layer, which took as an input *z*, and yielded a 28x28x1 pixel sized image. The hidden layer used the Leaky Rectified Linear Unit (ReLU) activation function. Unlike the standard ReLU function which would map a negative input to a 0, Leaky ReLU allows for a small positive gradient. Meanwhile, the output layer of the network utilized the tanh activation function, which was used to scale the output values in a range of -1 to 1, which has been shown to produce crisper images in relation to results obtained from other activation functions [108]. The discriminator network was developed with a two layer neural network, with 128 hidden units and again, the Leaky ReLU activation function. Binary cross-entropy was used for the loss function during model training, which was minimized by measuring the difference between computer probabilities and actual probabilities (which had two possible outcomes for classes for predictions matching the labels). Optimization was achieved

using the Adam optimization algorithm, an advanced gradient descent optimizer based on adaptive moment estimation, which itself is a method of stochastic optimization [109].

For consistency and balance in the data, 2,000 images were used for each class – that is, each combination of laser velocities (0.2, 0.4, 0.6, 0.8, 1.0, 1.2, and 1.4 meters per second) and laser intensities (150, 200, 250, 300, 350, and 400 Watts).

7.3 CGAN Modeling

Modifications are possible to the general form of the GAN network architecture such that performance and accuracy can be improved upon [110]. The use of the GAN to generate vapor depression and melt pool geometries can be supplemented by incorporating the process parameter values into the artificial learning process. The Conditional GAN is particularly well suited to learn not only from the image training data, but incorporate numerical inputs in the form of *y* into the generator network, as described previously. In this method, values were inputted, where *v* could be 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, and 1.4 for the training data as these were the velocity values that were experimentally performed (in meters per second). Likewise, values of laser intensity were inputted, where *p* (or power in Watts) could be 150, 200, 250, 300, 350, and 400.

Using the previously programmed GAN, modifications were made to incorporate the new input requirements, where embedding and an elemental multiplication step was used to combine the random noise vector *z*, and the numerical input labels, *y*, into a joint representation. This happens by taking in the *y* value as an integer value, and turn it into a vector of size equal to the length of the random noise vector using the function in Keras called Embedding layer. From here the embedding layer was combined with the noise vector, as mentioned above, using the Keras Multiply layer, which Keras uses to multiply the corresponding entries of two-equal length vectors together to create one single vector that is the product of those original two [111]. This is then fed

into the generator network, from which a new image is generated. The overall step by step framework of the network is listed below. As previously noted, this was coded in Keras and is based on, with several modifications, a CGAN used for image generation [111].

- Development of the input layer
- Development of a transposed convolution layer, which transforms the input from a 7x7x256 into a 14x14x128 tensor
- Application of batch normalization
- Application of the Leaky ReLU activation
- Development of another transposed convolution layer, which now transforms the input from a 14x14x128 into a 14x14x64 tensor
- Another application of batch normalization
- Another application of the Leaky ReLU activation
- Development of a third transposed convolution layer, which transforms the input from a 14x14x64 into a 28x28x1 tensor
- Development of output layer with the tanh activation function
- Input of random noise vector *z*
- Input of conditioning label as an integer
- Label embedding step to turn labels into dense vectors
- Flattening step to embed 3d tensor into 2d tensor
- Calculation of the element-wise product of the vectors z and the label embeddings
- Output which generates an image for the given label

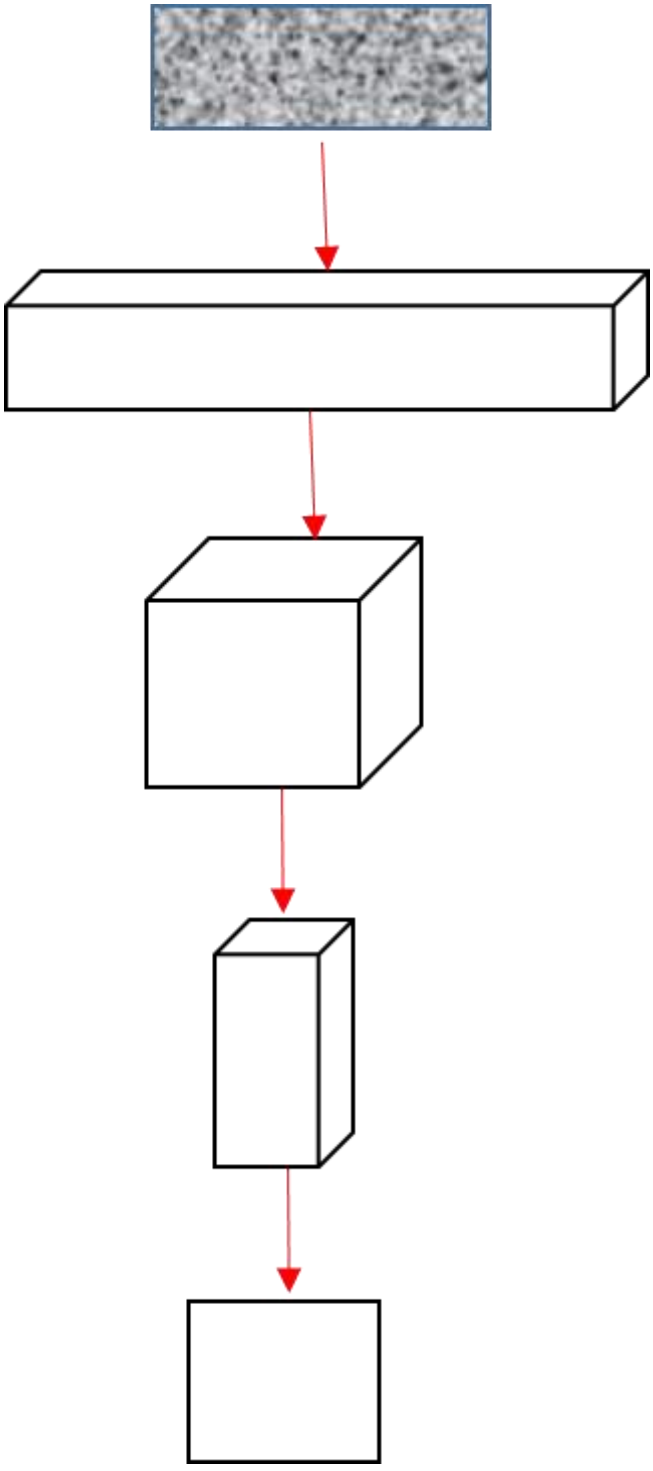

Fig. 31. GAN layers

The discriminator network is also similar to that of the general GAN with a couple of exceptions, also involving how its input is received. For the discriminator, the input is a three-dimensional image rather than the flat vector that the generator receives as its input (see Fig. 31). Just like the generator network, the discriminator uses the Keras Embedding layer call to accomplish this. It takes a label, in the form of an integer, and uses that Embedding to transform the label into a dense vector of size 28x28x1 = 784, which is the length of the flattened input image [111]. The label embeddings then must be reshaped into the image dimensions, which was 28x28x1. From here, the label was reshaped by being concatenated onto the corresponding image, which created a joint representation with the appropriate shape of 28x28x2. That last digit 2 represents the embedding on top of the image. Finally, that concatenated image-pair is fed into the neural network of the discriminator, which is made to take as input the 28x28x2 shape that it receives. The overall step by step framework of the network is listed below. Like the generator, this framework was based on, with some modifications, a CGAN used for image generation [111].

- Convolutional layer to transform input from 28x28x2 into 14x14x64 tensor
- Leaky ReLU activation
- Convolutional layer to transform input from 14x14x64 into 7x7x64 tensor
- Batch normalization
- Leaky ReLU activation
- Convolutional layer to transform input from 7x7x64 into 3x3x128 tensor
- Batch normalization
- Leaky ReLU
- Output layer with the sigmoid activation function
- Input image is received and label

- Label embedding turns labels into dense vectors into tensor with size 28x28x1
- Flattening of 3d tensor from the image into 2d tensor of size 28x28x1
- Reshaping of label embeddings to match the dimensions of the input image
- Concatenation of label-pair
- Output classification of the image-label pair

By incorporating a continuous feature representation into the training, the model can potentially learn the distributional relationships of that feature with regard to the underlying principals governing that representation. For instance, the thermodynamics and physics that inform the fluid nature of the gaseous vapor depression and the liquid melt pool formations. While the laser intensity for each build is relatively constant, the heat at the surface of the build is changing over time, due to buildup in energy as the laser moves across an area. Changing the laser intensity or velocity across the build surface will therefore have a direct impact on the physical features of the in-situ build. Generating by incorporating this continuous representation would also have a direct impact on LPBF applications as complex builds will be a constant focus; to create objects that can be utilized in aerospace projects. As described previously, the heat buildup in the corners and crevices of these builds as the process develops from layer to layer can have a deleterious effect on the microstructural stability. Thus, involving a feature representation for the thermal variations in the build can improve the CGAN's ability to accurately model and generate those new microstructure representations. As with the GAN model, 2,000 images were used for each class – that is, each combination of laser velocities (0.2, 0.4, 0.6, 0.8, 1.0, 1.2, and 1.4 meters per second) and laser intensities (150, 200, 250, 350, and 400 Watts). The next section will describe the images that were produced as a result of building this model.

7.4 Generated Image Analysis

Outputs from both the GAN and CGAN were analyzed to determine how closely these models were able generate new data to depict the LPBF process, focusing on the vapor depression area from the heat affected zone (HAZ). There was a GAN model run with identical parameters for each collection of training data at each laser velocity setting (0.2, 0.4, 0.6, 0.8, 1.0, 1.2, and 1.4 meters per second) for each power setting. Fig. 32 shows a visual example of model outputs, where the training data included only images collected experimentally at 400 Watts. Likewise modeling with the CGAN was performed similarly, with the exception of the inclusion of the aforementioned *y* values. For the 400W collection of data, a random sample of 15 images were inspected from the real data, the GAN output, and the CGAN output. Contouring using OpenCV in Python allowed for calculations of the pixel area for each image, which is depicted in Fig. 33. The outer boundary of each vapor depression was established, within which the area of the object could be calculated by a count of the pixels within that boundary.

A visual inspection of the images produced by both the GAN and the CGAN was also conducted, to determine if human observers could detect the differences between those generated images. Calculations of the area alone could be uninformative in that the areas of the generated images could still be similar to the real data while also providing objects with extremely divergent shapes. However, this turned out to not be the case as the images sampled all resembled the characteristics of an appropriately shaped vapor depression from an LPBF build in-situ. This qualitative test helped to accept the data output from the models prior to more rigorous quantitative evaluations to determine the statistical properties of those data.

Fig. 32. Real image (left), GAN generated image (center), and CGAN generated image (right)

Images generated were also generally noise free. Despite the denoising steps described in Chapter 3, many of the training data images still retained random individual white pixels in the images. Yet the output images from the GAN and CGAN had noticeably less pixel noise at a visual inspection. This suggested that the models' capabilities for mapping latent features did not include noise as a feature. The models therefore learned the relevant information from the images, and were able to disregard such irrelevant information from training.

7.5 Train on Synthetic Test on Real

Testing the performance of the GAN and the CGAN over 1,000 epochs started with a novel methodology where a supervised learning task can be defined on the domain of the training data [112]. The generated images of the entire HAZ were considered, which depicted the vapor depression areas and the melt pool areas from the build plate. This procedure uses the output of the GAN to train a secondary model, which is also tested on a hold-out portion of the original data – known as Train on Synthetic Test on Real (TSTR). The appeal of this method of evaluation is that it helps to demonstrate the ability of the synthetically derived data from the two GANs to be used in real-world applications where there may be many features in the training data upon which the synthetic observations are made.

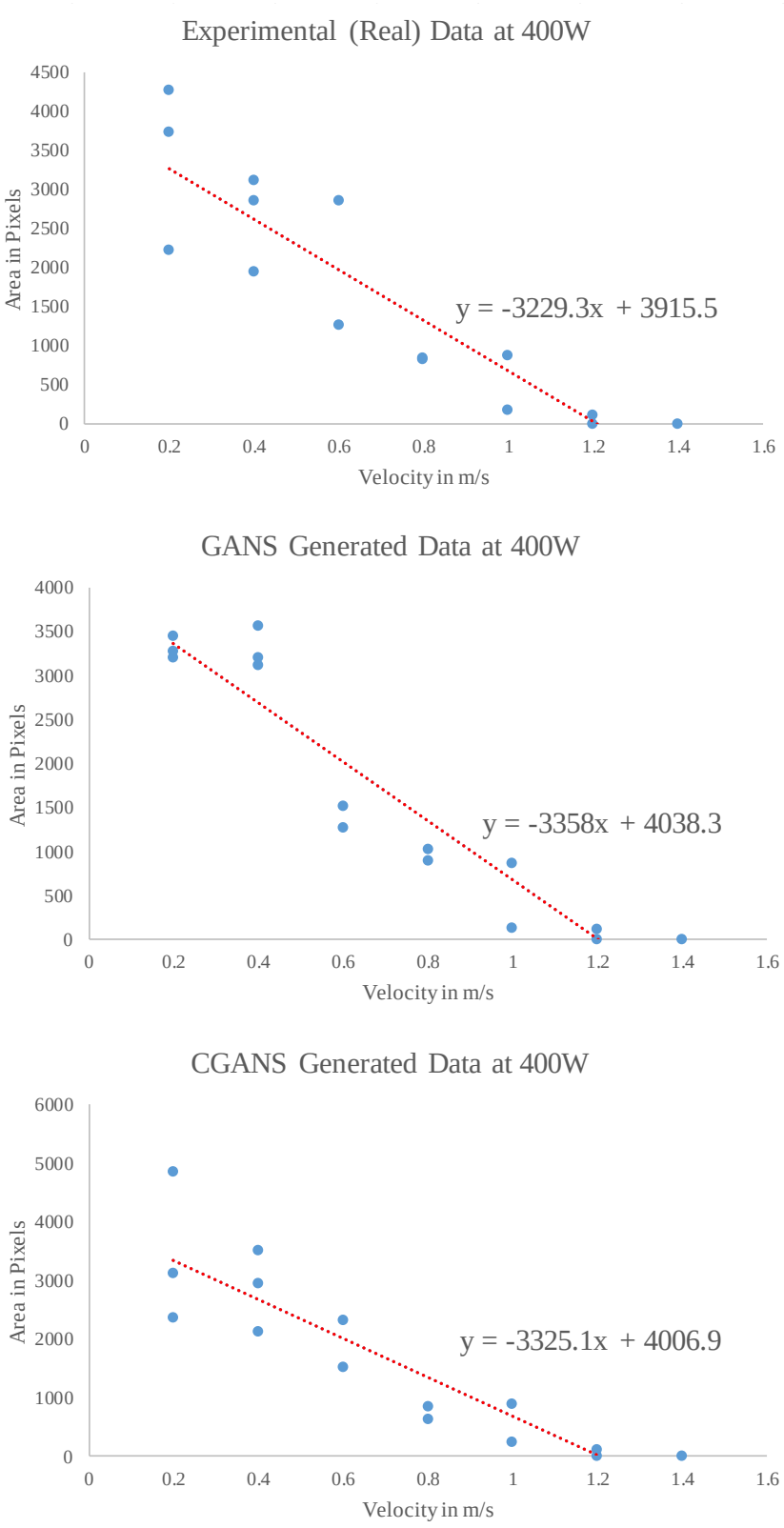

Fig. 33. Area measurements of vapor depression images for real data (top), GAN generated data (center), and CGAN generated (bottom)

A random forest classifier, a supervised machine learning classification model, was used to determine whether a given image was real or synthetic. The performance of that classifier, when trained with the original data and then trained with the synthetic data, was used to provide the area under the precision-recall curve (AUPRC) and area under the receiver operating characteristic curve (AUROC). These values show how well the classifier was able to correctly identify a real image as being real. That same classifier is then utilized to determine if an artificial image generated by the GAN and CGAN were "real", which is to say, did the classifier conclude that an artificially created image was so close to a real image that it was indistinguishable. These metrics are provided in Table 6.

TABLE 6
AUPRC and AUROC from TSTR

	GAN	**CGAN**
AUPRC		
REAL	0.8135	0.8677
TSTR	0.7569	0.7901
AUROC		
REAL	0.7745	0.7925
TSTR	0.6155	0.6678

7.6 Intersection Over Union

Intersection over Union (IoU) is a commonly used procedure for many image processing tasks. Two images, or objects, when evaluated against one another, will be evaluated on how much common overlap there is among those two images (Fig. 34) [113]. The goal is to quantitatively evaluate the overlap between two images, to see how much they differ. In this case, the two images

used were a real observation and the artificially generated observations from the GAN and CGAN (see Fig. 35). The calculation for the IoU is:

$$IoU = \frac{(Area\ of\ Intersection\ of\ Two\ Objects)}{(Area\ of\ Union\ of\ Two\ Objects)} \tag{34}$$

This can also be expressed as:

$$IoU = \frac{(A \cap B)}{(A \cup B)} \tag{35}$$

IoU evaluates the number of pixels that are correctly attributed to a particular class and is defined by the equation above. The number of pixels that overlap between the ground-truth mask and the predicted mask is denoted by $A \cap B$ (intersection), and the number of pixels that are occupied by at least one mask is denoted by $A \cup B$ (union).

The average IOU score for 100 generated images that were compared against real images was calculated and presented in Table 7. This provided a quantitative method for examining how much the artificial images produced by both generative networks had in common with the real images.

7.7 Hausdorff Distance

The Hausdorff distance (HD) is another quantitative metric used for evaluating the closeness of two images. It is a non-linear operator, which measures the amount of non-matching in two different sets of points [114]. Essentially, the HD is looking at how much each point on one image set lies near some point on another image (Fig. 36). The uniqueness of this approach is that the goal is to evaluate closeness of a point and many points in the second image, that is, it is not looking for an exact corresponding point in both images. This makes the algorithm more robust to deviations in points in the images, as proximity is more important than an exact location. Additionally, the metric is particularly useful when boundaries are more important than the area which is the case when evaluating the outlines of the HAZ in the build dataset of images.

Calculation of the HD is derived from finding the distance between two points, *a* and *b*, in this case, in a Euclidean space [115]:

$$d(a,b) = \|a - b\| \tag{35}$$

The distance between point *a* and a set of points *B* is given by:

$$d(a,B) = \min_{b \in B} d(a,b) = \min_{b \in B} \|a - b\| \tag{36}$$

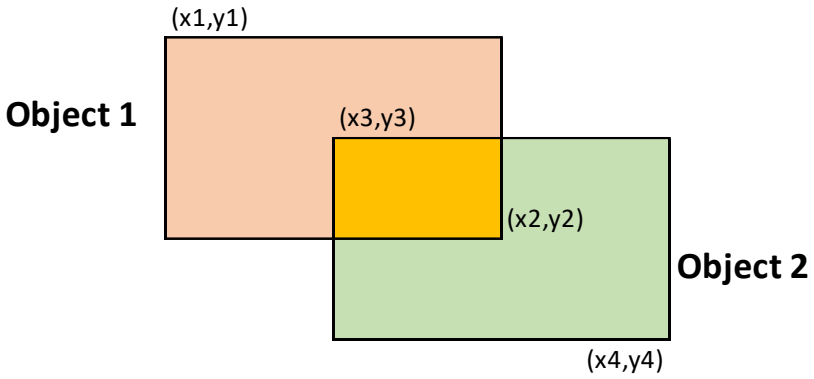

Fig. 34. Intersection over Union approach for overlap

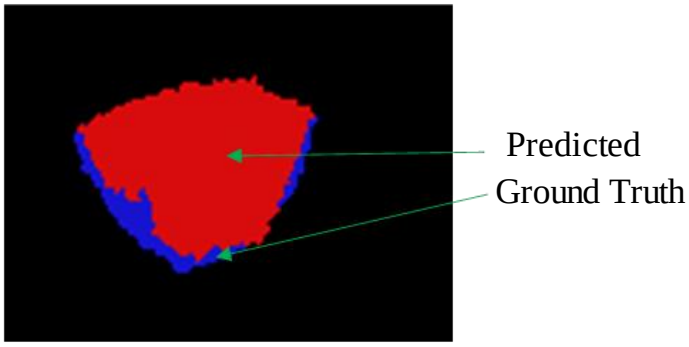

Fig. 35. IoU overlap of predicted (red) and ground truth (blue) images

The directed distance between a set of points *A* and a set of points *B*, referred to as *h(A,B)*, then becomes:

$$h(A,B) = \max_{a \in A} d(a,B) = \max_{a \in A} \min_{b \in B} d(a,b) = \max_{a \in A} \min_{b \in B} \|b - a\| \quad (37)$$

So that the Hausdorff distance can be calculated by:

$$H(A,B) = max(h(A,B), h(B,A)) \quad (38)$$

For segmentation of vapor depressions, all pixels in the training images and in the generated images that comprise that object of interest on the edge of the predicted mask will be paired to the nearest neighbor of the ground-truth mask and vice-versa. Out of these nearest neighbor pairs, the pair with the largest distance between them— i.e., the greatest mismatch between the ground-truth mask and the predicted mask—make up the Hausdorff points, with the HD being the distance between those nearest neighbors.

A summary of the results for 100 images generated by the GAN and CGAN that were compared against ground truth images is presented in Table 7. As this metric looked for the largest distance between two sets of nearest neighbors, the units in the metric were in number of pixels.

7.8 Maximum Mean Discrepancy

The goal of a generative model, in this case the GAN or the CGAN, is to learn the underlying features that make up the distribution of the data, so that it can take from that distribution the means with which to represent that data. If this distribution is learned appropriately, the artificial representations that the generator produces will closely match what the real data shows; the better it learns the distribution, the better it is able to make new images. Therefore, evaluations of the output from the GAN and CGAN should include a metric that directly evaluates the discrepancy in the distribution of the data. For this reason, Maximum Mean Discrepancy (MMD) is ideal, which quantifiably judges whether two data sets – in this case the

real data and the generated data – were generated by the same distribution [116]. It is a statistical test that uses a kernel-based approach for evaluating the sameness of the distributions [117].

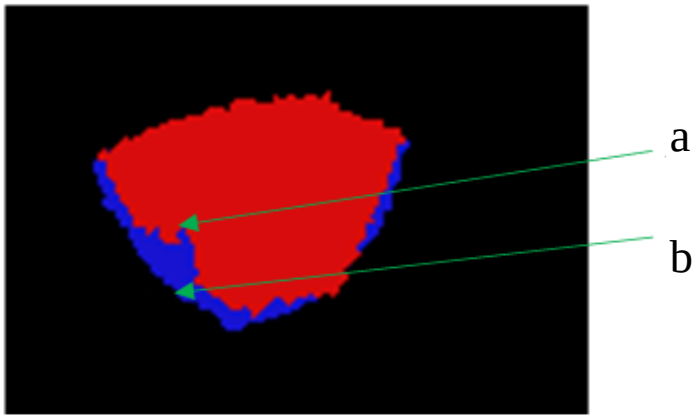

Fig. 36: HD between two points located by the end of the arrows

TABLE 7
IoU and HD Measurements

	IOU	**HD**
GAN	0.9345	120
CGAN	0.9562	95

Formally, MMD can be defined as a probability metric to calculate the distance difference between feature means [118]. If given a set of variables X, a feature map ϕ is then generated over

that set which maps X to another space, F such that $\phi(X) \in F$. The feature means can be calculated by taking the probability measure P on X, which is a set of variables, and generating a feature map that takes $\phi(X)$ and maps it to every other coordinate of $\phi(X)$ in this fashion:

$$\mu_\rho\big(\phi(X)\big) = [\mathrm{E}[\phi(X_1)], \ldots . [\mathrm{E}[\phi(X_n)]]^T \tag{39}$$

While the inner product of the feature means of $X \sim P$ and $Y \sim Q$ is expressed as the kernel function:

$$\big\langle \mu_P\big(\phi(X)\big), \mu_Q\big(\phi(Y)\big)\big\rangle_F = E_{P,Q}[\big\langle\big(\phi(X)\big), \big(\phi(Y)\big)\big\rangle_F] = E_{P,Q}[k(X,Y)] \tag{40}$$

From this, the MMD can be obtained for X and Y to calculate the distance between the feature means of X and Y:

$$MMD^2(P,Q) = \|\mu_P - \mu_{PQ}\|_F^2 \tag{41}$$

From which equation the above is used to make the expression:

$$MMD^2(P,Q) = E_P[k(X,X)] - 2E_{P,Q}[k(X,Y)] + E_Q[k(Y,Y)] \tag{42}$$

In order to properly define the kernel function, previous work has shown success using a radial basis function (RBF), using the Frobenius norm between vectors [119]:

$$K(x,y) = \exp\left(-\|x-y\|^2 / (2\sigma^2)\right) \tag{43}$$

Using the MMD metric to evaluate the real data and the GAN generated data yielded a value of 0.33, with a sample of that distribution depicted in Fig. 37. The calculation for the real data and the CGAN data, which yielded an MMD value of 0.18 is likewise depicted.

7.9 Avoidance of Model Overfitting

During the training procedure, it is possible for the GAN to essentially learn to completely memorize the data during training, and simply reproduce that memorization in its output [120]. The MMD method of evaluation has the benefit in that it can be used to evaluate model overfitting, that is, learning to match the real data distribution too exactly. To determine overfitting, a null

hypothesis was constructed, that the MMD between the generated data and the experimental data is at most as large as the MMD between the generated data and the holdout training data. Thus, a MMD three sample test was constructed whereby *X* represented the generated samples, *Y* represented the test set, and *Z* represented the training set [121]. These sets were identified as such under the logic that if the MMD between *X* and *Y* is less than or close to the MMD between *X* and *Z*, that would indicate that there was as much closeness to the real data from the generated data, thereby indicating the existence of data memorization. The expectation for the hypothesis is that MMD(synthetic, test) ≤ MMD(synthetic, train) will be false. This is essentially a test for a null hypothesis that the model has not memorized the training data, and if that can be rejected [110]. Upon running this test for the GAN generated data, the average p-values were 0.27, and for the CGAN generated data the average p-values were 0.33. This indicates that the null hypothesis should not be rejected, that the MMD between the synthetic set and the test set is at most as great as the MMD value between the synthetic set and the training set. The artificially generated samples did not look more identical to the real data than they did to the test set, which indicated that neither the GAN nor the CGAN was an overfit model.

Using MMD alone cannot guarantee avoidance of overfitting as this method may not be sensitive enough to differences in the distributions [122]. However, the visualization shown in Fig. 37 from a sample distribution does indicate that there is a difference between the distribution of the real data and those generated by the two GAN models. Those distributions for the GAN and CGAN distributions are similar to one another, yet both have distribution patterns different from the real data, as measured and quantified. The test therefore verifies that the model was not overfitting the data, and that the generated images are in fact unique.

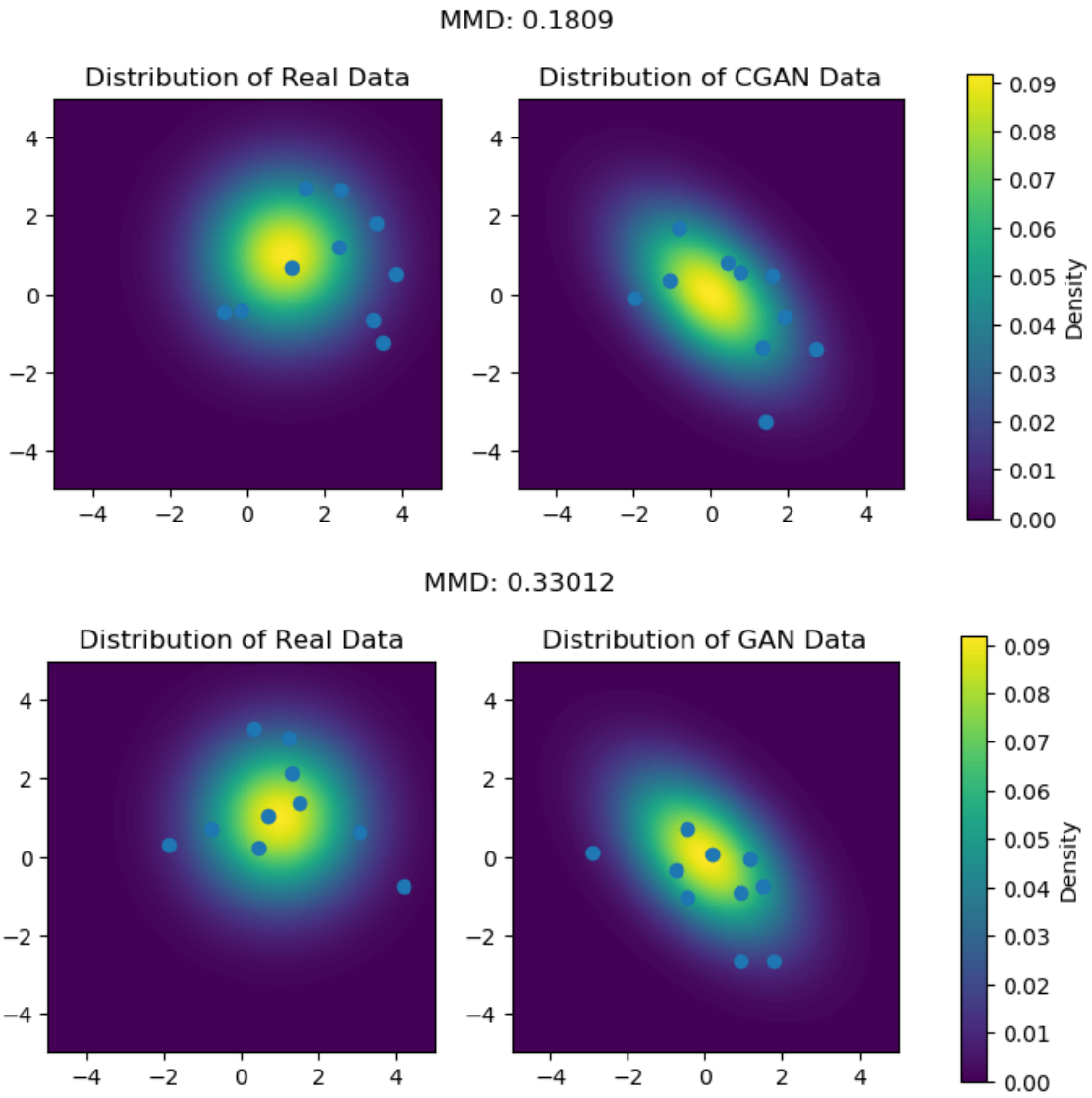

Fig. 37. Sample distributions in comparison of the real data along with both the GAN (top) and CGAN (below) generated data

7.10 GAN results with RMSE

Testing of both GAN models also used the root mean squared error (RMSE). This was chosen as a metric for evaluation as it provides a quantitative measure of the difference between

the predicted values and the real values, based on absolute difference. This therefore considers predictions that are both above and below the true value. Additionally, it is an average of the total amount of error, as opposed to the absolute average error. As vapor depression characteristics can vary widely, even when all parameters are held the same, evaluating the magnitude of the error was more appropriate than measures that evaluate based on average error where extreme values would not be penalized as highly – which is of interest in this case where large error, that is predictions very far off from the real values, are particularly of interest. RMSE can be calculated by:

$$RMSE = \sqrt{\frac{1}{2}\sum_{i=1}^{n}(\hat{r}_{ij} - r_{ij})^2} \tag{44}$$

where $(\hat{r}_{ij} - r_{ij})^2$ is the differences between the predicted value and the actual value, squared, and *n* is the total number of observations.

The RMSE values have the same value as that of the original data. In order to interpret the predictions, another metric was used in conjunction, the Scatter Index (SI). The SI value provides insight into the RMSE, the performance of the predictions, in a ratio that helps determining how far off that error actually is from the real data. SI is a normalized measurement of the error in a system, where lower values generally indicate better performance in comparison to larger numbers [123]. It is calculated by dividing the RMSE by the mean of the observations, and multiplying by 100 to achieve a percentage score. Thus, SI presents RMSE with respect to the mean value of the observations in the data. It can be expressed as:

$$SI = \frac{RMSE}{\bar{X}} \tag{45}$$

The results from these measures are presented in Table 8 for the GAN model and Table 9 for the CGAN model. Here, the RMSE was calculated for all images produced for each class of velocity setting. The RMSE and subsequently the SI were higher for area and convex hull area,

than they are for depth and width. Overall, generated data at the 0.2 and 0.6 m/s velocity setting images resulted in a lower RMSE and SI value. This is likely due to the narrower range of physical fluctuations for those builds as well as the amount of data available. RMSE and SI values for the width and depth were lower, which was also likely due to the more limited range in experimental values obtained for these metrics. The vapor depression depth and width did not have as much overall variation as the area and convex hull, which consider the overall size and shape rather than a single dimension.

Overall, the RMSE and SI results indicate the same trend as has been discussed with the previous evaluation methods, that is, that the results obtained for the CGAN were more favorable to the GAN. While many of the generated images for the GAN model seemed to be close to their original counterpart, with many SI values below 20.0, the CGAN consistently outperformed the general GAN in terms of the quality of the images produced when compared to the experimental data upon which these generated images were based. This metric of uncertainty quantification for the results of the GAN and CGAN output thereby helps to validate the usefulness of these approaches, especially for the CGAN, in developing images to represent data at lower velocities. This is already the ideal scenario, as the experimental data already suggested that velocities at the lower ranges induce deeper vapor depressions, rather than no vapor depression at all for the higher velocities. The penetration of the vapor depression through the material is necessary to ensure proper fusion of substrate particles, for a proper solidification and resultant microstructure.

7.11 Effectiveness of Generated Images

The output images for each network were near indistinguishable from the real, experimental data (examples of which were shown in Fig. 31). At a visual inspection these generated images seemed to match the data that was collected experimentally. It has been shown

TABLE 8
GAN Evaluation with RMSE and SI Metrics

Test Parameter	Area RMSE	Area SI	C.H. RMSE	C.H. SI	Depth RMSE	Depth SI	Width RMSE	Width SI
0.2m/s	299.11	28.08	417.70	30.61	28.63	29.41	22.39	16.74
0.4m/s	426.98	30.47	448.90	49.46	43.97	57.24	25.28	53.77
0.6m/s	261.13	27.66	330.45	24.51	32.19	58.05	21.20	23.20
0.8m/s	443.95	103.3	538.76	119.5	55.77	188.7	71.52	130.9
1.0m/s	312.15	302.8	309.43	301.5	78.04	199.0	46.53	68.72

TABLE 9
CGAN Evaluation with RMSE and SI Metrics

Test Parameter	Area RMSE	Area SI	C.H. RMSE	C.H. SI	Depth RMSE	Depth SI	Width RMSE	Width SI
0.2m/s	279.05	18.52	301.53	19.01	27.89	12.85	22.22	11.82
0.4m/s	315.21	20.21	397.51	22.63	39.34	19.55	38.59	17.83
0.6m/s	250.63	19.64	291.07	23.89	25.03	15.84	18.78	13.86
0.8m/s	405.12	50.25	518.88	68.27	63.78	45.69	76.60	29.35
1.0m/s	360.15	75.96	375.89	94.35	71.46	105.1	66.32	49.55

that the model was not overfitting, thus these images could reliably be used as approximations of actual DXR data for LPBF builds.

As shown in Fig. 33 previously, the output from the GAN had a much narrower range of variation in the pixel area of the objects generated, when compared to both the real data and the CGAN. This would imply that while the GAN was not overfitting in its model, it was still learning a more limited distribution in the training data, such that its output images were very similar for each velocity setting group of images. For instance, for images taken at a laser velocity setting of

0.4 meters per second, the GAN created images that ranged in pixel area of 3,120 to 3,562 pixels. Meanwhile, for that velocity in the experimentally captured images (the real, training data), the pixel area range was 1,947 to 3,122 pixels and for the CGAN it was 2,130 to 3,523. This indicates that the conditional input for *v* led the model to better map the distribution of features for the training data, to better create images that approximated the range of object characteristics better than the standard GAN, which did not have that additional condition.

The linear trend lines for the area distributions also showed more similarity in the CGAN images compared to the real images, with a slope closer to the real data when compared to the trend line slope for the GAN image areas. Although that factor in itself does not indicate similarity alone as the range for each velocity setting group of images could still fluctuate greatly yet over the entire dataset average out to create a trend line with a similar slope to the real data. Nevertheless, for the random samples taken of images collected - the data show that the CGAN was able to create images that more closely matched the dynamic range of vapor depression sizes from the experimental builds.

The novel TSTR method further explored the closeness of the images produced by both generative models in comparison to the real data. This analysis also quantifiably validates the assertion that the CGAN was able to make more realistic images compared to the GAN model. As shown in Table 7, the CGAN observations consistently resembled the real data more often than the general GAN. These results suggest the CGAN was capable of generating artificial observations that were close enough to the real data that it could then be used to train another classifier to identify those observations to a degree very close to its own performance on classifying the real data.

The IoU analysis and the HD analysis both suggest the same observation in terms of the data generated. While the metrics were similar, the CGAN did outperform the general GAN in terms of generating images that more closely matched the real data, or ground truth. Similar to the previous evaluation metrics, the CGAN consistently provided a lower HD score than the GAN. The distance of a set of points along the boundary of an object in a sampled image produced by the CGAN will generally have a smaller distance to the points along the boundary of a real image – compared to a sampled image from the GAN output, thereby indicating its performance in developing images that more closely matched the ground truth data was superior to the general GAN.

Further examination of the data generated by both networks using the MMD method justified the assertation that the models are appropriately learning the distribution of features in the underlying training data. In general, the lower the MMD, the more evidence there is that the distributions are the same [124]. As the objects in the dataset vary as the experimental parameters used to generate that data varied – both the GAN and the CGAN were able to pick up on those different expressions. And this was established while also evaluating the potential for model overfitting, a common concern in many machine learning frameworks. Indeed, it was established using the three sample MMD approach that the feature map developed in the generated datasets were significantly different from that of the real data. This speaks to the ability of these complex networks to learn effectively from the training data.

7.12 New Process Parameter Generation Results

Given the results from the outputs of the GAN and the CGAN, where the CGAN model resulted in images that were quantifiably closer to a comparison set of real images, the CGAN was used to create new artificial images of vapor depression geometries based on process parameter

combinations that were not conducted experimentally. This would result in data which could not be compared to original data, as these generated images were based on new inputs. Thus, the CGAN was also preferred as this was the model that allowed differing numerical inputs in combination to the training images. The previous results demonstrate that the model is able to learn how the vapor depression shape fluctuates with the new parameters.

The original data was collected at laser velocity settings that were spaced 0.2 increments apart, for a total of seven different settings: 0.2m/s, 0.4m/s, 0.6m/s, 0.8m/s, 1.0m/s, 1.2 m/s, and 1.4 m/s. Additionally, there were a total of five laser intensity settings that were also involved, set at 50 Watt increments: 150W, 200W, 250W, 350W, and 400W. Due to equipment failure during the experimental builds, there were no data captured for builds at 300W, and therefore the resulting data omits this experiment. The combination of the laser velocity settings and the intensity settings therefore produced a total of 35 different experimental combinations of those two parameters, from which data was collected and used for this data-driven experiment.

As there were more velocity settings than intensity settings, it was those values which were the focus of the data generation initiative. Additionally, the data show that the vapor depression metrics (i.e., depth, width, geometric area, and convex hull area) increase linearly with regards to velocity, while holding the intensity constant. That is, at 150W there is a steady increase in the metrics. The data show that when the next intensity is used, say at 200W, the vapor depression metrics drop at the lower velocity of 0.2 m/s compared to the 1.4 m/s of the previous intensity of 150W. Essentially, at each intensity setting the vapor depression starts smaller and grows bigger as the velocity is increased – until some point is reached where it begins to decrease again. It is however worth noting that the size does not continue to increase for each velocity. Instead, at 1.2 and 1.4m/s – there is a drop off to a zero reading for the metrics as the vapor depression generated

at a near-zero depth which was not enough to be measured. This was described thoroughly in previous chapters as the thermal intensity was not strong enough for the evaporation of metal to accumulate due to the laser moving too fast over the build plate. For some combinations of intensity and velocity, there were not data collected at 1.0 m/s either – which was the case at 150W, 200W, and 250W. At those intensities, the power was again not strong enough to cause the heat buildup necessary, until it got to 350W. Given the zero or near-zero vapor depressions at some combinations, the new data to be generated by the CGAN would only be produced at intensity and velocity segments in between intervals for which data was actually captured. Using this requirement, new data was generated at increments in between the velocity settings originally produced, to include: 0.3m/s, 0.5m/s, 07.m/s, and 0.9m/s. This would help to achieve vapor depressions which could be visually compared to a dataset experimentally captured just above, and just below that new data – in terms of the parameters used to express those geometries.

The new combinations of parameters that was used to create artificial data with the CGAN were therefore: 150W and 0.3m/s, 200W and 0.3 m/s, 200 W and 0.3 m/s, 200W and 0.5m/s, 250W and 0.3m/s, 250W and 0.5m/s, 250W and 0.7m/s, 350W and 0.3m/s, 350W and 0.5m/s, 350W and 0.7m/s, 350W and 0.9m/s, 400W and 0.7m/s, and 400W and 0.9m/s. It should be noted that at intensity setting 350W, the vapor depressions were captured experimentally at five different laser velocity settings, produced more geometries than any other laser intensity. It is possible that 400W also produces this result, but at 400W and 0.4m/s there was no vapor depression detected. Since data was collected depicting a vapor depression at the level below for 0.2m/s and the level above at 0.6m/s for that intensity – it is likely that the lack of a vapor depression at 0.4m/s is the fault of data compromise, possible due to sensor malfunction during that experiment. As discussed in Chapter 4, the convex hull geometry could be more informative than the geometric area in terms

of which vapor depressions are more likely to cause defects. In general, this value is higher than the geometric area – but the pattern shown in the data for geometric area and convex hull area match in terms of increases and decreases over the range of velocity settings. The new generated data by the CGAN are included in bold. Here, the averages of the CGAN produced images were taken for 2,000 images, which therefore matched the number of images used to calculate the average values for each metric from the experimental data.

The new data that was generated fell within the bounds of the experimental data above and below in terms of the laser velocity as expected. The model learned the distribution of features that constituted the vapor depression objects in the images with regards to the input values, or labels, for those velocity settings and therefore was able to reproduce that distribution in the resultant images, even with the new inputs for those new velocities. However, it is worth noting that while 0.5m/s is equidistantly between 0.4m/s and 0.6m/s – the generated images at that velocity (and all of the others) did not produce geometries that were exactly in between the geometries measured at 0.4 and 0.6m/s. This indicated that the model did not simply take an average of the geometries from the data above and below to produce the new images. As the actual experimental data did generally provide a curve-like distribution of areas that increased to a point and then decreased as the velocity increased – these values were not a perfect distribution for those distribution of values. And therefore, the model did not learn to generate the new data along a perfect curve. Rather it appeared to have learned the latent features in the data that influenced the rise and fall in those metrics. This is especially true with the depth measurement, as the experimental data for that metric fluctuated more than the other metrics. Depth can be described as the most dynamically changing property of a vapor depression – likely due to the penetration of the laser into the material directly beneath the surface of the material on the build plate where the laser strikes. Table 10 depicts a

summary of the original data, which gives an average value for all images of vapor depressions taken at each of the 35 experiments – as well as the newly generated data. The metrics depicted are depth, width, geometric area, and convex hull area.

7.13 Generative Output Summary

This work represents an application of advanced generative modeling to a computational materials science problem where the lack of experimental data prohibits the understanding of the LPBF process. By using the images available, extrapolating the latent features and their distributions across the image-based data, these generative models can supplement the experimental data with entirely new synthetic data. While a GAN can reliably deliver new representations of DXR images, the further contribution of the CGAN improves upon the closeness of the artificial data to the original real data. Incorporating additional inputs to the generator in the form of laser velocity and laser power works to incorporate the underlying mechanisms by which those factors influence the build's structures during the build process. These approaches have shown to match the real-world data so closely that another image matching supervised learning model does not differentiate among the real data and the new artificial data. This provides an inexpensive method for developing more data that could then be used for training future predictive models' expectations of microstructure feature characteristics.

Given the results of the CGAN, that model was chosen as the generator for additional data, for image representations of vapor depressions at process parameter combinations what were not experimentally conducted and for which no real data exists. The new data was therefore representative of notional experiments, that is, experiments that were not conducted but for which the generated data could be representative of what the data would have looked like if that experiment actually had occurred. The basis for accepting these new images as valid comes from

TABLE 10
Average Values for Vapor Depression Geometries by CGAN

P	V	DEPTH	WIDTH	G. AREA	C.H. AREA
150	0.2	63.06	38.22	492.46	502.32
150	**0.3**	**25.24**	**36.51**	**510.12**	**452.65**
150	0.4	9.52	25.63	57.49	69.52
150	0.6	0.00	0.00	0.00	0.00
150	0.8	0.00	0.00	0.00	0.00
150	1.0	0.00	0.00	0.00	0.00
150	1.2	0.00	0.00	0.00	0.00
150	1.4	0.00	0.00	0.00	0.00
200	0.2	14.95	42.86	148.38	175.84
200	**0.3**	**30.45**	**45.56**	**210.51**	**203.54**
200	0.4	39.69	32.03	339.50	452.52
200	**0.5**	**41.62**	**48.56**	**581.12**	**628.23**
200	0.6	54.95	68.87	743.68	856.23
200	0.8	0.00	0.00	0.00	0.00
200	1.0	0.00	0.00	0.00	0.00
200	1.2	0.00	0.00	0.00	0.00
200	1.4	0.00	0.00	0.00	0.00
250	0.2	45.65	42.78	475.28	489.51
250	**0.3**	**74.21**	**41.12**	**1023.62**	**995.00**
250	0.4	109.51	77.06	1299.75	1532.23
250	**0.5**	**84.63**	**83.21**	**1002.42**	**892.21**
250	0.6	34.33	81.19	532.50	635.72
250	**0.7**	**12.32**	**32.20**	**201.11**	**451.95**
250	0.8	9.66	26.08	57.56	58.21
250	1.0	0.00	0.00	0.00	0.00
250	1.2	0.00	0.00	0.00	0.00
350	0.2	87.77	54.47	2803.14	2925.91
350	**0.3**	**184.21**	**65.21**	**2754.12**	**2784.78**
350	0.4	202.41	76.58	2410.46	2665.32
350	**0.5**	**124.62**	**94.65**	**2789.21**	**2741.24**
350	0.6	106.44	134.49	2715.03	2987.55
350	**0.7**	**78.1**	**145.15**	**1562.23**	**1842.32**
350	0.8	53.96	112.09	1043.80	1222.54
350	**0.9**	**36.51**	**118.21**	**598.22**	**991.84**
350	1.0	16.13	61.28	185.11	211.74
350	1.2	0.00	0.00	0.00	0.00
350	1.4	0.00	0.00	0.00	0.00
400	0.2	262.73	86.93	4270.17	4676.23
400	0.4	0.00	0.00	0.00	0.00
400	0.6	139.92	119.54	2852.02	2962.46
400	**0.7**	**74.01**	**112.63**	**1908.65**	**2003.82**
400	0.8	42.56	110.81	842.34	952.95
400	**0.9**	**36.87**	**95.52**	**697.65**	**752.62**
400	1.0	16.17	58.32	189.31	195.62
400	1.2	0.00	0.00	0.00	0.00
400	1.4	0.00	0.00	0.00	0.00

the validity of the same model's performance on images for which there could be a quantitative evaluation on the results – the ground truth that exists from the experimental data. And while the new generated images do not have the same ground truth, since there was no experimental build at those combinations of settings, they do follow the same patterns and trends observed when evaluating the real data.

It can also be concluded based on the work in this chapter that the dataset size was adequate enough for the two generative models to train and learn. Limited data is a practical issue in many experimental settings, and in the next chapter, the problem of limited data is explored in more detail. However, for the generation of new artificial DXR-like images for LPBF process parameters of laser intensity and laser velocity – the 2,000 images per each experiment (70,000 in total) was enough. From that data, the model was able to learn the underlying features that determine a vapor depression size and shape, with regards to those parameters of intensity and velocity.

Limitations exist in using generative models to create representations based on desired inputs for classes. The original use of the CGAN was to provide images that pertain to a specific object. The input classes however were not a continuous numerical variable. There were not infinitely many kinds of objects from which the training data supplied to the model. In the case of this work, a continuous variable is being used. While the laser parameters held a limited number of values, it could and should be possible to make predictions for an unlimited number of power inputs. For example, 325W, or 325.6W, or 325.65W, etc. When using this model to make predictions beyond the intervals tested and documented above, the model failed to provide anything meaningful. The MMD evaluation for 350W and 0.3m/s showed a distribution that was statistically insignificantly different from 350W and 0.35m/s. This could be either due to a failure

in the model to learn the intricacies of the features that represent the vapor depression geometries, or that there is no meaningful difference in those expressions. While the differences in vapor depression geometries may be minimal, future materials (besides IN718) could be of interest, where there may be more features in the images for learning.

Better generative output from continuous variables has been explored recently [125]. In that recent work, scalar conditions were used, described as regression labels. That work showed early results, but did have some limitations that prevented adoption for the DXR modeling. The results of using the CcGAN, or Continuous Conditional Generative Adversarial Network failed to converge. This was likely due to the errors bounds on the discriminator losses. Potentially having more data could alleviate this issue, however, as has been noted that is a logistical issue which can not be resolved.

The results from these approaches provided a wealth of new data that can be used to accomplish two further goals. First, the additional data produced to supplement the experimental data can help validate the thermophysics models discussed in Chapter 2. Second, the generation of the images for the new combinations of parameters can be used to determine new geometries. Those can in turn be used in those same models discussed in Chapter 2 to derive expectations on the motion of the melt pool, using the diameters that are obtained from the new images. These help to incorporate the physical properties of melt pool dynamics, the thermodynamics, that govern its motion into the machine learning output that was discussed in this chapter. Therefore, the benefits of these generative models will be of value for thermal modeling of melt pool behavior and solidification.

CHAPTER 8

GAN WITH LIMITED DATA FOR MELT POOL IMAGE SEGMENTATION

8.1 Melt Pool Dynamics

This chapter focuses on applying machine learning to limited datasets. In particular, datasets of the melt pool area, which is a challenging region to capture in image-based data due to the narrow gradient of pixel values encompassing the melt pool compared to the relatively similar pixel values of the surrounding area. Therefore, training data for image based learning is limited, as the DXR process does not capture the melt pool region in dense materials like IN718, or another material of interest to the aerospace community, titanium alloy Ti-6Al-4V. The goal of this chapter is therefore 1) to evaluate the potential to capture this difficult region autonomously from limited data, and 2) to evaluate the ability of the GAN to effectively learn how to model melt pool boundaries – given the accuracy and performance of the GAN in the previous chapter.

For laser powder-bed fusion processes, part of certification involves thermal modeling of the printing process. An accurate model can help to predict thermal conditions within melt pools, and to therefore prevent the flaws caused by temperature gradients such as delamination of layers or cooling based microstructural changes [126]. For these models to be valid, they must be compared to as-printed parts. The simplest test to calibrate is to perform a single-bead, powderless scan on the build-plate and to compare a cross-sectional view of the solidified microstructure to that predicted by the thermal model. The melt pool, the portion of the material at a temperature above the solidus, has a distinct microstructure as compared to the build-plate. This allows a cross section of the microstructure to be compared to the solidus isotherm predicted by the model. For Ti-6Al-4V, a properly calibrated model should predict an isotherm of 1605°C (the solidus temperature) with contours that match the melt pool of the printed single-track [127]. However, as

discussed previously, the shape of the melt pool is a dynamic phenomenon that is little understood despite a multitude of ideas [128]. A single micrograph is therefore not enough to say a model is properly calibrated; multiple micrographs of the same laser power and velocity parameters are needed to have a statistical representation of melt pool qualities. There is need for a way to automate the extraction of melt pool contours from cross-sections of single-track LPBF samples.

Computer vision applications offer a solution, giving a high-throughput method to extract melt pool contours, width, and depth. The process of classifying each pixel within an image—here, as either part of the melt pool or background— is considered as an image semantic segmentation problem. The difficulty is that the algorithm often needs a large set of data to train a model. The expense of performing experiments in many fields of materials science, not just AM, limits the data available to train these models. There is therefore motivation to evaluate different algorithms to determine which can perform best on a limited data set.

The goal of this chapter was to develop a machine learning method to capture the melt pool boundary, leveraging the limited data available from micrographs taken post-processing – that is, after the LPBF build has completed. To accomplish this, a GAN was developed in order to test its ability to generate new images based on a limited training set. Given the limited amount of data available, evaluation of the model including also using a particular convolutional neural network, CNN, that has shown results in learning object features from small datasets [129]. Convolutional neural networks, specifically using the U-Net architecture are often utilized for semantic segmentation tasks [130]. Meanwhile GAN provides an alternative to the pixel-level classification technique of the U-Net. Therefore, using this model architecture here presents a contrast to the GAN model, while also potentially accomplishing the goal for melt pool identification.

8.2 Micrograph Dataset

The dataset consisted of optical micrographs of cross-sectioned single scan tracks, manufactured by laser powder-bed fusion on a bare Ti6-4Al-V substrate. The resultant build was then cross-sectioned for evaluation under light microscopy, from which images were taken of the build area. The dataset was comprised of 57 images from two sources, one with 42 color images and one with 15 grayscale images. Within the combined set, 8 images depicted keyholing, 7 lack-of-fusion, and 4 balling with the remainder being in the conduction regime. Masks, binary images denoting the melt pool region, were created in Adobe Photoshop so that the melt pool was filled with white pixels with all else black. Given the small size of the dataset, a leave-one-out validation approach was taken, with 12 images for the test set, and 36 for the training set.

8.3 Pix2Pix and U-Net for Image Translation

The GAN utilized in this chapter is based off of the Pix2Pix GAN originally developed [131]. This was developed to perform style transfer of images, such as changing sketch into a realistic image of a purse or inputting a photograph captured during the day and generating a view of the scene at night. This capability made the pix2pix network particularly well suited for the problem at hand because image translation requires the model to thoroughly learn object features, in order to properly reproduce new images. In this case, the object of interest is the melt pool and its surrounding boundary. Unlike simpler implementations of GANs, this model uses convolutional layers for the generator and the discriminator networks – thereby utilizing batch normalization within the hidden layers.

The generator model takes in images as inputs, but uses a different source of randomness. Instead of sampling from a point in a latent space, the randomness comes from having dropout layers that occurs during training. Overall, the model architecture involves having the generator

taking in an image as an input and down-sampling it over several layers (similar to the variational autoencoder mechanism), until it reaches a bottleneck layer, where the representation is now up-sampled over and over for several layers until an output image is produced [131]. The utilization of this mechanism is very similar to the U-Net described later, with the exception that the GAN output product is different from the image classification of the U-Net.

Meanwhile, the discriminator network, working in tandem, takes in an image from the training data as well as an image produced from the generator to predict the likelihood that the generated sample came from the training dataset – much like the traditional GAN scheme. The difference here is that the Pix2Pix uses a unique network architecture to accomplish this, known as PatchGAN [132]. This is a CNN that classifies images based on looking at portions of an image, or patches rather than the entire image. In this fashion, the model classifies if those individual patches are real or fake, rather than that overall image. The output then becomes a single feature map of what has been determined to be real versus fake predictions which can be averaged to give a single score [133].

Additional modifications to the traditional GAN include slowing down the discriminator's learning process, as the above described network architecture allows the discriminator to learn much faster than the generator. The generator's training occurs using both the adversarial loss for the discriminator model and the mean absolute pixel difference between the generated image translation and the expectation from the training image. These losses are then combined into what is known as a composite loss function – which constantly updates the generator model [133]. While the adversarial loss determines whether the generator has successfully produced a valid image based on the training data, the mean absolute difference loss helps the generator to create images that are passable as new translations of the training data image, thereby accomplishing the picture

to picture objective referenced in the name of the Pix2Pix model name. The two types of losses together can be controlled by a hyperparameter lambda value. For instance, when set to 10, that indicates that the mean absolute difference is 10 times more important to the model than the adversarial loss to the generator [132].

The Pix2Pix model framework improves upon the general GAN model by using a conditional-image input, which provides larger output images compared to other GAN models. In addition to training images, these images are also paired with labels, whereby the model learns a label-pair to improve the image translation capabilities. For this work, the model was modified to have a batch size of 6 for training, a batch size of 9 for validation, and to train on single channel images rather than 3 channel color images in addition to the preprocessing steps mentioned below.

The U-Net approach was originally developed by Ronneberger et al. for use in biomedical image segmentation [134]. It is a typical CNN, or deep neural network, albeit with a modification. It also allows down-sampling to a bottleneck layer, then up-sampling to the output image. However, during the up-sampling layers, there are a large amount of feature channels, which allows the model to pass on locally important information, or contextual information, as it goes from larger to larger resolution layers [135]. The down-sampling operations are convolutional layers, with each layer followed by a ReLU and a max pooling operation. The objective is to reduce the spatial information from the image while maximizing the feature information picked up from the image through those layers. The symmetry between the down-sampling layers and the up-sampling layers gives the network architecture a u-like shape, hence the name U-Net.

For this work, notable deviations from the source include using 16 filters instead of 64, changing hyperparameters such as learning rate, adjusting the early stopping epoch, and adding the preprocessing steps that follow. The preprocessing steps of the two methods differed slightly

as each was chosen to maximize the performance of the particular approach. The GAN approach was processed with contrast limited adaptive histogram equalization (CLAHE). The U-Net approach performed global histogram equalization before performing CLAHE. Both models converted images to 256 x 256 and a single channel (black and white), and all training images were horizontally flipped, doubling the size of the training data.

The models were evaluated on the test set and compared by two metrics: Intersection over Union, (IoU) and Hausdorff distance (HD). Many semantic segmentation algorithms use metrics that evaluate the area of pixels correctly identified in similar fashion to IoU. While useful for model performance during training, IoU was not the best metric for evaluating performance after training because each pixel is given equal weight. IoU is best described as an area metric, but a more accurate way would be to utilize an edge metric. The desired outcome of this work was to have accurate representations of the melt pool contours, and a metric was needed to evaluate the error of edge locations, hence the inclusion of the HD evaluation.

8.4 Pix2Pix and U-Net Performance Evaluation

As seen in Fig. , there are two optical micrographs from the test set (Image 2 and Image 6), their respective ground-truth masks, and the comparison mask of both models. Each comparison mask is comprised of the generated mask from a single model overlaid onto the ground-truth mask of the same image. Purple areas denote pixels of intersection, whereas blue denotes pixels where only the ground-truth mask was present, and red denotes pixels where only the generated mask was present. The orange and blue dots correspond to the end points of the HD, with orange denoting the point on the ground-truth mask and blue denoting the point on the predicted mask. For Image 2 (Fig. 38a-d), both algorithms predicted a melt pool that had large overlap with the ground-truth mask. The generated areas matched well with the ground-truth, although neither

algorithm predicted the top left ‘corner’. For Image 6, a melt pool exhibiting keyholing, neither model generated a mask that visually matched the ground-truth mask. This is likely due to the training set only containing one keyhole melt pool. While the melt pool predicted by the GAN (Fig. h) showed a better match to the ground-truth mask than the U-Net. The U-Net falsely predicted a region separate from the melt pool; these predicted areas distinct from the true melt pool are referred to as artifacts within this discussion. The GAN did not predict this false positive, though it did predict an excess on the left side of the melt pool, and both models predicted a filled region rather than a cleft in top center. Melt pools in the conduction regime were well-represented in the training set, and most masks generated from those melt pools closely matched the ground-truth masks, but due to the lack of training data, the masks generated from melt pools outside the conduction regime did not match as well visually.

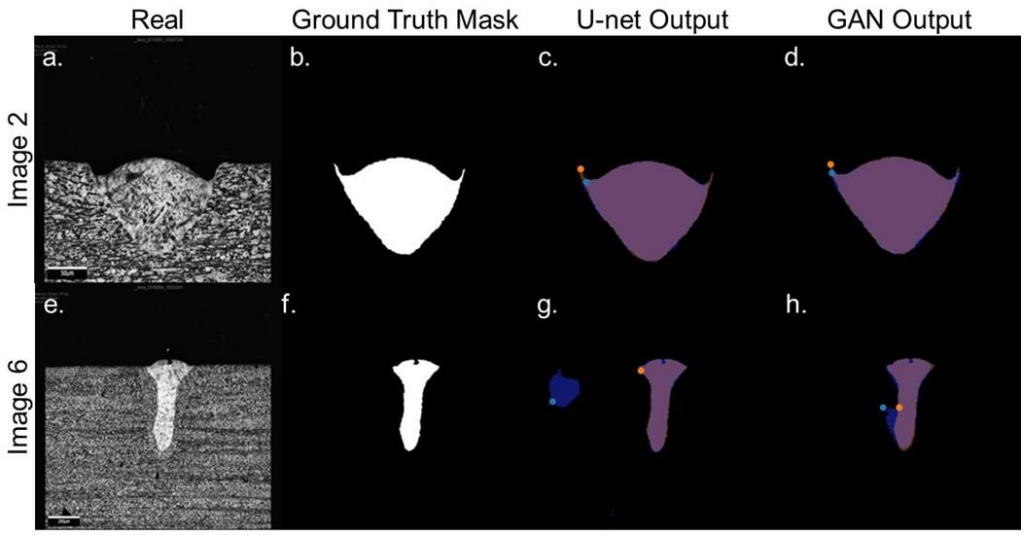

Fig. 38. Qualitative comparison of two melt pool types: within the conduction regime (a-d) and keyholing (e-h). Parts a and e are the input micrographs, image 2 and image 6 respectively. Parts b and f are the ground-truth masks. Parts c and g are comparison masks from the U-Net algorithm. Parts d and h are comparison masks from the GAN algorithm. Orange points denote the Hausdorff point on the ground-truth masks; blue points denote the Hausdorff point on the generated masks.

Predictions of the two models are quantitatively compared by IoU and Hausdorff distance in Table , with the metrics for each image within the test set shown in Fig. 39. Though masks generated by models were visually similar, the U-Net approach performed better according to both IoU (the area metric) and Hausdorff distance (the edge metric). Masks generated by the U-Net resulted in IoU values greater than 90% for 9 out of 12 test images. For masks generated by the GAN, only 6 out of the 12 produced IoU values above 90%. Furthermore, all predictions by the U-Net resulted in an IoU over 84%, and masks generated by the U-Net outperformed those generated by the GAN in 9 out of 12 images by both IoU and Hausdorff distance.

Table 11
Average Metrics with 95% Confidence Interval as Evaluated on All Test Set Images

	IOU	**HD**
GAN	89 ± 5.1	32 ± 23
U-Net	93 ± 3.2	16 ± 15

The U-Net outperformed the GAN for segmentation of single-track melt pool images in most cases. This is not unexpected: in the case of segmenting text from scanned documents, the two methods have been compared showing the U-Net approach out-performing the GAN in that case as well [136]. While the two cases have notable differences— the use of error metrics specific to text segmentation, the availability of a larger dataset (136 images), and a different implementation of the U-Net— the U-Net performed better than the GAN by most metrics [136]. This too is not unexpected: the U-Net approach was developed specifically for pixel-by-pixel

classification [134], but the Pix2Pix GAN was developed for the broader application of style transfer, though it has been utilized to produce segmentation masks [137].

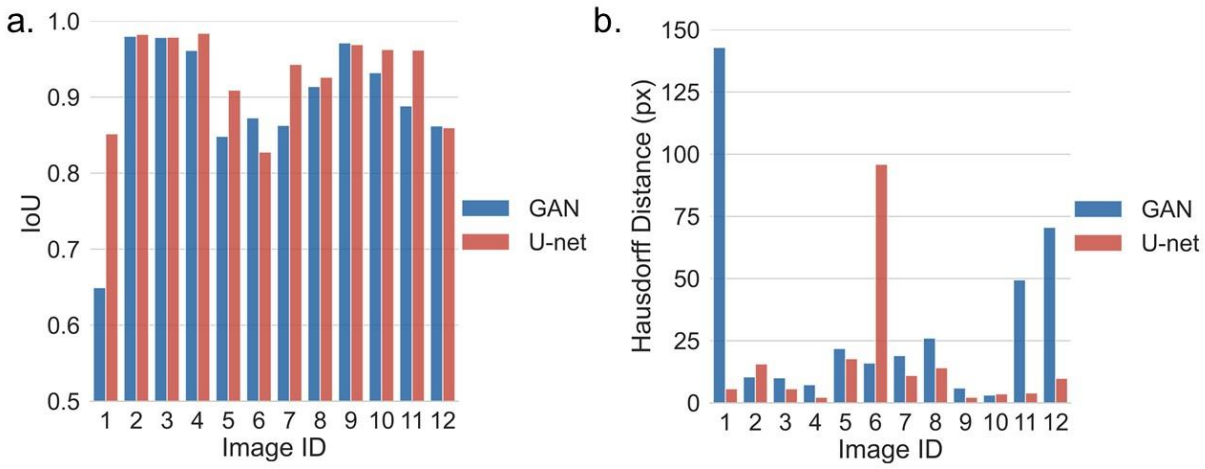

Fig. 39. Quantitative metrics on individual test images. a) IoU b) HD in pixels

The exceptions for this case, when evaluating with the IoU, is image 6 as seen in Figure 39a above. Here, the IoU was slightly higher for the U-Net result than the GAN result. An inspection of this image does show more overlap in the GAN output compared to the U-Net output. When comparing the results for the HD, the GAN result generally had a higher score, with the same exception, image 6. Here, the U-Net again scored higher. This is due to this particular image having a wider boundary around the real image when overlayed with the generated image from the U-Net. While the U-Net was usually able to create a better approximation of the image boundary, for image 6 this was not the case. Nevertheless, it was generally a more reliable model output as seen from the remaining image tests.

The GAN, while a powerful tool, is of greater complexity than is needed for this application, and that complexity creates difficulties for the training process. Not only was training of the GAN slower than the U-Net but obtaining a converged model with the former was non-trivial. A GAN produces its most realistic images when both the generator and the discriminator updates change little between each update in training weights. This problem is inherent to the two network architectures of the GAN because updating the weights of one could reverse an improvement in the other [138]. While the GAN could produce reasonable results without convergence, achieving consistent convergence was the most reliable way to obtain quality generated masks. However, this stability was hard to achieve with the small training set available. The larger batch size of 6 (as compared to 1 in the source) helped to achieve convergence with the present dataset, but bigger batch sizes are not guaranteed to improve convergence.

With the U-Net approach identified as the preferred approach for the objective described at the opening of this chapter, the remainder of the discussion will be on use and benefits of the HD, as this metric has not previously been used to evaluate additive manufacturing computer vision models in literature. Fig. 40 plots the IoU value against the Hausdorff distance, one point for each generated mask for both models. Though there is correlation of a decrease in HD with an increasing IoU value, a linear regression of all points resulted in an R^2 value of only 0.68, supporting the use of HD due to the inherent differences between the two. Because of the high area to edge ratio of these melt pools, high IoU values can be achieved despite edges not matching closely with the ground-truth and vice-versa. The best example of this is the mask of image 1 generated by the U-Net which had an IOU of 85% and a Hausdorff distance of 11 pixels.

Hausdorff distance also provides unique advantages unavailable with IoU. Because HD is a maximum, it only gives information about the worst mismatch of the generated mask,

irrespective of how other points in the generated mask relate to the ground-truth. For an end-user extracting melt pool contours, this can help to flag outlier images. HD is beneficial here as the extracted contour is only as useful to thermal model validation as its least accurate part. In Fig. 38b, any generated mask that had HD above 30 pixels was seen to have an artifact; those below 30 pixels correctly predicted one melt pool per image. For a small data set, the HD could be utilized to identify trends that will allow evaluators to build a more robust training set.

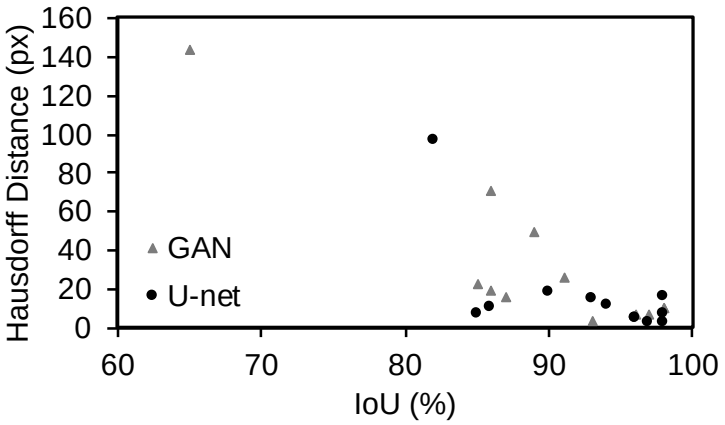

Fig. 40. Correlation of HD to IoU

Utilizing a metric based on a maximum has inherent disadvantages, however. When a model was performing poorly, for example including multiple artifacts, the HD would give little indication whether the generated image was completely inaccurate or just had a single artifact. The metric's sensitivity to artifacting also limits use as a loss function during training.

8.5 Summary of Pix2Pix Analysis

For the single-track melt pool micrographs of additively manufactured Ti-6Al-4V, a GAN approach was proven to be able to segment melt pools from optical micrographs. However, the U-Net algorithm (a pixel-wise approach) generally outperformed a GAN (a generative approach) in both IoU and Hausdorff distance. The results of these tests for image 6 was seen as an exception, and can be attributed to the U-Net segmentation not completely learning an indentation in the real image and therefore creating a slightly wider outer edge for the object than there should have been. Hausdorff distance was introduced to evaluate the accuracy of generated melt pool boundaries specifically due to interpretability, sensitivity to the largest error, and being an edge metric. However, Hausdorff distance should be utilized in tandem with IoU. For the small dataset of 57 images, the U-Net approach was able to achieve an IoU of over 80% for all test images and over 90% for 9 out of 12 test images. The GAN achieved an IoU of over 80% 11 out of 12 test images and over 90% for 6 out of 12 test images.

While the previous chapter demonstrated the ability of the GAN and the CGAN to make valid new images, this chapter discusses some limitations. Those limitations are primarily due to the limited amount of data presented here. The task of identification of melt pool boundaries is a significant problem, and having a valid way to accomplish that is highly desirable for process certification endeavors. Nevertheless, while the GAN was unable to learn from this small dataset to reproduce images with a reliable melt pool boundary – the CNN U-Net did show more promise in accomplishing this task for classification purposes.

CHAPTER 9

IMAGE SYNTHESIS USING VAE

9.1 Variational Autoencoder Implementation

A variational autoencoder (VAE) model was developed and implemented based on the literature that suggested the autoencoder model is a capable generative model in many circumstances, discussed in section 2.10. The fundamental mechanisms that govern the process of sampling from the latent space and thereby collecting a new representation of data, in this case, image data was discussed in that section. Of particular interest is the statistical underpinning of this approach, where the VAE is essentially a deep learning based probabilistic model, or deep Gaussian model. For instance, in a statistical Gaussian model, it can generally be assumed that there is a distribution of observed variables which are associated with corresponding latent variables. And those latent variables are drawn from a prior density $p(z)$ and are associated with the data observations based on a likelihood, which can be expressed as $p_\theta(x|z)$. This is the case for a type of deep learning class of models known as Deep Latent Gaussian models (DLGMs) – where the observed variable is governed by a *hierarchy* of latent variables, and the latent variables at each level of the hierarchy are Gaussian *a priori* [139]. Overall, this implies that the VAE is not explicitly a generative model per se, rather it is a DLGM that has generative abilities as a consequence. This stands in contrast to the GAN whose output is a generated observation.

In this work, sampling from the latent space encoded by the VAE was used to generate new examples of vapor depression images, similar to the goal of Chapter 7 with the GANs. The training images again consisted of 2,000 images based on experimental combinations of laser velocities (0.2, 0.4, 0.6, 0.8, 1.0, 1.2, and 1.4 meters per second) and laser intensities (150, 200, 250, 300, 350, and 400 Watts). The VAE model was initially trained with each intensity set of

experiments separately. That is, the first VAE was run with all images from 150W – which included all five velocity settings for a total of 10,000 images. A second model was trained on only 200W data, then 250W, and so on. The idea was to test the model's ability to learn the distribution of features across each velocity first, and evaluate the results. Should those results be favorable, then a new model trained on all images together would be used.

The model was again created in Python using Keras. The training images were all initially resized into 28x28 images, then flattened to be a vector of size 784. A description of the steps developed in the code for developing and running the VAE model are as follows:

- Creation of an input layer
- Creation of intermediate layer
- Application of ReLU activation function
- Development and definition of latent feature space
- Definition of the log variance of the latent space
- Deployment of sampling function to learn mean and variance
- Linkage of the input to decoder from latent space
- Map latent space to intermediate dimension
- Define loss function from binary cross entropy and KL divergence
- Sample from latent space to obtain new image representation

For the decoder, the input comes from sampling from the latent space, based on learning the mean and the variance of the distribution. Then, at the image output or generation step, a sampling function based on learning those values is fed through the decoder to get to the model output. Unlike the GAN where there is a loss function for the discriminator and the generator as each network functions separately, the loss function for the VAE works differently as the network

architecture is structured differently. Here there is one loss function, although it is a combination of two factors: binary cross entropy and the KL divergence (discussed in section 2.10). Essentially, there is a comparison from each image in greyscale for the pixel value to the value encoded and reconstructed. The model was optimized using the Adam optimization algorithm.

9.2 Effectiveness of VAE Generated Images

At a visual inspection, the images produced did not show variation among the laser velocities, as would be expected. The images in general had very little difference, despite the training images showing clear variations among the different velocities. This is noteworthy as this particular model does not learn the labels for the images. In an ideal scenario, the model would have learned how the features changed across each class of image, without needing to know the label. Fig. 41 shows an example of images from the model – with the first row of images being actual images for each velocity at 350W, and the second row showing images taken from the VAE for each velocity at 350W. These results show how the model in general produced vapor depression images that were roughly the same, which was not an ideal outcome.

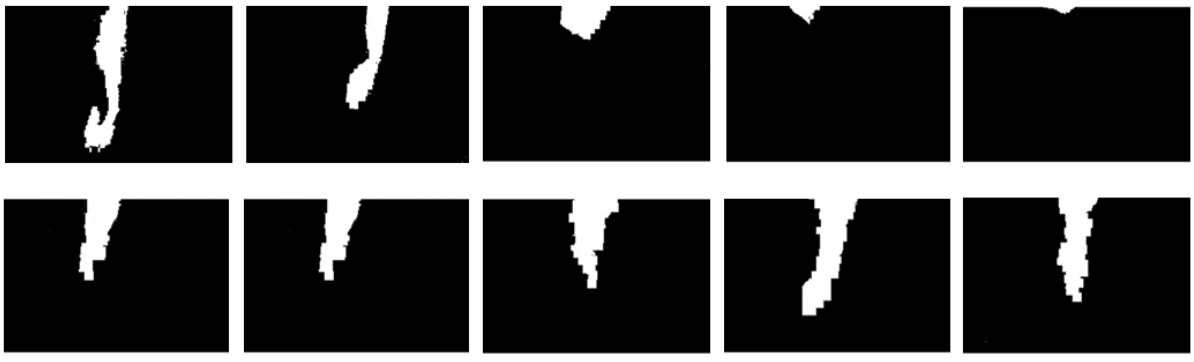

Fig. 41. Images from real dataset at 350W (top) at 0.2, 0.4, 0.6, 0.8, and 1.0 m/s, with images generated by VAE (bottom) from that training data

A quantitative test was performed to look at the average area of the images produced compared to the average area from the real examples. It was shown that the average area was not comparable to the actual average area – even when each class of velocity images were compared individually (see Table 12). No class of images (0.2, 0.4, 0.6, 0.8, or 1.0 m/s) had an average area that was statistically similar to the average area produced by the VAE. Additionally, as the images shown in Fig. 41 make clear, the VAE output were all similar. The variance across the images from the VAE were much smaller than the overall variance for the real dataset. The model had poor performance from a basic evaluation.

Upon visual inspection, the output images were rejected based on the closeness of the similarity to one another, and the lack of closeness to any of the training images. Therefore, additional testing on the effectiveness of these images were not necessary as they could clearly not be used for quantification of vapor depression geometries or validation of the thermophysics of the LPBF process.

The results here suggest that the VAE was unable to perform as a generative model. The likely reason for this is the mechanism by which the VAE samples from its latent space. The VAE uses a Gaussian distribution to map the features from the images upon which it has been trained. By definition of a Gaussian distribution, 99.7% of the probability distribution will collected within three standard deviations from the mean. Likewise, when sampling, the VAE will tend to sample from that middle area, or closer to the top of the Gaussian curve – which can be thought of as a safe middle ground upon which to sample [111]. This leaves out the complexities of the features from the images when selecting, thereby generating images that look similar in this case.

TABLE 12
Average Area of Real Images Compared to VAE Images at 350W

Velocity	**Real**	**VAE**
0.2 m/s	2803.14	2945.51
0.4 m/s	2410.46	3025.36
0.6 m/s	2715.03	2935.87
0.8 m/s	1043.80	2889.16
1.0 m/s	185.110	2976.94

9.3 Conditional VAE Implementation

The VAE has a limitation in terms of image generation not unlike the GAN in that it is difficult to generate a specific image from a specific class when there are many classes. In the case of this work, there were five classes: the five different velocity settings. The general VAE model has the encoder *Q(z|X)* and the decoder *P(X|z)*, where the encoder models the latent variable *z* based on *X*, regardless of any class of *X* [111]. Likewise, the decoder models *X* based on the latent variable *z*, regardless of any class or label. As this produces images that may not exactly correspond to a particular class, a modification to the overall scheme can be made where both the encoder and the decoder are conditioned to a new variable, *c*. For the encoder, the model is now *Q(z|X,c)* and for the decoder it is now *P(X,c|z)*. The distribution of data then becomes conditioned based on the overall lower bound of the variational objective which has then become:

$$logP(X|c) - D_{KL}[Q(z|X,c)||P(z|X,c)] = E[logP(X|z,c)] - D_{KL}[Q(z|X,C)||P(z|c)] \quad (46)$$

where the latent variable is now distributed by *P(z|c)* and for every possible value of *c*, there is a *P(z)* [140].

Modifications to the VAE previously discussed included incorporating the conditional variable *c*. In this case, this conditional variable represented the class labels for the velocity

variable, which was a categorical input. It therefore had to be one-hot encoded for the model to receive as an input. This new vector was then concatenated with the encoder and decoder network. From there, the new CVAE model was run and images from the class of interest was then extracted by using the corresponding one-hot encoded value.

9.4 Conditional VAE Results

The CVAE model produced images that at a visual inspection were an improvement over the general VAE performance. The vapor depression sizes in the images appear to shift in size appropriately, following the trends seen in the real data (Fig. 42). However, while the images appear to be an improvement over those from the VAE, they did not match up with the real data enough to be considered interchangeable with the images from the real set. As seen in Fig. 42, the object shrinks as the velocity increases, from 0.2 to 1.0 m/s – as seen in the real data. However, those decreases are not as dramatic as they should be. This indicates that the features encoded in the latent space are still not representative to the impact that the velocity has on the geometries.

Fig. 42. Images from real dataset at 350W (top) at 0.2, 0.4, 0.6, 0.8, and 1.0 m/s, with images generated by CVAE (bottom) from each class

A quantitative analysis of the CVAE results included measuring samples from the generated data with the Intersection over Union (IoU) and Hausdorff distance (HD), the background of which were described in Sections 7.6 and 7.7 respectively. Table 13 shows the results of these measurements, with values that were significantly above the values that would be expected, and higher than those produced by the CGAN in Chapter 7. The values indicate the relatively poor job the CVAE did in creating new images, in that they did not match well with the original data in both cases.

TABLE 13
IoU and HD of CVAE Generated Images

Velocity	**IOU**	**HD**
0.2 m/s	0.7013	205
0.4 m/s	0.6203	378
0.6 m/s	0.3127	951
0.8 m/s	0.1798	1350
1.0 m/s	0.0210	2034

The addition of the input label did help with the goal of producing new images, but the output was not comparable to the generative capabilities of the GAN models. Given these results, it was determined that attempting to generate new images to represent new velocity parameters should not be performed, as the images produced for those known velocity parameters failed to adequately capture the real area distributions. Future work in the area of autoencoders could include using the results from the CGAN to supplement the image datasets. These additional data could potentially include the VAE training, thereby helping to better map the features from the objects in the latent space. Some success has been found in merging the two models together, the

GAN and the VAE, known as the Adversarial Generative Autoencoder, or VAEGAN [141]. However, these models would still be limited in their ability to produce specific images for a particular label, as there is not a conditioning aspect to the model training and image generation. The results here, as well as from Chapter 7 demonstrate that model conditioning is necessary for the successful creation of new images for specific combinations of process parameter variables.

9.5 VAE results with RMSE

The RMSE and SI evaluation metrics were applied to the VAE and CVAE results, in a similar fashion to how those were used to evaluate the uncertainty of the GAN and CGAN model predictions. The results for the VAE model are shown in Table 14 and the results for the CVAE model are shown in Table 15. For the VAE and CVAE models, a similar result was found where the CVAE model outperformed the VAE model, just like how the CGAN outperformed the GAN.

TABLE 14
VAE Evaluation with RMSE and SI Metrics

Test	**Area**		**C.H.**		**Depth**		**Width**	
Parameter	**RMSE**	**SI**	**RMSE**	**SI**	**RMSE**	**SI**	**RMSE**	**SI**
0.2m/s	562.64	78.91	754.85	101.6	133.8	105.3	183.6	98.65
0.4m/s	542.61	132.2	964.68	285.6	167.3	206.6	128.7	174.6
0.6m/s	781.21	145.6	930.85	245.6	158.2	265.3	97.16	151.4
0.8m/s	894.12	354.8	996.72	349.5	118.0	378.6	175.9	453.6
1.0m/s	1238.7	513.6	689.50	456.6	96.96	456.9	135.0	512.5

TABLE 15
CVAE Evaluation with RMSE and SI Metrics

Test Parameter	Area RMSE	Area SI	C.H. RMSE	C.H. SI	Depth RMSE	Depth SI	Width RMSE	Width SI
0.2m/s	469.12	73.43	573.28	75.51	123.59	43.03	37.68	48.82
0.4m/s	446.57	90.15	552.43	107.0	107.52	107.7	79.06	79.56
0.6m/s	693.88	61.35	590.95	36.12	136.28	46.69	85.84	86.58
0.8m/s	814.64	147.6	912.86	231.3	231.50	232.1	167.29	167.4
1.0m/s	1149.2	431.2	535.11	415.4	415.98	416.1	109.82	110.7

Overall, these results further emphasize how the autoencoder learning mechanism does not compete in this problem set for delivering generated images that can be relied upon, to supplement the experimental data. While the GAN model discussed in Chapter 7 was inferior to the CGAN model discussed in that same chapter, even that lower performing GAN was superior to the better of the two autoencoder models, the CVAE. This was consistent across all evaluation metrics discussed in this chapter. Deep learning has provided a method for generating new data – several in fact, as described here in Chapter 9 and previously in Chapter 7. The computational learning mechanism involved with multiple hidden layers in the deep learning paradigm, along with the convolution and other operations, has shown that it is possible to artificially create new data. While some methods work better than others, it has been shown that there is at least more than one way to approach this problem of limited data. As both of these deep learning methods were image based, the next chapter provides an approach to non-image based generative modeling. In that chapter the recommender system will be used to generate, or recommend, new geometries, which can then be evaluated for its accuracy and compared to the aforementioned image based approaches.

CHAPTER 10

RECOMMENDER SYSTEMS FOR MICROSTRUCTURE GENERATION

10.1 Alternating Least Squares

Recommender systems provide a non-deep learning approach to modeling the DXR data and making predictions on future potential geometries. While not typically considered a generative model – as discussed in Section 2.9, recommender systems when using the underlying matrix completion approaches, can be used to make predictions based on unknown factors. For instance, when a new user enters an e-commerce environment, and the system strives to make predictions, or recommendations, on what that new user may favor. Using this approach, recommender systems can be structured such that the user is a particular process parameter, and the recommendation is the geometry measurement for that "user", thereby delivering a new measurement despite there being no training data available for that "user". Section 2.9 reviewed one of the most commonly used recommender systems, based on the Alternating Least Squares (ALS) algorithm, which is explored here.

The task of generating or approximating missing data can be accomplished through matrix completion. Oftentimes these computational techniques are incorporated with various additional algorithms to leverage the results to make predictions, or recommendations, for an output given a system of inputs. Known as recommender or recommendation systems, these approaches have obvious benefits to the retail industry, but can also be applied to many other domains as well [37]. Nevertheless, one of the most common use cases involves making movie recommendations for users of an online streaming service, Netflix.com. This example is due to the advancement of the recommender system inspired in part by the Netflix challenge where the company sought help from researchers to develop and apply a more optimal system for making movie suggestions to its

users [142]. Section 2.9 discussed the theory behind the ALS approach, which was employed to make predictions on the following characteristics: area, convex hull area, depth, and width of the vapor depression.

The data was prepared by forming into an *m* x *n* matrix where all data from all 35 experiments were collated into a single dataset. During the data collection process, 35 experiments were conducted, where 2,000 images were taken from each. Using the previously described image contouring approach, and the measurements of area, depth, width, and the convex hull area for each frame – a full dataset was constructed using the aggregated measurements of all experimental runs. A matrix was then created where the velocity of the laser at each time interval populated columns and the laser intensity setting for each time interval represented the rows. The values populating the matrix were the area that was measured for each combination of laser velocity and intensity – for every time interval. Therefore, the final matrix contained the measured area for every image that was experimentally collected, in a 70,000 by 70,000 matrix. However, as there was no measured activity for experiments where the laser velocity was set to 1.2 and 1.4 meters per second, data for those experiments were excluded. This resulted in a final matrix of 50,000 by 50,000. This procedure was then repeated to create matrices for vapor depression depths, widths, areas, and convex hull areas. Similar to all previous work, all steps and calculations were conducted using Python, with Numpy.

The ALS model works by solving for the user vector and the item vector. For this work, the user vector, x_u, is the parameter of the laser intensity. Meanwhile, the item vector, y_i is the second parameter, the laser velocity. As discussed in Section 2.9, the recovered matrix R_{ui} is difficult to fully compute as the objective function in non-convex. However, the ALS approach works by fixing the variables as constants and then solving for one, with respect to the other. The

problem then becomes a minimization problem, to minimize the now convex function of y_i. Holding this item vector constant while taking the derivative of the loss function with respect to the user vector results in this expression:

$$\frac{\partial L}{\partial x_u} = -2\sum_i(r_{ui} - x_u^T * y_i) y_i^T + 2 \lambda_x x_u^T \quad (47)$$

which reduces to:

$$0 = -(r_u - x_u^T Y^T)Y + \lambda_x x_u^T$$

followed by:

$$x_u^T (Y^T Y + \lambda_x I) = r_u Y$$

and finally,

$$x_u^T = r_u Y (Y^T Y + \lambda_x I)^{-1} \quad (48)$$

This is then repeated, except this time for the item vectors:

$$\frac{\partial L}{\partial y_i} = -2\sum_i(r_{ui} - y_i^T * x_u) x_u^T + 2 \lambda_x y_i^T$$

$$0 = -(r_i - y_i^T X^T)X + \lambda_x y_i^T$$

$$y_i^T (X^T X + \lambda_x I) = r_i X$$

$$y_i^T = r_i X(X^T X + \lambda_x I)^{-1} \quad (49)$$

To test the ability of this approach to generate new measurements of vapor depression area (followed by convex hull area, depth, and width separately), the dataset was prepared such that all measured values for columns with velocity at 0.2m/s were removed. The ALS approach then attempted to calculate the values that would fall into a column for 0.2m/s, which was then repeated 2,000 times so that there would now be a number of predictions equal to the number of observations in the original dataset – for each combination of 0.2m/s and each power setting of 150W, 200W, 250W, 350W and 450W. This was repeated for each velocity setting. The output

was a set of predictions for vapor depression areas for a chosen velocity setting. These predictions could then be compared against the original data, which was held out from the model.

10.2 ALS results with RMSE

Similar to previous chapters for evaluating GAN and VAE model performance, testing of the ALS approach used the root mean squared error (RMSE). This was chosen as a metric for evaluation as it provides a quantitative measure of the difference between the predicted values and the real values, based on absolute difference. The results from these measures are presented in Table 16. Here, the resultant predictions from removing all data for experiments run at 0.2m/s (and all other velocities subsequently) are shown. Predictions for each geometric characteristic is then presented, for which the model was run separately for each. Beginning with the area matrix, this model treated the power setting as the user and velocity as the item. For instance, for user 350W, the system is making recommendations on an area measurement for item 0.2m/s. Initially, the system has information on that user, 350W in the sense of other items it is associated with, such as 0.4, 0.6, 0.8, and 1.0 m/s. In this sense, the new item, 0.2m/s, is roughly the same as a new item added to an online marketplace, upon which a recommender system makes predictions for users on whether or not they will want that new item based on ranks given to other items. Therefore, this represents the cold-start problem, where a new item is introduced and the system must make a recommendation for it.

As seen in Table 16, the RMSE and subsequently the SI are higher for area and convex hull area, than they are for depth and width. The smaller range in those values helped the recommender system to better learn their distribution. Also noteworthy is the quality of the predictions were higher for the lower velocities of 0.2, 0.4, and 0.6 m/s. This is due to the volume of data available at those velocity settings, whereas for 0.8 and 1.0 m/s there were runs where no

vapor depression was detected. Overall, the results show that the ALS approach was able to make recommendations on area measurements for the process parameter combinations, yet the performance for many of these is still lacking. This is likely due to the lack of "users" for the system, having only five. As discussed previously, collaborative filtering scenarios expect many users in order to properly gauge behavior. Due to the overall high RMSE and SI values, an additional attempt at another recommender system was considered, developed, and discussed in the next section.

TABLE 16
ALS Results of Parameter Tested by Removal from Dataset

Test Parameter	Area		C.H.		Depth		Width	
	RMSE	SI	RMSE	SI	RMSE	SI	RMSE	SI
0.2m/s	397.33	30.15	456.7	32.78	37.53	39.58	19.92	37.55
0.4m/s	430.18	52.37	469.7	52.62	45.88	63.52	23.64	55.93
0.6m/s	419.57	30.66	374.6	23.56	41.91	62.43	51.28	63.45
0.8m/s	506.37	130.3	842.7	167.7	49.42	232.7	70.41	141.4
1.0m/s	297.18	396.9	287.0	362.6	88.83	335.7	35.49	148.3

10.3 Accelerated Proximal Gradient

The above results of the ALS approach suggested than the data may have not provided enough information to the model to make accurate predictions. For instance, data at 400W did have signal noise present as there were zero readings for those experiments, likely due to sensor or equipment failure. Thus, an alternate approach was explored and performed.

In applications for real world data, approaches for matrix completion can be pursued whereby instead of applying regularization on a decomposed matrix [143], we instead apply

regularization on the nuclear norm (also known as the trace norm) of the recovered matrix *R*. In this fashion we focus on that nuclear norm, which is the sum of the singular values in matrix *R*. The goal of this regularization approach is to find a solution that effectively balances the minimization of the approximation of the error in the known entries as well as the nuclear norm of matrix *R*, such as

$$\min_{R} \frac{1}{2} \|P_{\Lambda}(R) - P_{\Lambda}(A)\|_F^2 + \lambda_1 \|R\|_*, \tag{50}$$

where λ_1 is the regularization parameter controlling the extent of the nuclear norm. It is important to note that this is a convex model for completing matrix A.

In the low-dimensional state, and therefore as a low-rank matrix, the matrix completion problem can be formulated as a matrix rank optimization problem such that

$$\min_{R} Rank(R), s.t.$$

$$R_{ui} = A_{ui}, (u,i) \in \Lambda. \tag{51}$$

where *Rank(R)* denotes the rank of matrix *R*. While it is not feasible to find the exact solution of the recovered matrix as this problem is known to be NP-hard [44], there are ways to leverage the low-rank matrix approximation to yield results that can come close to an optimal solution, which is the general goal of several computational algorithms (which can then even be utilized by recommender systems as described previously). For instance, the rank optimization problem can be reconstrued as a nuclear norm optimization problem [46] by working to minimize the sum of the singular values in the recovered matrix *R*. This can be demonstrated as

$$\min_{R} \|R\|_*,$$

$$s.t.\ R_{ui} = A_{ui}, (u,i) \in \Lambda. \tag{52}$$

Where $\|.\|_*$ denotes the nuclear norm. In this approach, the solution obtained by optimizing the nuclear norm is equivalent to the one by the rank minimization model [144].

In a real-world application where datasets contain noise, the above approach for minimizing the nuclear norm can be reformatted as follows:

$$\min_R \|R\|_*, \quad s.t.\ |R_{ui} - A_{ui}| < \delta, (u,i) \in \Lambda, \tag{53}$$

where δ is the tolerance parameter to relax the $R_{ui} = A_{ui}, (u,i) \in \Lambda$ condition. This allows for the flexibility in the model to arrive at a solution where the missing entries due to noise, or some other data integrity issue, can be successfully approximated in a computationally non-taxing method. Further, when the observed values of the matrix are randomly sampled, matrix *R* can be recovered using only a small portion of the original data and achieve this with a high probability of accuracy using the nuclear norm regularization [145].

Upon achieving a completed approximation for the matrix whereby the noise is computationally resolved, predictions can be made from the input data by further incorporating into an algorithmic model. While methods exist based on the regularization approach applied to the underlying data, utilizing the nuclear norm regularization can be accomplished by the proximal gradient algorithm. One such approach uses the proximal gradient algorithm, which has been shown to be capable of solving closed convex optimization problems [38]. However, past approaches have been computationally slow to converge on a solution, which has led to recent attempts to improve the approach. Accelerated Proximal Gradient (APG) is an algorithm developed precisely for this reason [144], which converges in $O(1/\sqrt{\epsilon})$ iterations to solve the

nuclear norm minimization model. For this approach, for a known Y, a quadratic approximation of $\frac{1}{2}\|P_\Lambda(R) - P_\Lambda(A)\|_F^2$ at Y is given such that:

$$\frac{1}{2}\|P_\Lambda(R) - P_\Lambda(A)\|_F^2 \approx \frac{1}{2}\|P_\Lambda(Y) - P_\Lambda(A)\|_F^2 + \langle P_\Lambda(Y) - P_\Lambda(A), R - Y\rangle + \frac{1}{2\tau}\|R - Y\|_F^2, \quad (54)$$

where $\tau > 0$ is a proximal parameter. Substituting the quadratic approximation into the previously examined equation for matrix regularization, the minimization model then becomes:

$$min \;\; \lambda_1\tau\|R\|_* + \frac{1}{2}\left\|R - \left(Y - \tau\left(P_\Lambda(Y) - P_\Lambda(A)\right)\right)\right\|_F^2 \quad (55)$$

Then, APG generates $(R^{(j)}, Y^{(j)}, t^{(j+1)})$ in the following iterative fashion:

$$Y^{(j)} \leftarrow R^{(j)} + \frac{t^{(j-1)} - 1}{t^{(j)}}\left(R^{(j)} - R^{(j-1)}\right)$$

$$R^{(j)} \leftarrow D_{\lambda_6\tau}\left(Y^{(j)} - \tau\left(P_\Lambda\left(Y^{(j)}\right) - P_\Lambda(A)\right)\right)$$

$$t^{(j+1)} \leftarrow \frac{1+\sqrt{1+4(t^{(j)})^2}}{2} \quad (56)$$

As utilizing the nuclear norm regularization approach for matrix completion has the benefit of being able to handle noisy data, it has clear advantages for experimentally derived data where noise is present. Further, applying that methodology into APG can therefore yield practical predictions on future events or conditions from the underlying data.

10.4 APG results and analysis

To compare the results of the APG approach with the ALS approach, the same evaluation metrics were used. Table 17 shows the RMSE and SI for each result of using this approach. Overall, these metrics were an improvement compared to those obtained from the ALS based systems. Similar trends are seen here, where RMSE and SI are higher for the area and convex hull recommendations while significantly lower for the depth and width recommendations. Again, recommendations for the higher velocity parameters of 0.8 and 1.0 m/s were significantly higher,

which was also due to the sparsity of the data for those. As the overall RMSE values were lower for this exploration, it is possible that could be due to the inherent benefit of the APG approach in that it can handle noise reduction better, as described in the literature. Given the dataset has portions of missing or zero values, in addition to aberrant measurements likely due to noise, this approach was better suited for modeling the system.

TABLE 17
APG Results of Parameter Tested by Removal from Dataset

Test Parameter	Area		C.H.		Depth		Width	
	RMSE	SI	RMSE	SI	RMSE	SI	RMSE	SI
0.2m/s	254.3	19.29	382.6	27.46	23.95	25.25	22.64	42.67
0.4m/s	231.2	28.15	451.0	50.53	20.03	27.73	45.21	107.0
0.6m/s	425.1	31.06	521.6	32.80	32.21	47.98	52.12	64.50
0.8m/s	654.9	168.5	927.5	184.6	55.72	262.4	78.10	156.8
1.0m/s	251.0	335.2	362.8	458.2	75.16	284.1	71.12	297.3

This approach produced SI values near or under 25% for multiple instances, which is an improvement over those achieved under the ALS approach. This therefore suggests that the APG approach is more suitable for use in data generation, or making predictions on entirely new items, or new velocity settings that were not experimentally captured. This exploration is discussed in the Section 10.6.

10.5 Noise reduction with ALS and APG

One way to determine if the APG method outperformed ALS would be to evaluate each approach on a noisy dataset. This would help show how each handles those aberrant observations, and therefore help substantiate the notion that noise in the data is why one model outperformed

the other. The matrix creation process included all images experimentally obtained. Therefore, they included observations where the previously described noise elements were present. These values were identified as those for which there was no accurate measurement obtained as the vapor depression shape was missing or mostly obscured from that frame. For the matrix of vapor depression areas, these noise values represented 9% of the values. For the vapor depression depth matrix they were 11%, for width 8%, and for the convex hull matrix they were 9% of the values. As matrix completion is the underlying mechanism by which both of these recommender systems are based, the ability to successfully complete a noisy matrix suggests advantages for one system over another. This has relevance as there is a high likelihood of noise to be present in any experimental dataset.

In order to test the performance of the APG and ALS systems for noise reduction, each dataset was scrubbed of all missing data, by removing any column and row with a missing value. This effectively removed that frame or observed vapor depression image at that particular time instance from the data. Upon creating a matrix with only known values for area measurements, the phenomenon of missing data was simulated by applying Gaussian noise to artificially mask 10% of that known data, thereby approximating the amount of missing data. Each system was then applied to this new dataset, and the resultant values were obtained and evaluated based on the known data that was intentionally masked. This was done for both the ALS recommender system and the APG recommender system. After results were obtained, another 10% of the known data was masked and the models were run again. This was repeated to obtain results for missing data at increments of 10 up to 50% missing. The RMSE and SI values for these predictions are shown in Table 18.

Both models were able to approximate the missing data, with both having an SI near or under 20% for the small amount of missing data at 10% lost. This initial increment best describes the experimental dataset's missing values for each metric, as described above. The focus of this model was the area matrix, which had 9% of its data missing in the original dataset. From there, performance decreased for both approaches as more and more data was missing from the matrix. Yet APG was still able to recover 20% of the missing data with an SI at 23%. Overall, the RMSE and SI values were lower for the APG model, as expected. And as any dataset collected experimentally is likely to have some amount of data quality consideration, this technique is effective, which can be applied to systems with both stochastic and deterministic noise [42]. This suggests that approach is better suited for experimental datasets with small data integrity issues, both for recapturing lost data, and for making predictions with that data.

TABLE 18
ALS and APG Results with Induced Noise

Noise Level Induced	**ALS**		**APG**	
	RMSE	**SI**	**RMSE**	**SI**
10%	506.47	20.96	355.75	14.73
20%	651.92	26.98	562.32	23.28
30%	981.63	40.63	998.87	41.35
40%	3354.8	138.9	1874.3	77.58
50%	5598.2	231.7	4526.0	187.34

10.6 APG for data generation

Based on the results of the APG recommender system in comparison to the ALS, the former was used to make new predictions of geometries that were not experimentally derived. That is,

create new measurements, or new data, for combinations of process parameters that were not, and cannot, be physically measured. The goal of this work is to provide another metric to evaluate new geometries, using the learning mechanisms based on the matrix completion methods discussed in this chapter. This is in contrast to the deep learning based methods discussed in previous chapters. Using the matrix factorization approach, this new data generation could be accomplished without the need for image processing. This thereby removes the task of image pro-processing, and contouring, thus reducing the overall complexity of the workflow.

As was done with the CGAN discussed in Chapter 7, the new combinations of parameters that were used to create new data were: 150W and 0.3m/s, 200W and 0.3 m/s, 200 W and 0.3 m/s, 200W and 0.5m/s, 250W and 0.3m/s, 250W and 0.5m/s, 250W and 0.7m/s, 350W and 0.3m/s, 350W and 0.5m/s, 350W and 0.7m/s, 350W and 0.9m/s, 400W and 0.7m/s, and 400W and 0.9m/s. Table 19 depicts a summary of the original data, which gives an average value for all images of vapor depressions taken at each of the 35 experiments. The metrics depicted are depth, width, geometric area, and convex hull area. The new generated data by the APG system are included in bold. Here, the averages of APG produced measurements were taken for 2,000 observations, which therefore matched the number of images used to calculate the average values for each metric from the experimental data.

Overall, the model produced new measurements that followed the pattern of the distributions for the rise and fall of each characteristic method, with only two individual exceptions. The greatest fluctuations were for the 400W experiments, which again could be due to the missing data for the 0.4m/s run. In the matrix factorization mechanics, those zero values are considered and could have skewed some of these results. The results of model training with the known values, discussed in section 10.3 show that the depth and width measurements are more

reliable in terms of predictions. The RMSE and SI scores were considerably higher for area and convex hull in comparison. Therefore, these results show that the results for depth and width are also more reliable, with the artificial experiments displayed for 0.3, 0.5, 0.7, and 0.9 m/s. These values in particular are also those that can serve as inputs for the equations discussed in section 2.5, for thermodynamic model comparison and validation.

10.7 Recommender system summary

The work described in this chapter describes two separate recommender system approaches for modeling data collected experimentally. These systems, which are a type of machine learning, are generally used for predicting user behavior. In their most common applications, recommender systems are used to make recommendations (hence their name) for users and items. When a new item is introduced, the system can attempt to include it in its recommendations based on past user behavior when held in comparison quantitatively to other users' behaviors. This scenario, known as collaborative filtering, is one of the foundational principals of the recommender system. The recommendations are based upon a matrix of values which represent numerical rankings for user-item combinations.

In the case of the LPBF process, the laser intensity is modeled as the user and the laser velocity is modeled as the items. The dataset is arranged such that the area measurements corresponding to that intensity and velocity is populated in the data matrix – in the place of user-item numerical rankings. The goal then becomes for the recommender system to make recommendations, or predictions for the area measurements, and subsequently convex hull area, depth, and width of the vapor depression. However, prior to introducing a new untested velocity setting into the system, which is similar to a new item being added to the system, the performance of the model should be tested on a held out portion of the data. Therefore, known values were

withheld from the matrix such that the system could be used to attempt to reproduce that withheld data.

The two recommender systems evaluated were the Alternating Least Square, ALS, and Accelerated Proximal Gradient, APG, based approaches. The latter of which consistently outperformed the former, thereby suggesting its stronger likelihood for modeling and making predictions from the LPBF data. The APG approach was then chosen to make new predictions. In this fashion the recommender system was used to generate new data – new measurements. While the model was able to accomplish this task, the results from the holdout evaluations for known data suggest that the APG approach works best for depth and width measurements, as opposed to those characteristics that have more variability and a larger range of values – the geometric area and convex hull area of the vapor depressions. The results shown here suggest that recommender systems can best be used for modeling those parameters as the underlying matrix decomposition approaches were better able to encode those values and the latent features therein.

Limitations in these approaches exist. First, the data must be arranged in a two dimensional matrix format, which thereby only allows two dimensions of data for analysis. Manipulating the data into an appropriate format does take effort, as the build sensors do not output data in this format automatically. While the work described here focuses only on two dimensions, laser power and laser velocity, in Chapter 2 it was discussed how there are many more process parameters that influence microstructure quality. Should we want to generate structure geometries to also factor in hatch spacing, laser spot size, and layer thickness – these additional dimensions of data would require different methodologies. The solution to this problem would likely involve higher dimensional tensor factorization and completion methods. These techniques are similar to the goals

TABLE 19
Average Values for Vapor Depression Geometries by APG

P	V	DEPTH	WIDTH	G. AREA	C.H. AREA
150	0.2	63.06	38.22	492.46	502.32
150	**0.3**	**38.94**	**24.65**	**844.12**	**214.14**
150	0.4	9.52	25.63	57.49	69.52
150	0.6	0.00	0.00	0.00	0.00
150	0.8	0.00	0.00	0.00	0.00
150	1.0	0.00	0.00	0.00	0.00
150	1.2	0.00	0.00	0.00	0.00
150	1.4	0.00	0.00	0.00	0.00
200	0.2	14.95	42.86	148.38	175.84
200	**0.3**	**8.881**	**67.81**	**278.12**	**298.59**
200	0.4	39.69	32.03	339.50	452.52
200	**0.5**	**43.68**	**35.71**	**654.45**	**600.84**
200	0.6	54.95	68.87	743.68	856.23
200	0.8	0.00	0.00	0.00	0.00
200	1.0	0.00	0.00	0.00	0.00
200	1.2	0.00	0.00	0.00	0.00
200	1.4	0.00	0.00	0.00	0.00
250	0.2	45.65	42.78	475.28	489.51
250	**0.3**	**100.85**	**45.62**	**874.78**	**1001.69**
250	0.4	109.51	77.06	1299.75	1532.23
250	**0.5**	**105.45**	**98.24**	**1155.63**	**901.85**
250	0.6	34.33	81.19	532.50	635.72
250	**0.7**	**24.62**	**88.32**	**412.56**	**678.51**
250	0.8	9.66	26.08	57.56	58.21
250	1.0	0.00	0.00	0.00	0.00
250	1.2	0.00	0.00	0.00	0.00
350	0.2	87.77	54.47	2803.14	2925.91
350	**0.3**	**303.89**	**75.01**	**2363.95**	**2602.89**
350	0.4	202.41	76.58	2410.46	2665.32
350	**0.5**	**125.26**	**152.31**	**3052.98**	**2543.51**
350	0.6	106.44	134.49	2715.03	2987.55
350	**0.7**	**98.62**	**89.65**	**1795.88**	**2649.13**
350	0.8	53.96	112.09	1043.80	1222.54
350	**0.9**	**55.45**	**205.36**	**945.27**	**836.94**
350	1.0	16.13	61.28	185.11	211.74
350	1.2	0.00	0.00	0.00	0.00
350	1.4	0.00	0.00	0.00	0.00
400	0.2	262.73	86.93	4270.17	4676.23
400	0.4	0.00	0.00	0.00	0.00
400	0.6	139.92	119.54	2852.02	2962.46
400	**0.7**	**203.51**	**354.51**	**2656.62**	**2862.69**
400	0.8	42.56	110.81	842.34	952.95
400	**0.9**	**58.42**	**99.87**	**752.93**	**808.44**
400	1.0	16.17	58.32	189.31	195.62
400	1.2	0.00	0.00	0.00	0.00
400	1.4	0.00	0.00	0.00	0.00

of the matrix completion work described in this chapter; however, they can accommodate high dimensional datasets. It should be noted that there are very little results in the literature that demonstrates success in tensor completion; these techniques are still very much in their infancy compared to other areas of machine learning. Future work could therefore explore tensor completion by nuclear norm minimization, which theoretically should yield desirable results.

CHAPTER 11

COMPARISON OF GENERATIVE MODELS

While observations of the original 35 experimental combinations were used for drawing insight, the potential to have additional data for experimental combinations that were not produced would be invaluable for additional thermophysics model validation. Generative models were therefore developed and evaluated for this purpose, with the goal of increasing the variety of the data collected through computational processes for artificial data generation. Two generative adversarial networks were developed. The difference between the two being that one model had a conditional aspect to it, that is, an additional component was used for model training in the form of a numerical variable for an input. This is in addition to the image based data used for training, which is the sole input for the general network. Using the laser intensity and laser velocity as numerical inputs allowed the network to create more accurate images, as evaluated by a variety of metrics to include the intersection over union, Hausdorff distance, root mean squared error, and normalized root mean squared error also known as the scatter index. In all four of these metrics, the conditional model outperformed the general model, and produced images that were indistinguishable by human inspection from the original experimentally derived images. Based on these results, new data was generated for new combinations of laser process parameters, which increased the total available data by nearly 50% using the conditional model. The additional data was therefore capable of adding additional validation for process characterization.

Next, a comparable deep learning based method for data generation was developed, using variational autoencoders and conditional variational autoencoders. The conditional aspect worked similarly to that of the generative adversarial network, however model training commenced differently. The results of both of the VAE models were poor compared to those of the GAN

models, for all of the evaluation metrics. Although the conditional autoencoder proved superior to the general variational autoencoder, its output was still low in comparison to either of the generative adversarial networks. New data could potentially be derived from these models, but given the superior performance of the previous approaches, the autoencoder was not used for artificial data generation. However, even the better performing CVAE trailed the performance of both the GAN model and the CGAN. By comparison, the GAN had three tests where the SI value was under 30. The lowest SI value was 36.12, as seen for the CVAE's output at the 0.6m/s parameter for convex hull area. And while this only occurs once for the CVAE output, it occurs for three out of four of the geometric measures for the GAN at 0.6 m/s (area at 27.66, convex hull at 24.51, and width at 23.20).

A third approach for data generation was then developed, where instead of using image-based approaches, a purely numerical approach was used. To accomplish this, the dimensions and measured characteristics of the vapor depressions were encoded in a matrix format, where matrix completion was then performed with two recommendation systems employed to recommend, or generate, new measurements. The alternating least squares and the accelerated proximal gradient algorithms were used to generate new measurements. The latter of which was shown to produce more accurate values for geometric measurements of area, convex hull area, depth, and width of the vapor depressions. While not an image-based process and therefore not able to be evaluated by some of the previous methods, the root mean squared error and the standard index could still be calculated for the output of the models. Therefore, a direct comparison could be made to the deep learning based frameworks. Overall, these models were successful in generating new data, with results in between those of the generative adversarial networks and the variational autoencoders.

A summary of the results for the area calculations with the CGAN, CVAE, and APG models are show in Table 20. Here, the superior of the two models for each technique – the generative adversarial network, variational autoencoder, and recommender system based models were chosen for comparison. While the GAN did provide superior results to the CVAE, as seen in Tables 8 and 15 – the CVAE was included Table 20 to depict the disparity in the results for the better autoencoder methodology. The goal of Table 20 therefore being to show how the better output from each of the 3 learning methodologies differed. While the CGAN performed the best, the recommender system based on APG performed second best, with results that closely matched the GAN in Table 8. As shown above, the APG model had two tests where the SI was close to 30. These results suggest that the matrix decomposition approach and nuclear norm minimization operation can learn the latent features in the image data comparatively well to the deep learning convolutional approach employed by the GAN.

TABLE 20
Comparison of Generative Results for Area Calculations

Test Parameter	**CGAN**		**CVAE**		**APG**	
	RMSE	**SI**	**RMSE**	**SI**	**RMSE**	**SI**
0.2m/s	279.05	18.52	469.12	73.43	254.3	19.29
0.4m/s	315.21	20.21	446.57	90.15	231.2	28.15
0.6m/s	250.63	19.64	693.88	61.35	425.1	31.06
0.8m/s	405.12	50.25	814.64	147.6	654.9	168.5
1.0m/s	360.15	75.96	1149.2	431.2	251.0	335.2

Using a recommender system approach has the benefit of not needing to directly process all of the images in the dataset, as the input data is numerical. While work still needed to be done to extract the numerical measurements from the image data, once done, the computational complexity of deriving actionable intelligence from these data becomes computationally less taxing compared to the resource intensive convolutional approach. That makes this an ideal technique for future use in image datasets, as the numerical data can also be used for other modeling and forecasting purposes. Additionally, the recommender system approach has the benefit of being able to make many generative outputs at the same time. As was discussed extensively, the incomplete matrix for a new user, and therefore a new laser speed setting for which no data was obtained, can be computed. This allows the recommender system approach to produce measurements for all laser intensity combinations at one time, unlike the conditional GAN where the y inputs are manipulated one at a time for each combination of parameters. Recommender systems can therefore be faster computationally, and more efficient for computing these outputs.

CHAPTER 12

CONCLUSION AND FUTURE WORK

Manufacturing using advanced complex materials, such as metal superalloys is a discipline of major interest to the aerospace industry, among many others. The advantages of developing structures for these materials using additive manufacturing have shown many advantages in the ability to create new custom made parts, quickly and relatively cheaply compared to historical methods. However, in order to move such a system to high yield production, there must be a thorough and comprehensive evaluation criteria on the outcome of that manufacturing process – the quality of the final build. As yet, there is no certification standard for parts made using additive manufacturing systems, which includes the industry leading technology of laser powder bed fusion additive manufacturing. Therefore, ongoing research into way to qualify and quantify the quality of the build process is ongoing throughout government, academia, and industry.

While challenges exist in finding a novel methodology to certify additive manufactured components, one advantage is that these systems can be configured to collect massive amounts of data in an experimental setting. For instance, high speed imaging of the build process, which can provide image-based data from the build in-situ, which allows developers to see how the quality of the material is affected by the actual build parameters – to include the characteristics of the laser which is the heat source for the fusion of the metal particles in the additive manufacturing process. These data can then be used to validate the theoretical framework, the expected values for certain physical characteristics and phenomena as informed by thermophysics and fluid dynamics models. Such characteristics include the behavior of the liquid bodies, which are created by the laser heat which melts the metal substrate powder in a near instantaneous fashion as the laser moves over the material. This area, the liquid melt pool, solidifies to form the final internal microstructure of the

material. And its shape and solidification pattern have a direct impact on the physical quality of the material, which includes any defects in the material. In addition to the liquid melt pool, there is the gaseous body known as the vapor depression, which is the metal area directly beneath the laser that instantaneously evaporates into a gas -of which the melt pool surrounds.

Collecting data during the build using dynamic x-ray radiography provides tens of thousands of images for each build, nearly one hundred thousand images for a single object built in an experimental setting. Using this technique, data was collected for a variety of builds using different combinations of build process parameters - namely the laser velocity and the laser intensity. The laser velocity determines heat buildup or how the material cools as it moves over the material. Slower velocities lead to deeper vapor depressions and larger melt pools as the material is under the heat source for longer times, compared to when the laser is moving quicker and there is less time for the material to evaporate or liquify. Similarly, the laser intensity which determines how strong the heat source will be, will also determine the size and shape of the vapor depression and melt pool. The dynamic x-ray radiography provides an easy to examine visualization of each of these areas for each of the experimental builds, where it can clearly be seen how the size and shape of those two bodies change depending on the experimental settings.

While the amount of data collected was vast, that in itself creates a two-fold problem limiting its usefulness for product evaluation. First, the volume of data did not mean that there was a variety of data. The 35 experiments provided a large amount of data, but there were many combinations of laser process parameters that were not conducted, and therefore no data exists for the characteristics of materials for these hypothetical combinations. Second, the large amount of data makes a visual inspection an impossible task. In order to mitigate these challenges in data

evaluation, machine learning applications were developed to leverage the computational ability of advanced algorithmic mechanisms to learn from these data and make quantifiable determinations.

The outcomes of the machine learning employed were as follows. First, using image based image classification techniques for deep learning, the quality of builds was evaluated to determine the presence, or lack thereof, of defects in the various experimental builds. This also included a thorough computational geometry evaluation of the gaseous vapor depression which is responsible for the most common types of defects as gas is trapped and left behind in the solidified structure. Geometric feature tracking allowed for a link to be developed in a quantitative fashion for the vapor depression and the melt pool. It was shown that it is possible to predict the presence of defects, and therefore predict which combinations of process parameters will lead to defects – before those defects even occur. The underlying features that constitute the vapor depressions can be learned by their representation in the feature map created by the convolutional processes from the deep learning mechanisms. These results also provided direct validation for some of the underlying thermophysics models, thereby justifying that those models are appropriate mechanisms for process certification.

Further this work improves upon the state-of-the-art in generative models by developing a methodology for incorporating a continuous feature representation into the training such that the model can potentially learn the distributional relationships of that feature with regard to the underlying principals governing that representation. While work exists in this regard in conditional generative adversarial networks, the process developed here used thermophysics based equations to characterize and predict fluid properties that are encoded in the deep learning framework as latent features. Additionally, the work described here trained a generative model in an end-to-end

fashion which can be generalized to many other image representation, generation, and restoration problems in the machine learning subfield of computer vision.

Future work in generating artificial data for LPBF investigations could include evaluating utilizing the CGAN with datasets for various materials, in addition to images collected for IN-718 builds. This additional component will be conditioned by incorporating another input value for the CGAN to learn to distinguish images from those builds using other superalloy materials, such as the titanium allow Ti64, or the Nickel based superalloy CMSX-4. Additionally, future work will incorporate the LPBF laser spot size, as this process parameter is also highly influential in the heat absorption in the substrate material during the build process.

The work performed here contribute to the computational science breadth of knowledge in image generation by the development of the multiple generative models for image generation with a unique series of characteristics. While a materials science problem set was explored in this work, the methods developed here could be applied to other areas, such as biological sciences where x-ray image capture is also used. Generating new data could therefore be used to demonstrate hypothetical manifestations of certain organismal components, including cancer tumor genesis among many other use cases. Furthermore, the time resolved geometric sequencing approaches that were then fed into the matrix decomposition architecture and recommender system could be applied to many other image-based problems where additional data could prove useful. The recommender system operates in a fundamentally different way than deep learning, with advantages to include lower computational complexity and therefore possibly easier model convergence. This has benefits for users who might not have the computational resources for the deep learning framework, especially if more complex and higher resolution images are required. The recommender system framework developed here can be conducted regardless of the resolution

or size of the images, both of which have major impacts on the complexity of a deep learning implementation.

Finally, this work had a direct benefit to government and industry research as NASA Langley Research Center has utilized the outputs of the data here to contribute to the larger body of work around additive manufacturing process certification. Following that acceptance, this work has been presented to researchers throughout the agency who research and develop machine learning and artificial intelligence for the multitude of problems faced by NASA as it works to accomplish its mission. Future work will continue to apply these image based and non-image based generative modeling techniques to better understand problems faced throughout industry. While additive manufacturing was the focus of this work, these techniques can be applied to other domains within materials science to include non-destructive evaluation, where CT scan data can provide other image-based datasets for investigation. Ultimately, the development of GAN and CGAN for aerospace applications will allow for more artificial renderings of hypothetical constructions.

CHAPTER 13

PUBLICATIONS

1. “Time-resolved geometric feature tracking elucidates laser-induced keyhole dynamics,” *Integrating Materials and Manufacturing Innovation*, vol. 10, pp. 677-688.
2. “Image synthesis using conditional GANs for selective laser melting additive manufacturing,” submitted to 2022 IEEE International Joint Conference on Neural Networks.
3. “A comparison of generative adversarial networks to convolutional neural networks for single track melt pool segmentation as evaluated by Hausdorff distance,” submitted to Additive Manufacturing Letters.
4. "Understanding the Keyhole Dynamics in Laser Welding with Computer Vision and Data Analytics Applied to Time-resolved X-ray Imaging," accepted by 6th World Congress on Integrated Computational Materials Engineering.
5. “Convolutional Neural Networks for Image Classification in Metal Selective Laser Melting Additive Manufacturing,” accepted by First World Congress on Artificial Intelligence in Materials and Manufacturing.
6. “Understanding the Keyhole Dynamics in Laser Processing Using Time-Resolved X-ray Imaging Coupled With Computer Vision and Data Analytics.” To be published in conference proceedings for *TMS 2021, Symposium: Data Science and Analytics for Materials Imaging and Quantification.* March 2021.

REFERENCES

[1] I. Gibson, A. Srinath, "Simplifying Medical Additive Manufacturing: Making the Surgeon the Designer," *The International Design Technology Conference*, July, 2015.

[2] S. Ridinger, "NASA Advances Additive Manufacturing for Rocket Propulsion." Retrieved from NASA: https://www.nasa.gov/centers/marshall/news/nasa-advances-additive-manufacturing-for-rocket-propulsion.html, 2008.

[3] L. Gibson, "TTT Project Technical Challenges." Retrieved from NASA: https://www.nasa.gov/aeroresearch/programs/tacp/ttt/technical-challenges, 2016.

[4] L. Gibson, "Transformational Tools and Technology." Retrieved from NASA: https://www.nasa.gov/aeroresearch/programs/tacp/ttt, 2020.

[5] T. M. Pollock, S. Tin, "Nickel-based Superalloys for Advanced Turbine Engines: Chemistry, Microstructure and Properties." *Journal of Propulsion and Power*, vol. 22, iss. 2, pp. 361-374, 2006.

[6] Y. Huang, C.M. Leu, J. Mazumder, A. Donmez, "Additive Manufacturing: Current State, Future Potential, Gaps and Needs, and Recommendations". *Journal of Manufacturing Science and Engineering*, vol. 137, iss. 1, 2015.

[7] F. Calignano, D. Manfredi, E. P. Ambrosio, S. Biamino, M. Lombardi, E. Atzeni, A. Salmi, P. Minetola, L. Iuliano, P. Fino, "Overview on Additive Manufacturing Technologies". *Proceedings of the IEEE*, vol. 105, iss. 4, pp. 593-612, 2017.

[8] T. McMahan, "NASA 3-D Prints First Full-Scale Copper Rocket Engine Part." Retrieved from NASA https://www.nasa.gov/marshall/news/nasa-3-D-prints-first-full-scale-copper-rocket-engine-part.html, 2015.

[9] T. J. Horn, O. L. A. Harrysson, "Overview of Current Additive Manufacturing

Technologies and Selected Applications." *Science Progress*, vol. 95, iss. 3, pp. 255-282, 2012.

[10] T. Monaghan, A. J. Capel, S. D. Christie, R. A. Harris, R. J. Friel, "Solid-state Additive Manufacturing for Metallized Optical Fiber Integration." *Composites Part A: Applied Science and Manufacturing*, vol. 76, pp. 181-193, 2015.

[11] N. D. Parab, C. Zhao, R. Cunningham, L. I. Escano, K. Fezzaa, W. Everhart, T. Sun, "Ultrafast X-ray Imaging of Laser-Metal Additive Manufacturing Processes." *Journal of Sunchrotron Radiation*, vol. 25, pp. 1467-1477, 2018.

[12] J. O. Milewski, *Additive Manufacturing of Metal: From Fundamental Understanding Technology to Rocket Nozzles, Medical Implants, and Custom Jewelry*, Springer, 2017.

[13] M. Yakout, A. Cadamuro, M. A. Elbestawi, S. C. Veldhuis, "The Selection of Process Parameters in Additive Manufacturing for Aerospace Alloys." *International Journal of Advanced Manufacturing Technology*, vol. 92, pp. 2081-2098, 2017.

[14] H. Gong, K. Ra, H. Gu, T. Starr, B. Stucker, "Analysis of Defect Generation in Ti6Al4v Parts Made Using Powder Bed Fusion Additive Manufacturing Processes." *Additive Manufacturing*, vol. 4, pp. 87-98, 2014.

[15] Q. Jia, H. Gu, "Selective Laser Melting Additive Manufacturing of Inconel 718 Superalloy Parts: Densification, Microstructure and Properties." *Journal of Alloys and Compounds*, vol. 585, pp. 713-721, 2014.

[16] M. Khalil, G.H Teichart, C. Allerman, N. M. Heckman, R. E. Jones, K. Garikkipati, B. L. Boyce, "Modeling Strength and Failure Variability Due to Porosity in Additively Manufactured Metals." arXiv eprint=2001.02058, 2019.

[17] R. Cunningham, C. Zhao, N. Parab, C. Kantzos, J. Pauza, K. Fezzaa, A. D. Rollett, "Keyhole Threshold and Morphology in Laser Melting Revealed by Ultrahigh Speed X-ray Imaging." *Science*, vol. 363, pp. 849-852, 2019.

[18] C. Zhao, K. Fezzaa, R. Cunningham, W. Haiden, F. De Carlo, L. Chen, T. Sun, "Real Time Monitoring of Laser Powder Bed Fusion Process Using High Speed X-ray Imaging and Diffraction." *Nature*, vol. 7, iss. 3602, pp. 1-11, 2017.

[19] F. Tenner, B. Berg, C. Brock, F. Klämpfl, M. Schmidt, "Experimental Approach for Quantification of Fluid Dynamics in Laser Metal Welding." *J. Laser Appl.* vol.27, 2015.

[20] I. Eriksson, J. Powell, A.F.H. Kaplan, "Melt Behavior on the Keyhole Front During High Speed Laser Welding." *Opt. Lasers Eng*, vol. 51, pp. 735–740, 2015.

[21] A. Matsunawa, N. Seto, M. Mizutani, S. Katayama, "Liquid Motion in Keyhole Laser Welding." *Int. Congr. Appl. Lasers Electro-Optics, Laser Institute of America*, pp. G151–G160, 1998.

[22] M. Sohail, S.-W. Han, S.-J. Na, A. Gumenyuk, M. Rethmeier, "Characteristics of Weld Pool Behavior in Laser Welding with Various Power Inputs." *Weld. World*, vol. 58, pp. 69–277, 2014.

[23] B. Chang, C. Allen, J. Blackburn, P. Hilton, D. Du, "Fluid Flow Characteristics and Porosity Behavior in Full Penetration Laser Welding of a Titanium Alloy." *Metall. Mater. Trans. B*, vol. 46, pp. 906–918, 2015.

[24] L. Aucott, H. Dong, W. Mirihanage, R. Atwood, A. Kidess, S. Gao, S. Wen, J. Marsden, S. Feng, M. Tong, T. Connolley, M. Drakopoulos, C.R. Kleijn, I.M. Richardson, D.J. Browne, R.H. Mathiesen, H.V. Atkinson, "Revealing Internal Flow Behaviour in Arc Welding and Additive Manufacturing of Metals." *Nat. Commun.*, vol. 9, pp. 5414, 2018.

[25] M. Bachmann, V. Avilov, A. Gumenyuk, M. Rethmeier, “About the Influence of a Steady Magnetic Field on Weld Pool Dynamics in Partial Penetration High Power Laser Beam Welding of Thick Aluminium Parts.” *Int. J. Heat Mass Transf,* vol. 60, pp. 309–321, 2013.

[26] B. Schoinochoritis, D. Chantzis, K. Salonitis, “Simulation of Metallic Powder Bed Additive Manufacturing Processes with the Finite Element Method: A Critical Review.” *Proc. Inst. Mech. Eng. Part B: J. Eng. Manuf.,* vol. 231, pp. 96–117, 2017.

[27] E. Charniak, *Introduction to Deep Learning*, MIT Press, 2018.

[28] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2016.

[29] F. M. Bayat, M. Prezioso, B. Chakrabarti, “Implementation of Multilayer Perceptron Network with Highly Uniform Passive Memristive Crossbar Circuits.” *Nat Commun,* vol. 9, iss. 2331, 2018.

[30] R. Venkatesan, B. Li, *Convolutional Neural Networks in Visual Computing: A Concise Guide,* CRC Press, 2018.

[31] D. Ciresan, U. Meier, J. Masci, L. M. Gambardella, J. Schmidhuber, “Flexible, High Performance Convolutional Neural Networks for Image Classification.” *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, vol. 2, iss. 2, pp. 1237-1242, 2011.

[32] S. Khan, H. Rahmani, S. A. A. Shah, *A Guide to Convolutional Neural Networks for Computer Vision,* Morgan Claypool Publishers, 2018.

[33] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*, Springer, 2018.

[34] U. Michelucci, *Advanced Applied Deep Learning: Convolutional Neural Networks and Object Detection*, Apress, 2019.

[35] L. Lu, X. Wang, G. Carneiro, L. Yang, *Deep Learning and Convolutional Neural*

Networks for Medical Imaging and Clinical Informatics, Springer, 2019.

[36] M. Hirz, B. Walzel, “Sensor and Object Recognition Technologies for Self-Driving Cars,” *Computer Aided Design and Application*, vol. 15, iss. 4, 2018.

[37] W. Carrer-Neto, M. L. Hernandez-Alcaraz, R. Valencia-Garcia, F. Garcia-Sanchez, "Social Knowledge-Based Recommender System: Application to the Movies Domain," *Expert Systems with Applications*, vol. 39, pp. 10990-11000, 2012.

[38] H. Luo, M. Li, S. Wang, Q. Liu, Y. Li, J. Wang, "Computational Drug Repositioning using Low-Rank Matrix Approximation and Randomized Algorithms," *Bioinformatics*, vol. 34, iss. 11, pp. 1904-1912, 2018.

[39] M. Kunaver, T. Pozrl, "Diversity in Recommender Systems - A Survey," *Knowledge-Based Systems*, vol. 123, pp. 154-162, 2017.

[40] Y. Liang, D. Wu, G. Liu, Y. Li, C. Gao, Z. Ma, "Big Data-Enabled Multiscale Serviceability Analysis for Aging Bridges," *Digital Communications and Networks*, vol. 2, pp. 97-107, 2016.

[41] J. Bobadilla, F. Ortega, A. Hernando, A. Gutierrez, "Recommender Systems Survey," *Knowledge-Based Systems*, vol. 46, pp. 109-132, 2013.

[42] E. J. Candes, T. Tao, "The Power of Convex Relaxation: Near-Optimal Matrix Completion," *IEEE Transactions on Information Theory*, vol. 56, pp. 2053-2080, 2010.

[43] M. Udell, C. Horn, R. Zadeh, S. Boyd, "Generalized Low Rank Models," *Foundations and Trends in Machine Learning*, vol. 9, pp. 61-68, 2016.

[44] Y. Koren, R. Bell, C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, pp. 30-37, 2009.

[45] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, "Application of Dimensionality Reduction in Recommender System - A Case Study," in *ACM WebKDD Workshop*, 2000.

[46] Xu, P. Pan, "A New Algorithm for Positive Semidefinite Matrix Completion," *Journal of Applied Mathematics*, vol. 2016, pp. 1-5, 2016.

[47] D. Foster, Generative *Deep Learning: Teaching Machines to Paint, Write, Compose and Play*, O'Reilly Media Inc., 2019.

[48] B. Bowman, *Anomaly Detection Using a Variational Autoencoder Neural Network with a Novel Objective Function and Gaussian Mixture Model Selection Technique*, Naval Postgraduate School, 2019.

[49] J. Kelleher, *Deep Learning*, MIT Press, 2019.

[50] T. Bossomaier, L. Barnett, M. Harrè, J. T. Lizier, *An Introduction to Transfer Entropy: Information Flow in Complex Systems*, Springer, 2016.

[51] O. Calin, *Deep Learning Architectures: A Mathematical Approach (Springer Series in the Data Sciences)*, Springer, 2020.

[52] J. An, S. Cho, "Variational Autoencoder Based Anomaly Detection Using Reconstruction Probability". *Speacial Lecture on IE*, vol. 2, iss. 1, 2015.

[52] X. Li, J. She, "Collaborative Variational Autoencoder for Recommender Systems." Proceedings of the 23rd *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 305-314, 2017.

[54] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Oziar, A. Courville, Y. Bengio, "Generative Adversarial Nets." *Proceedings of the Internatioal Conference on Neural Information Processing Systems*, pp. 2672-2680, 2014.

[55] J. Langr, V. Bok, *GANs in Action: Deep Learning with Generative Adversarial Networks,* Manning Publications, 2019.

[56] T. Schlegl, P. Seeböck, S. Waldstein, U. Schmidt-Erfurth, G. Langs, "Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Maker Discovery," *Proceedings of the IPMI 2017,* 2017.

[57] L. Ma, R. Shuai, W. Liu, C. Ye, "Combining DC-GAN with ResNet for Blood Cell Image Classification," Medical and Biological Engineering and Computing, vol. 58, iss. 6, pp. 1251-1264, 2020.

[58] T. W. M. Liu, A. Tao, J. Kautz, B. Catanzaro, "High Resolution Image Synthesis and Semantic Manipulation with Conditional GANs," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 8798-8807, 2018.

[59] K. Regmi, A. Borji, "Cross-View Image Synthesis Using Conditional GANs," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 3501-3510, 2018.

[60] J. Song, J. Zhang, L. Gao, X. Liu, H. T. Shen, "Dual Conditional GANs for Face Aging and Rejuvenation," *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence,* pp. 899, 2018.

[61] A. Brock, J. Donahue, K. Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis." arXix:1809.11096, 2018.

[62] T. Karras, S. Laine, T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks." *CVPR 2019,* 2019.

[63] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, D. Metaxas, “StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks.” arXiv:1612.03242v2, 2017.

[64] M. Dia, E. Savary, M. Melchior, F. Courbin, “Galaxy Image Simulation Using Progressive GANs.” *Astronomical Data Analysis Software and Systems XXIX*, vol. 527, pg. 175, 2019.

[65] Y. Skandarani, P. Jodoin, A. Lalande, “GANs for Medical Image Synthesis: An Empirical Study.” asXiv:2015.05318, 2021.

[66] N. Wood, D. Hoelzle, “Temperature States in Powder Bed Fusion Additive Manufacturing Are Structurally Controllable and Observable.” arXiv eprint=2001.02519, 2020.

[67] K. Kim, T. H. Chalidabhongse, D. Harwood, L. Davis, “Real Time Foreground-Background Segmentation Using Codebook Model.” *Real-Time Imaging*, vol. 11, iss. 3, pp. 172-185, 2005.

[68] N. Dey, S. Dutta, G. Dey, S. Chakraborty, R. Ray, P. Roy, “Adaptive Thresholding: A Comparative study”. *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies*, 2014.

[69] F. Shafait, D. Keysers, T. M. Breuel, (2008) "Efficient Implementation of Local Adaptive Thresholding Techniques Using Integral Images", *Proc. SPIE 6815,* 2008.

[70] L. Velasco, Y. Alanzi, E. McClellan, P. Ambrozewicz, N. Sato, T. Liu, M. Melnitchouk, P. Kuchera, Y. Li, “cFat-GAN: Conditional Simulatinon of Electron-Proton Scattering Events with Variate Beam Energies by a Feature Augmented and Transformed Generative Adersarial Network.” arXiv:2001.11103, 2020.

[71] A. Krizhevsky, I. Sutskever, G. Hinton, “ImageNet Classification with Deep

Convolutional Neural Networks," *Communications of the ACM*, vol. 60, iss. 6, pp. 84-90, 2017.

[72] B. Zoph, V. Vasudevan, J. Shlens, Q. Le, "Learning Transferable Architectures for Scalable Image Recognition." *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2018.

[73] G. Huang, Z. Liu, L. Van der Maaten, K. Weinberger, "Densely Connected Convolutional Networks." *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition,* 2017.

[74] B. Zoph, Q. Le, "Neural Architecture Search with Reinforcement Learning." *Proceedings of the 5th International Conference of Learning Representations,* 2017.

[75] S. Ioffe, C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." in *Proceedings of the 32nd International Conference of Machine Learning*, 2015.

[76] Y. Boureau, J. Ponce, Y. LeCun, "A Theoretical Analysis of Feature Pooling in Visual Recognition." *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[77] X. Glorot, A. Bordes, Y. Bengio, "Deep Sparse Rectifier Neural Networks." *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 2011.

[78] S. J. Pan, Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions in Knowledge Data Engineering*, vol. 22 iss. 10, pp. 1345-1359, 2010.

[79] M. Abadi, et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems." arXiv: 1603.04467v2, 2016.

[80] F. Chollet, “Transfer Learning and Fine-Tuning.” Created: April, 15, 2020. [Online]. Available: https://keras.io/guides/transfer_learning/, 2020.

[81] Ioffe, S. “Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models.” *Proceedings of the 31st Conference on Neural Information Processing Systems*, 2017.

[82] M. De Berg, M. Van Kreveld, M. Overmars, O. Schwarzkopf, *Computational Geometry, 2nd Edition*, Springer, 2000.

[83] R. A. Jarvis, "On the Identification of the Convex Hull of a Finite Set of Points in the Plane". *Information Processing Letters*. vol. 2 pp. 18–21, 1973.

[84] R. L. Graham, “An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set.” *Information Processing Letters,* vol. 1, pp. 132–133, 1972.

[85] T. Hastie, J. Robert, J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2009.

[86] V. Cox, *Translating Statistics to Make Decisions*, Apress, 2017.

[87] M. Vega, R. Pardo, E. Barrado, L. Debán, “Assessment of Seasonal and Polluting Effects on the Quality of River Water by Exploratory Data Analysis.” *Water Research*, vol. 32, iss. 12, pp. 3581-3592, 1998.

[88] T. Nakaya. “Visualizing Crime Clusters in a Space-Time Cube: An Exploratory Data Analysis Approach Using Space-Time Kernel Density Estimation and Scan Statistics.” *Transactions in GIS*, vol. 14, iss. 3, 2010.

[89] E. W. T. Ngai, L. Xiu, D. C. K. Chau, “Application of Data Mining Techniques in Customer Relationship Management: A Literature Review and Classification.” *Expert Systems with Applications*, vol. 36, iss. 2, pp. 2592-2602, 2009.

[90] T. Kunungo, D. M. Mount, N. Netanyahu, C. D. Piatko, R. Silverman, A. Y. Wu., "An Efficient K-means Clustering Algorithm: Analysis and Implementation." *Pattern Analysis and Machine Intelligence*, vol. 24, iss. 7, pp. 881-892, 2002.

[91] S. Fortune, "A Sweepline Algorithm for Voronoi Diagrams." *Proceedings of the second annual symposium on computational geometry*, pp.313–322, 1986.

[92] K. Wong, H. A. Müller, "An Efficient Implementation of Fortune's Plane-Sweep Algorithm for Voronoi Diagrams", CiteSeerX 10.1.1.83.5571, 1991.

[93] L. Jin, D. Kim, L. Mu, D. Kim, S. Hu, "A Sweepline Algorithm for Euclidean Voronoi Diagrams for Circles." *Computer-Aided Design*, vol. 38, iss. 3, pp. 260-272, 2006.

[94] C. Franley, A. E. Raferty, "How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis." *The Computer Journal*, vol. 41, iss. 8, pp. 578-588, 1998.

[95] W. E. King, H. D. Barth, V. M. Castillo, G. F. Gallegos, J. W. Gibbs, D. E. Hahn, C. Kamath, A. M. Rubenchik, "Observation of Keyhole Mode Laser Melting in Laser Powder-bed Fusion Additive Manufacturing." *Journal of Materials Process Technology*, vol. 214, iss. 12, pp. 2915-2925, 2014.

[96] R. Cunningham, S. P. Narra, C. Montgomery, J. Beuth, A. D. Rollett, "Synchrotron-Based X-ray Microtomography Characterization of the Effect of Processing Variables on Porosity Formation in Laser Power-bed Additive Manufacturing of Ti-6Al-4V." *JOM*, vol. 69, pp. 479-484, 2017.

[97] C. Zhao, Q. Guo, X. Li, N. Parab, K. Fezzaa, W. Tan, L. Chen, T. Sun, "Bulk-Explosion-Induced Metal Spattering During Laser Processing". Phys Rev X, vol. 9, iss. 2, 2019.

[98] E. A. Holm, R. Cohn, N. Gao, A. R. Kitahara, T. P. Matson, B. Lei, S. R. Yarasi,

"Overview: Computer Vision and Machine Learning for Microstructural Characterization and Analysis." *Materials and Metallugrical Transactions A*, doi: 10.1007/s11661-020-06008-4, 2020.

[99] S. Van Der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, "Scikit-image: Image Processing in Python." *Peer J*, vol. 19, 2014.

[100] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. Van Kerkwijk, M. Brett, A. Haldane, J. F. Del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T. E. Oliphant, "Array Programming with NumPy." *Nature*, vol. 585, pp. 357-362, 2020.

[101] G. R. Bradski, A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly Media, 2018.

[102] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. Ter Haar Romeny, J. B. Zimmerman, K. Zuiderveld, "Adaptive Histogram Equalization and its Variations." Computer Vision Graphics and Image Processing, vol. 39, iss. 3, pp. 355-368, 1987.

[103] R. Fabbro, "Depth Dependence and Keyhole Stability at Threshold, for Different Laser Welding Regimes." *Applied Sciences*, vol. 10, iss. 4, 2020.

[104] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2009.

[105] Z. Gan, O. L. Kafka, N. Parab, C. Zhao, L. Fang, O. Heinonen, T. Sun, W. K. Liu, "Universal Scaling Laws of Keyhole Stability and Porosity in 3D Printing of Metals."

Nature Communications, vol. 12, iss. 1, 2021.

[106] T. Tullis, B. Albert, *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*, Morgan Kaufmann, 2013.

[107] J. Ye, S. A. Khairallah, A. M. Rubenchik, M. F. Crumb, G. Guss, J. Belak, M. J. Matthews, "Energy Coupling Mechanisms and Scaling Behavior Associated with Laser Powder Bed Fusion Additive Manufacturing." *Advanced Engineering Materials*, vol. 21, iss. 7, 2019.

[108] X. Zhang, Y. Zou, W. Shi, "Dilated Convolution Neural Network with Leaky ReLU for Environmental Sound Classification." *2017 22nd International Conference on Digital Signal Processing*, 2017.

[109] D. P. Kingma, J. Ba, "Adam: A Method for Stochastic Optimization." *3rd International Conference for Learning Representations*, 2015.

[110] L. Velasco, Y. Alanzi, E. McClellan, P. Ambrozewicz, N. Sato, T. Liu, M. Melnitchouk, P. Kuchera, Y. Li, "cFat-GAN: Conditional Simulation of Electron-Proton Scattering Events with Variate Beam Energies by a Feature Augmented and Transformed Generative Adersarial Network."*19th IEEE International Conference on Machine Learning and Applications*, pp. 372-375, 2020.

[111] J. Langr, V. Bok, *GANs in Action*, Manning Publications, 2019.

[112] S. Hyland, C. Esteban, G. Ratsch, "Real-Time (Medical) Time Series Generation with Recurrent Conditional GANs." arXiv:1706.02633, 2017.

[113] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, S. Savarese, "Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 658-667, 2019.

[114] D. P. Huttenlocher, G. A. Klanderman, W. J. Rucklidge, "Comparing Images Using the Hausdorff Distance*." IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, iss. 9, pp. 850-863, 1993.

[115] C. Zhao, W. Shi, Y. Dong, "A New Hausdorff Distance for Image Matching." *Pattern Recognition Letters*, vol. 26, iss. 5, pp. 581-586, 2005.

[116] A. Gretton, K. M. Borgwardt, M. Rasch, B. Scholkopf, A. Smola, "A Kernel Method for the Two-Sample Problem." *Journal of Machine Learning Research*, pp. 1-10, 2008.

[117] Y. Li, K. Swersky, R. Zemel, "Generative Moment Matching Networks." *arXiv:1505.03906,* 2015.

[118] D. Sutherland, H. Tung, H. Strathmann, S. De, A. Ramdas, A. Smola, A. Gretton, "Generative Models and Model Criticism Via Optimized Maximum Mean Discrepency." arxiv:1611.04488, 2016.

[119] G. K. Dziugaite, D. M. Roy, Z. Ghahramani, "Training Generative Neural Networks Via Maximum Mean Discrepancy Optimization." *arXiv:1505.03906,* 2015.

[120] Y. Yazici, C. Foo, S. Winkler, K. Yap, V. Chandrasekhar, "Empirical Analysis of Overfitting and Mode Drop in GAN Training." *IEEE International Conference on Image Processing*, 2020.

[121] W. Bounliniphone, E. Belilovsky, M. B. Blaschko, I. Antonoglou, A. Gretton, "A Test of Relative Similarity for Model Selection in Generative Models." *International Conference on Learning Representations*, 2016.

[122] R. Gao, F. Liu, J. Zhang, B. Han, T. Liu, G. Niu, M. Sugiyama, "Maximum Mean Discrepancy Test is Aware of Adversarial Attacks." *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[123] L. Mentaschi, G. Besio, F. Cassola, A. Mazzino. “Problems in RMSE-Based Model Validations.” Ocean Modeling, vol. 72, pp. 53-58, 2013.

[124] K. Borgwardt, A. Gretton, M. J. Rasch, H. Kriegel, B. Schölkopf, A. Smola, “Integrating Structured Biological Data by Kernel Maximum Mean Discrepancy.” *Bioinformatics*, vol. 22, iss. 14, pp. 49-57, 2006.

[125] X. Ding, Y. Wang, Z. Xu, W. Welch, Z.J. Wang, “CcGAN: Continuous Generative Adversarial Networks for Image Generation.” *ICLR 2021*, 2022.

[126] E. Mirkoohi, D. E. Seivers, H. Garmestani, S. Y. Liang, “Heat Source Modeling in Selective Laser Melting.” *Materials*, vol.12, pp. 1–18, 2019.

[127] S. Hocine, H. Van Swygenhoven, S. Van Petegem, “Verification of Selective Laser Melting Heat Source Models with Operand X-ray Diffraction Data.” *Additive Manufacturing*, vol. 37, 2021.

[128] L. Scime, J. Beuth, “Melt Pool Geometry and Morphology Variability for the Inconel 718 Alloy in a Laser Powder Bed Fusion Additive Manufacturing Process.” *Additive Manufacturing*, vol. 29, 100830, 2019.

[129] P. J. R. Prasad, O. J. Elle, F. Lindseth, F. Albregsten, R. P. Kumar, "Modifying U-Net for Small Dataset: A Simplified U-Net Version for Liver Parenchyma Segmentation." *Proceedings Volume 11597 of Medical Imaging 2021: Computer-Aided Diagnosis*, 2021.

[130] Q. Fang, Z. Tan, H. Li, S. Shen, S. Liu, C. Song, X. Zhou, Y. Yang, S. Wen, “In-situ Capture of Melt Pool Signature in Selective Laser Melting Using U-Net-Based Convolutional Neural Network.” *Journal of Manufacturing Process*, vol. 68, pp. 347–355, 2021.

[131] P. Isola, J. Y. Zhu, T. Zhou, A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks." *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition,* pp. 5967–5976, 2017.

[132] Y. Chang, Z. Liu, K. Lee, W. Hsu, "Free-Form Video Inpainting with 3D Gated Convolution and Temporal PatchGAN." *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV),* pp. 9066-9075, 2019.

[133] J. Liu, H. Liu, X. Zheng, J. Han, "Exploring Multi-Scale Deep Encoder-Decoder and PatchGAN for Perceptual Ultrasound Image Super-Resolution." *International Conference on Neural Computing for Advanced Applications,* pp. 47-59, 2020.

[134] O. Ronneberger, P. Fischer, T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", *MICCAI 2015: Medical Image Computing and Computer-Assisted Intervention*, pp. 234-241, 2015.

[135] O. Oktay, J. Schlemper, L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz, N. B. Glocker, D. Rueckert, "Attention U-Net: Learning Where to Look for the Pancreas." *1st Conference on Medical Imaging with Deep Learning*, 2018.

[136] A. Basu, R. Mondal, S. Bhowmik, R. Sarkar, "U-Net Versus Pix2Pix: A Comparative Study on Degraded Document Image Binarization." *Journal of Electronic Imaging,* vol.29, 2020.

[137] T. Iqbal, H. Ali, "Generative Adversarial Network for Medical Images (MI-GAN)." *Journal of Medical Systems,* vol. 42, 2018.

[138] D. Saxena, J. Cao, "Generative Adversarial Networks (GANs)." *ACM Computing Survey,* vol. 54, 2021.

[139] Rezende, D. J., Mohamed, S., & Wierstra, D. “Stochastic Backpropagation and Approximate Inference in Deep Generative Models.” *Proceedings of The 31st International Conference on Machine Learning*, vol. 32, pp. 1278–1286, 2014.

[140] Pagnoni, A., Liu, K., Li, S. “Conditional Variational Autoencoder for Neural Machine Translation.” https://arxiv.org/pdf/1812.04405.pdf, 2018.

[141] Yu, X., Zhang, X., Cao, Y., Xia, M. “VAEGAN: A Collaborative Filtering Framework Based on Adversarial Variational Autoencoders.” *Proceedings of the Twenty-Eigth International Joint Conference on Artificial Intelligence*, pp. 4206 – 4212, 2019.

[142] W. Carrer-Neto, M. L. Hernandez-Alcaraz, R. Valencia-Garcia, F. Garcia-Sanchez, "Social Knowledge-Based Recommender System: Application to the Movies Domain," *Expert Systems with Applications*, vol. 39, pp. 10990-11000, 2012.

[143] R. M. Bell, Y. Koren, C. Volinsky, “All Together Now: A Perspective on the Netflix Prize.” *Chance*, vol. 23, iss. 1,2010.

[144] M. Vermna, K. K. Shukla, “A New Accelerated Proximal Gradient Technique for Regularized Multitask Learning Framework.” *Pattern Recognition Letters*, vol. 95, pp. 98-103, 2017.

[145] K. Toh, S. Yun, “An Accelerated Proximal Gradient Algorithm for Nuclear Norm Regularized Least Squares Problems.” *Pacific Journal of Optimization*, vol. 6, iss. 3, 2010.

VITA

Andy Ramlatchan
NASA Langley Research Center
Katherine G. Johnson Computational Research Center
1 Lindbergh Way, Hampton, VA 23681
757-864-7698
Andy.ramlatchan@nasa.gov

Education:

- PhD Candidate in Computer Science, Old Dominion University, Norfolk, VA 23529 (August 2022)
- Graduate student in Computer Science, Northwestern University, Evansville, IL 60208 (August 2012 – August 2017)
- Master of Business Administration, Averett University, Danville, VA 24541 (December 2010)
- Bachelor of Science, Old Dominion University, Norfolk, VA 23529 (May 2008)

Professional Experience:

- Statistical Engineer, NASA Langley Research Center, Systems Engineering and Engineering Methods Branch (2021 – Present)
- Data Scientist, NASA Langley Research Center, Office of the Chief Information Officer (2017 – 2021)
- Cyber Researcher, U.S. Department of Justice (2016 – 2017)
- Director of Analytics and Data Management, Patient Advocate Foundation, (2011 – 2016)

Awards:

- NASA at Work Innovation Award 2019
- Best Paper 2020 Award for *Big Data Mining and Analytics* journal for "A Survey of Matrix Completion Methods for Recommendation Systems

Journal Publications

- "Time-resolved geometric feature tracking elucidates laser-induced keyhole dynamics," *Integrating Materials and Manufacturing Innovation*, vol. 10, pp. 677-688.
- "Image synthesis using conditional GANs for selective laser melting additive manufacturing," submitted to 2022 IEEE International Joint Conference on Neural Networks.
- "A comparison of generative adversarial networks to convolutional neural networks for single track melt pool segmentation as evaluated by Hausdorff distance," submitted to Additive Manufacturing Letters.