Manufacturability Analysis of Laser Powder Bed Fusion using Machine Learning

by

Daniyal Khan

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master

of

Computing and Information Systems

YOUNGSTOWN STATE UNIVERSITY

December, 2023

Manufacturability Analysis of Laser Powder Bed Fusion using Machine Learning

Daniyal Khan

Signature:

Daniyal Khan, Student Date

Approvals:

Dr Alina Lazar, Thesis Advisor Date

Dr. John R. Sullins, Committee Member Date

Dr. Hunter Taylor, Committee Member Date

Dr. Salvatore A. Sanders, Dean of Graduate Studies Date

ABSTRACT

Additive Manufacturing (AM), particularly LASER Powder Bed Fusion (LPBF), has gained prominence for its flexibility and precision in handling complex metal structures. However, optimizing L-PBF for intricate designs involves analyzing over 130 process parameters, leading to prolonged duration and increased costs. This thesis proposes a novel approach by harnessing statistical and machine learning algorithms to predict manufacturability issues before the printing process. By performing a comparative analysis of the intended design with the machine produced result, the study introduces two machine learning and one artificial neural network (ANN) algorithm to forecast the printability of new designs accurately. This innovative method aims to reduce or eliminate the need for iterative printing, reducing productivity costs and optimizing the LPBF additive manufacturing process.

Acknowledgments

I would first like to thank my thesis advisor, Dr. Alina Lazar of the Department of Computer Science and Information Systems at Youngstown State University. The door to Prof. Lazar's office was always open whenever I encountered a problem or was concerned about my research work or writing. She consistently allowed this thesis to be my own work while guiding me in the right direction whenever I needed it.

I would also like to thank the committee members Dr. John Sullins and Dr. Hunter Taylor for their precious time and advice during my thesis process.

I wish to extend my heartfelt appreciation to the Department of Computer Science and Information Systems and the College of Graduate Studies for their financial support throughout my graduate studies.

Finally, I want to convey my deep appreciation to my classmates and friends who consistently offered invaluable technical assistance, support, and unwavering encouragement during my academic journey, including the research and writing phases of this thesis. Their contributions were indispensable to the completion of this achievement, and I am truly grateful. Thank you.

Table of Contents

List of Figures

List of Tables

1 Introduction

In the past few decades, Additive Manufacturing (AM) has been one of the increasingly important metal processing methods on account of its advantageous high flexibility, low material wastage, short lead time, and complex geometry handling. Several researches have shown how AM is efficiently capable of working with complex structures using a vast variety of materials such as polymers [2], ceramics [3], and metals [4]. LASER Powder Bed Fusion (L-PBF), also known as Selective LASER Melting (SLM) or Direct Metal LASER Melting (DMLM), is one of the most efficient AM methods. L-PBF provides the capability of near net shape (NNS) manufacturing, which aims for fabricating with close in size and shape, with highest accuracy among other AM techniques. However, with the increase in complexity of the design model, L-PBF becomes extremely challenging to develop the product where more than 130 process parameters such as LASER power, scan speed, hatching-spacing, and hatching angles need to be analyzed with an intent of optimization.

Analyzing the L-PBF process has been an exponentially complex task, with the primary concern being the enhancement of build quality. Optimization of the AM process requires analysis of the design, the print results, and the modification of the design and associated parameters to mitigate the problems found in the analysis. Currently, used optimization methods include multiple iterations of the printing process. After each printing process, a post-build analysis is conducted on the produced object to identify any defects. Subsequently, the design is adjusted or the printing parameters are fine-tuned based on the findings of this analysis. While this strategy does not ensure an optimized design, the comparatively slow iterative printing of L-PBF leads to increased productivity costs and prolonged duration for analysis and optimization. Hence, expediting the optimization process requires eliminating the iterative printing of the artifact.

This thesis work attempts to coin a method by leveraging the capability of statistical and machine learning algorithms to analyze the manufacturability of the design subjected to be printed using L-PBF, and predict what parts of the design are more prone to cause defects in the final build. Several research studies are underway to formulate comprehensive machine learning algorithms, emphasizing the examination of the correlation between design, process parameters, and the quality of the manufactured parts. We propose a method for comparison of the design, which is subject to be printed with what the machine prints. Then, we use the result of the comparison to architect a machine learning model that can predict if a new design is printable by the machine or not. This thesis proposes two different machine learning and one artificial neural network (ANN) algorithm that can perform this task with high accuracy.

1.1 Laser Powder Bed Fusion

Metal additive manufacturing involves forming objects from 3D computer-aided design (CAD) model data by joining materials, typically in a layer-by-layer fashion, unlike subtractive manufacturing technologies. Using slicing software, the 3D CAD model files are sliced into two-dimensional (2D) layers. The surface roughness of the formed object depends on the selected thickness of the layer, or thinning the layers smooths the surface. There are several methods used in metal additive manufacturing, which include material jetting, material droplet printing, sheet lamination, powder bed fusion, material extrusion, binder jetting, and directed energy

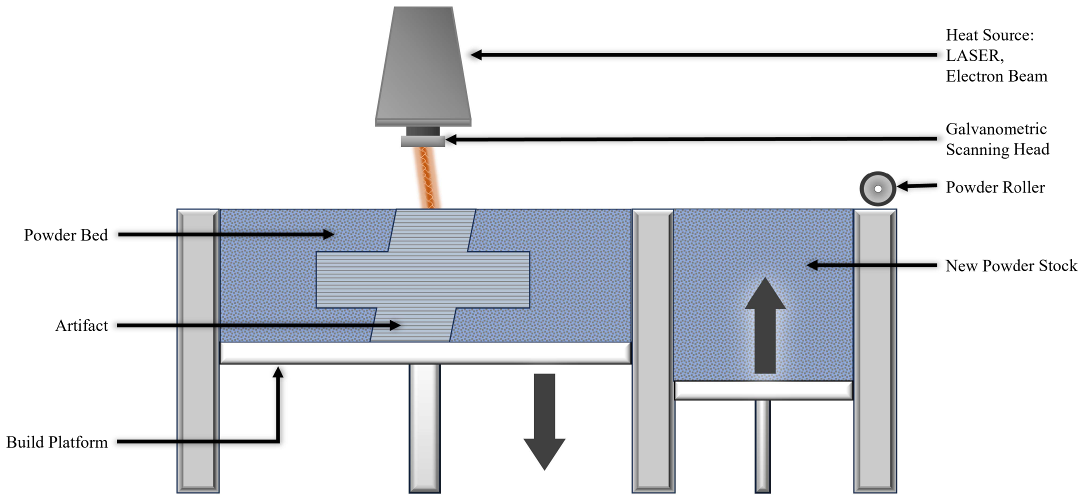


Figure 1: Representational architecture Laser Powder Bed Fusion System

deposition[5], among which laser bed fusion is most commonly used in current printing systems.

In powder bed fusion, a layer of metal powder is evenly spread on the build plate, and a heat source is employed to fuse the metal powder at locations specified by the design, facilitating the formation of the desired geometry. After one layer is completed, a new thin layer of powder is applied on top of the completed layer, and the same process is repeated until the complete 3D geometry is formed. In laser powder bed fusion (L-PBF), a laser beam is used as a heat source to fuse the metal powder. L-PBF additive manufacturing systems are setup as shown in Figure 1. The Galvanometric Scanning Head (GSH) controls the incidence angle of the laser beam and directs it to the locations specified by the design on the build platform. Once one layer is completely fused, a new layer is spread evenly by the powder roller, which shifts the new powder from the powder stock to the build plate.

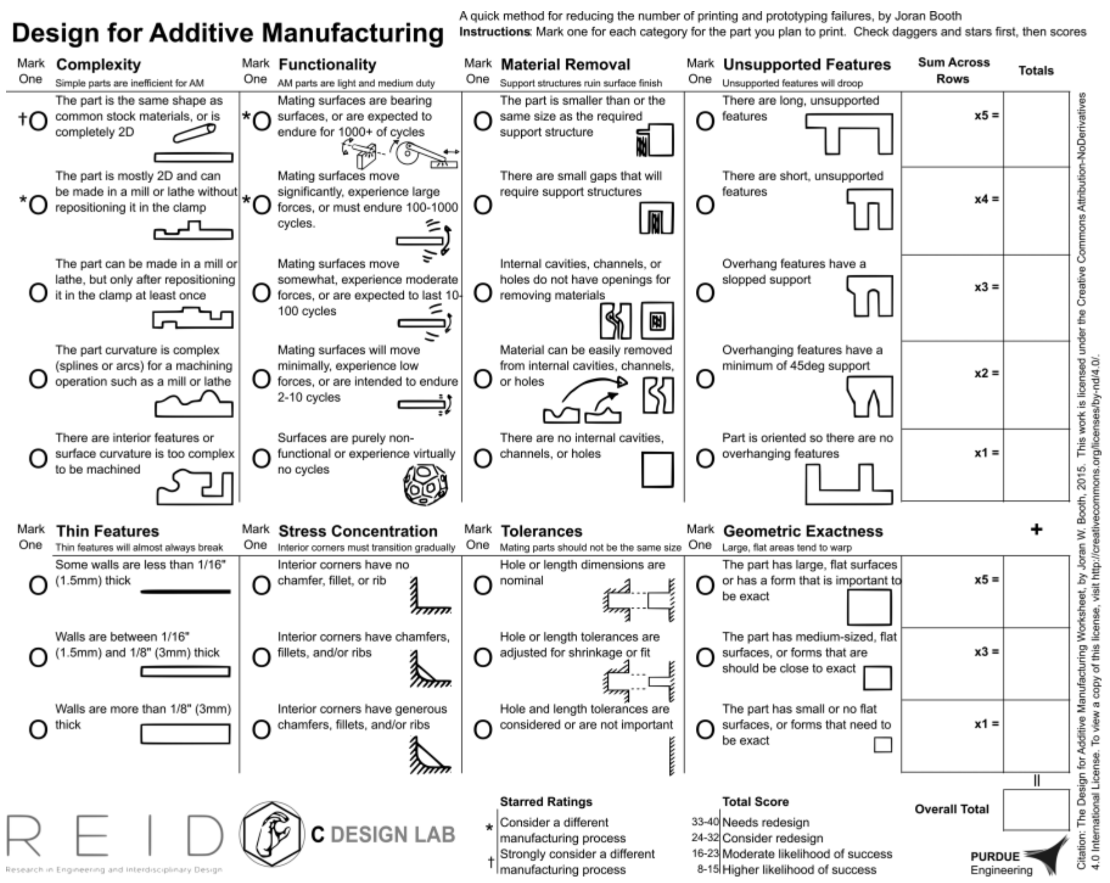
2 Related Works

There have been several attempts to devise an approach for pre-build analysis of manufacturability using the LASER Powder Bed Fusion (LPBF) process, which included mathematical, statistical, and machine learning (ML) methods. The most conventional approach is to use a design worksheet Figure 2 to evaluate the design [1]. This approach needed the user to consider manufacturability factors for LPBF, including but not limited to a minimum thickness, the most optimal printing orientation, the maximum number of overhang angles without supports, etc. In this method, before the design can be fabricated, the users need to verify the design manually based on the measures provided. The method is helpful in designs with simple structures, but evaluating designs with complicated structures is very difficult as this approach is incapable of providing a way to determine how different printing strategies and processes will affect the final quality of the artifact.

The second approach applies Bayesian Networks (BNs) to the knowledge management system to categorize the AM knowledge into different correlated layers. It leverages the BNs capacity of inferring under uncertainty. It generates its inference based on principles of conditional probability and Bayes' Theorem [6]:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

here A and B represent events with the condition that P(B) is not equal to zero. It makes it possible to repeatedly update the probabilities of interconnected nodes whenever new evidence is received. This method, then, is applied to find the probability of the state of certain variables affecting the build quality. For example, the



Design for Additive Manufacturing

A quick method for reducing the number of printing and prototyping failures, by Joran Booth

Instructions: Mark one for each category for the part you plan to print. Check daggers and stars first, then scores

Mark One	**Complexity** Simple parts are inefficient for AM	Mark One	**Functionality** AM parts are light and medium duty	Mark One	**Material Removal** Support structures ruin surface finish	Mark One	**Unsupported Features** Unsupported features will droop	Sum Across Rows	Totals
†O	The part is the same shape as common stock materials, or is completely 2D	*O	Mating surfaces are bearing surfaces, or are expected to endure for 1000+ of cycles	O	The part is smaller than or the same size as the required support structure	O	There are long, unsupported features	x5 =	
*O	The part is mostly 2D and can be made in a mill or lathe without repositioning it in the clamp	*O	Mating surfaces move significantly, experience large forces, or must endure 100-1000 cycles.	O	There are small gaps that will require support structures	O	There are short, unsupported features	x4 =	
O	The part can be made in a mill or lathe, but only after repositioning it in the clamp at least once	O	Mating surfaces move somewhat, experience moderate forces, or are expected to last 10-100 cycles	O	Internal cavities, channels, or holes do not have openings for removing materials	O	Overhang features have a slopped support	x3 =	
O	The part curvature is complex (splines or arcs) for a machining operation such as a mill or lathe	O	Mating surfaces will move minimally, experience low forces, or are intended to endure 2-10 cycles	O	Material can be easily removed from internal cavities, channels, or holes	O	Overhanging features have a minimum of 45deg support	x2 =	
O	There are interior features or surface curvature is too complex to be machined	O	Surfaces are purely non-functional or experience virtually no cycles	O	There are no internal cavities, channels, or holes	O	Part is oriented so there are no overhanging features	x1 =	

Mark One	**Thin Features** Thin features will almost always break	Mark One	**Stress Concentration** Interior corners must transition gradually	Mark One	**Tolerances** Mating parts should not be the same size	Mark One	**Geometric Exactness** Large, flat areas tend to warp		+
O	Some walls are less than 1/16" (1.5mm) thick	O	Interior corners have no chamfer, fillet, or rib	O	Hole or length dimensions are nominal	O	The part has large, flat surfaces or has a form that is important to be exact	x5 =	
O	Walls are between 1/16" (1.5mm) and 1/8" (3mm) thick	O	Interior corners have chamfers, fillets, and/or ribs	O	Hole or length tolerances are adjusted for shrinkage or fit	O	The part has medium-sized, flat surfaces, or forms that are should be close to exact	x3 =	
O	Walls are more than 1/8" (3mm) thick	O	Interior corners have generous chamfers, fillets, and/or ribs	O	Hole and length tolerances are considered or are not important	O	The part has small or no flat surfaces, or forms that need to be exact	x1 =	

=

Overall Total

Starred Ratings

*	Consider a different manufacturing process
†	Strongly consider a different manufacturing process

Total Score

33-40	Needs redesign
24-32	Consider redesign
16-23	Moderate likelihood of success
8-15	Higher likelihood of success

REID Research in Engineering and Interdisciplinary Design

C DESIGN LAB

PURDUE Engineering

Citation: The Design for Additive Manufacturing Worksheet, by Joran W. Booth, 2015. This work is licensed under the Creative Commons Attribution-NoDerivatives 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nd/4.0/.

Figure 2: Additive Manufacturing Worksheet proposed by Booth's Group [1]

probability of the surface inclination to the Z-axis being below or above a given angle can be determined using layer thickness. These probabilities help categorize layers, which in turn helps to determine the best fabrication AM process for the given part. While the method is indeed declarative and descriptive, it lacks the capability to handle design complexities on various LPBF machine types. Furthermore, each node in the Bayesian Network is discretized, but since many AM parameters are continuous, this leads to a significant loss of information. Consequently, as reported by the authors [6], this imposes a substantial accuracy limitation on the predictions made by the Bayesian Network.

As a subset of Artificial Intelligence, machine learning has been a feasible option for certain tasks that can be handled without human intervention and also some computationally intensive tasks where the number of factors affecting the process are too high to be handled by humans [7]. It has generated a major domain for developing machine learning in AM for pre-process parameter optimization, manufacturability prediction, *in-situ* monitoring, and post-process analysis, where quality control is a critical aspect. In the area of In-situ or real-time monitoring [8, 9, 10] and post-process analysis, there have been several researches [11] which generated efficient results in finding the major defects in the build.

However, in pre-process analysis or manufacturability, very little work is observed. In this problem [12] and [13] are major works which mainly focused on the use of Hybrid Machine Learning algorithms to evaluate the design to predict if the design is vulnerable. Zhang et al. [12] proposed a method that combines a voxel-based Convolutional Neural Network (CNN) aimed for analysis of the design and a Neural Network (NN) model, which aims at the inclusion of input variable design, material, and printing process in 3D objects, texts, and values data types. Zhang et. al. [13], on the other hand, omits the use of voxelization of 3D models for the design analysis. Instead, they propose the use of sparse metrics for the shape representation of the design. In both of the works, LPBF machine type differences were not considered in the analysis process, making it incompatible for some machines while working with others.

This thesis introduces a deep learning approach for manufacturability analysis. Our proposal centers on the utilization of a common layer interface (CLI) representation of the design, which consists of a sequence of coordinates that command the movement of the laser head on the build plate for the fabrication process. This CLI dataset is labeled by the use of associated Point Cloud Data(PCD), recorded during the fabrication of the artifact. For each of the CLI layers, there is one associated PCD layer that assists in labeling coordinate sequences on the layer level. This labeled data set is then used to train the neural network. The data-set section elaborates more on the structure of the CLI file and recording of PCD. The method is compatible with different machine types as the model can be trained specifically for a machine type using PCD generated from the machine.

3 Methods

3.1 Multi-Layer Perceptron Neural Network

Artificial Neural Networks (ANNs), due to their ability to imitate the human brain and robustness to handle complex problems, have gained enormous attention in the domain of artificial intelligence. Their application lies in a wide range of domains, such as pattern recognition, natural language processing, forecasting, and prediction. Multi-layer perceptron (MLP) is one of the most popular types of artificial neural network. A single perceptron model works as a threshold function that learns a binary classifying criterion. Its operation involves mapping the input variable x to an output variable $\hat{y} : \hat{y} = f(x)$. But single-layer perceptron is limited only to data points that are linearly separable and where the problem is to separate two classes by computation of a dividing hyperplane. But when the data is not linearly separable, then it becomes necessary to increase the number of perceptrons. However, when the data points are highly variable, the single-layer perceptron model fails to classify them. Therefore, a multi-layered perceptron is required to tackle such variables.

An MLP neural network comprises layers of units where each layer contains several perceptrons or nodes. The number of layers in MLP is required to be at least three, which consists of an input layer, one output layer, and one or more hidden layers that connect the input layer from the output layer as illustrated in Figure 3. Introducing non-linearity is important so that the model can handle complex relationships that lie in a highly variable data set. The use of the activation function helps to equip the model with a non-linear behavior[14]. Each node of the hidden layer takes some of the input variables and applies the activation function on it, and the resultant(activated output) is passed to the next layer. A general computation at a node of a hidden layer is expressed as in the following equation:

$$\hat{y}_i = Activation \left\{ \sum_{i-1}^{n} (W_i \times x_i + b) \right\} \tag{2}$$

Sigmoid and hyperbolic tangent (tanh) are two of the widely used activation functions[15]. But the choice of activation function depends completely on the type of task.

3.2 Adaptive Boosted Decision Tree

Introduced in the 1960s, decision tree [16] has proved itself to be one of the most effective supervised learning methods for classification and regression tasks. Their application lies in a wide variety of domains because of their robustness, even in the presence of missing values. The primary objective of a decision tree is the implementation of a model that can make predictions for the targeted variable. This is achieved by inferring decision rules from the features of the dataset. Figure 4

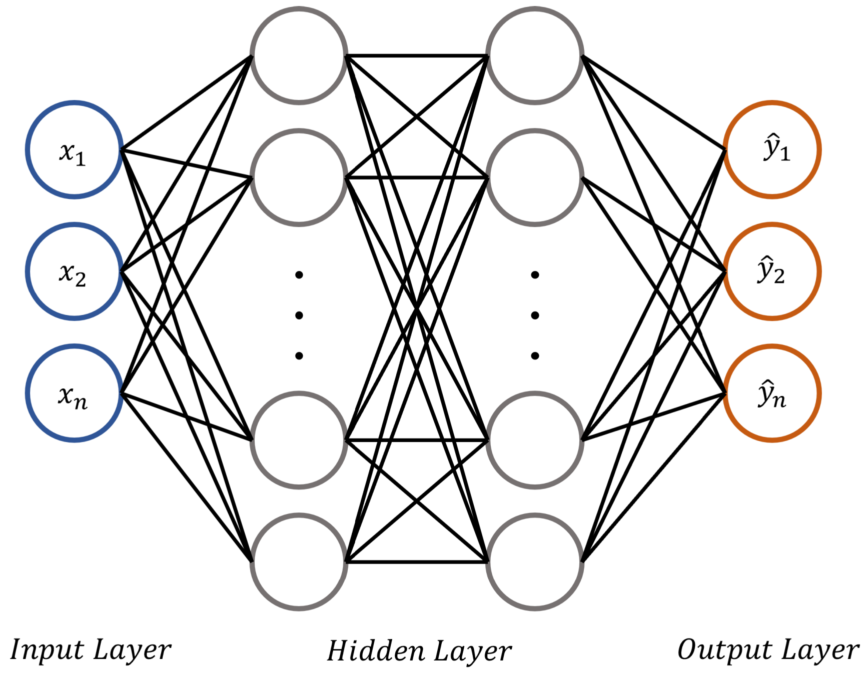


Figure 3: Architecture of a MLP Neural Network

illustrates a simple decision tree model that has a binary target variable. The root node or topmost node of the tree asks a most important question ($X > 5$) and divides the tree. Based on the answer, a child node is selected. Furthermore, the child node, on the second level of the tree, asks another question ($X < 2$), and this process is repeated until a leaf node (represented by a circle) is reached which categorizes (R1, R2, R3, R4, or R5) the instance.

AdaBoost[17] or adaptive boosting, is an ensemble learning method. In the proposed work, the AdaBoost Algorithm is used to generate a robust decision tree classifier that attains precise classification. AdaBoost, due to its accuracy, interpretability, and scalability, has been among the most popular boosting methods for classification tasks. AdaBoost trains multiple weak learners or stumps (a small decision tree having only one level in it, i.e., two leaves) consecutively. The new learners

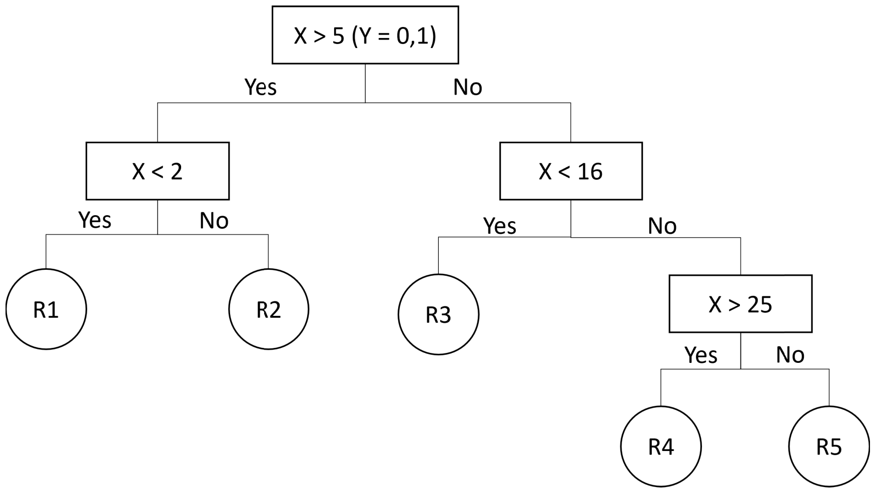


Figure 4: A Simple Decision Tree Model

focus on the misinterpreted or misclassified samples from the previous learners; as a result, a stronger learner than the previous one is achieved. In this process, previously misclassified samples are trained by combining them with new samples, and their weights are updated so that the new classifier focuses on more complex scenarios, which gradually develops a stronger classifier. These weak learners are also called estimators and, as hyper-parameters of the algorithm, the number of estimators and learning rate can shrink or expand the contribution of each weak learner in the training process[18].

3.3 XGBoost

XGBoost stands for Extreme Gradient Boost. It is an alternative decision tree ensemble method that extends the gradient boost algorithm[19] to create a more scalable method. The Gradient Boost algorithm creates weak learners, which can have more than two leaves in it, as opposed to AdaBoost, where weak learners are stumps (typically decision trees having only two leaves). In the traditional gradient boosting algorithm for binary problems, the objective function (Eq. 3) is primarily training loss that measures the difference between the predicted and actual label.

$$ObjectiveFunction = \sum_{i-1}^{n} LogLoss(y_i, \hat{y}_i) \tag{3}$$

In XGBoost, the objective function incorporates training loss and additional regularization terms.

$$ObjectiveFunction = \sum_{i-1}^{n} LogLoss(y_i, \hat{y}_i) + \sum_{k-1}^{K} \Omega(f_k) \tag{4}$$

Where n is the number of samples, and K is the number of trees in the ensemble. The regularization term $\Omega(f_k)$ gives more control over the complexity of the weak learners and also helps in the prevention of over-fitting of the model. The regularization function is defined as follows:

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{j-1}^{T} {\omega_j}^2 \tag{5}$$

Where T is the number of leaves in the tree, ω_j is the score associated with j^{th} leaf. And γ and λ are regularization hyper-parameters, the value of which is directly proportional to the complexity of the tree. A more comprehensive explanation of the algorithm is not in the scope of this thesis.

Feature	Infill	Contour	Downfill	Down Contour
Speed(mm/sec)	1000	1000	1000	1000
Power(watt)	250	300	200	150

Table 1: Criterion set for Printing of Quality Test Artifact

4 Datasets

The proposed method uses data collected from the standard Quality Test Artifact (QTA)[20], which was printed at Tailored Alloys in association with W.M. KECK CENTER FOR 3D INNOVATION. The QTA was sliced into 846 layers of 5-millimeter thickness. This dataset included two types of data files. The first type is called the command data file, as it is generated after the design phase and is fed into the LPBF machine for the fabrication of the artifact. Whereas, the second type of data file is called the actual data file. The actual data file is generated while the artifact is being printed in the machine. The QTA was printed using the criterion mentioned in Table 1. Table 1 shows only the default power and speed set for the printing process in which the speed remains constant, whereas the power can vary with each vector.

An elaborated description of these datasets is given in the following subsections.

4.1 Commanded Data

Commanded data is a set of Common Layer Interface files that are generated from the AutoDesk Netfabb slicing procedure. Once the designing phase of the artifact is complete, the design can be divided into layers, where the number of layers depends on the required surface roughness of the artifact. The process of dividing the design into layers is called slicing, and it generates a set of Common Layer Interface files as shown in Figure 6. This set of files includes different types (the type represents different features of the layer as shown in Figure 5) of CLI files which commands the Galvanometeric Scanning Head (GSH)(see Figure 1) to direct the LASER Beam on the build plate. Following is the description of relevant fields that a CLI file includes:

1. LAYER - Layer Number.
2. POWERS - Commanded LASER powers for vectors in the Layer Number defined above.
3. HATCHES
   (a) Number of Vectors in the Layer (N).
   (b) Start and end coordinates ((Xs, Ys), (Xe, Ye)) of N vector.

Each layer is divided into different CLI files which may include commands for separate features of the layer. The sequence in which the machine reads commands for each layer from different files can be controlled at the slicing phase or in the machine itself. This sequence is recorded in the model-section (metadata) file.

4.2 Actual Data

Actual Data is collected during the fabrication of the artifact. It records the movements of the GSH. A GSH contains a set of reflective mirrors that direct the LASER beam in the intended direction on the build plate. One of the mirrors controls the x-axes, and another controls the y-axes, whereas the focal point of the LASER beam provides the z-axes. These records of the GSH movements can be used to

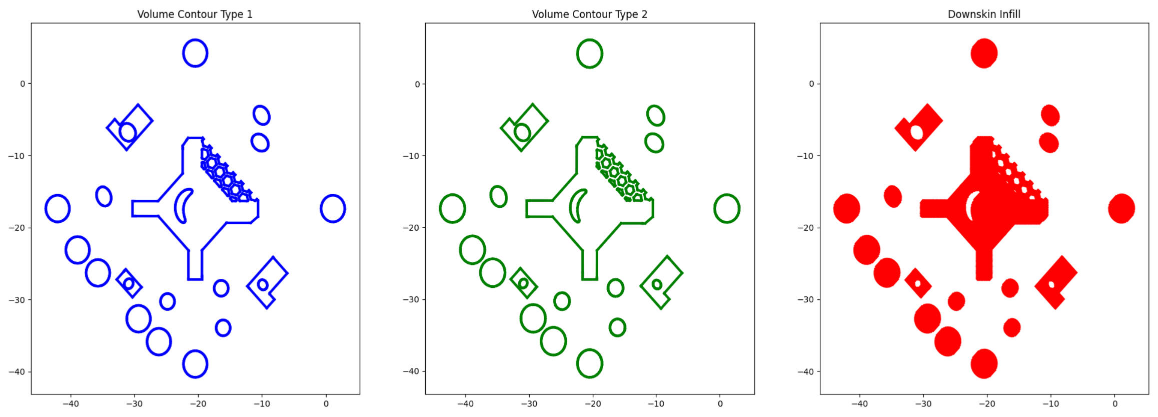


Figure 5: CLI files representing different features of the Layer 30.0

simulate the position on the build plate where the LASER defuses the powder(see Figure 1) and can be used to analyze if the LASER followed the commanded path efficiently or was deviated. This data is stored in a Point Cloud Data(PCD) file format. For each layer, this file records the time, GMS position, Power of the LASER, and state of the LASER that tells whether the LASER was in ON state or OFF state, and sensor values, if any, used in the machine.

4.3 Data Pre-processing

The raw data contains a set of files that are required to be processed, cleaned, and converted into a format that matches the proposed ML model criterion. The pre-processing of the files includes steps as follows:

4.3.1 Common Layer Interface

As mentioned in the previous subsection, in a CLI file, each layer is represented as a set of vectors(see Figure 6) called HATCHES that contains N vectors, where N represents the number of vectors in each layer. The vectors, which are stored as lines

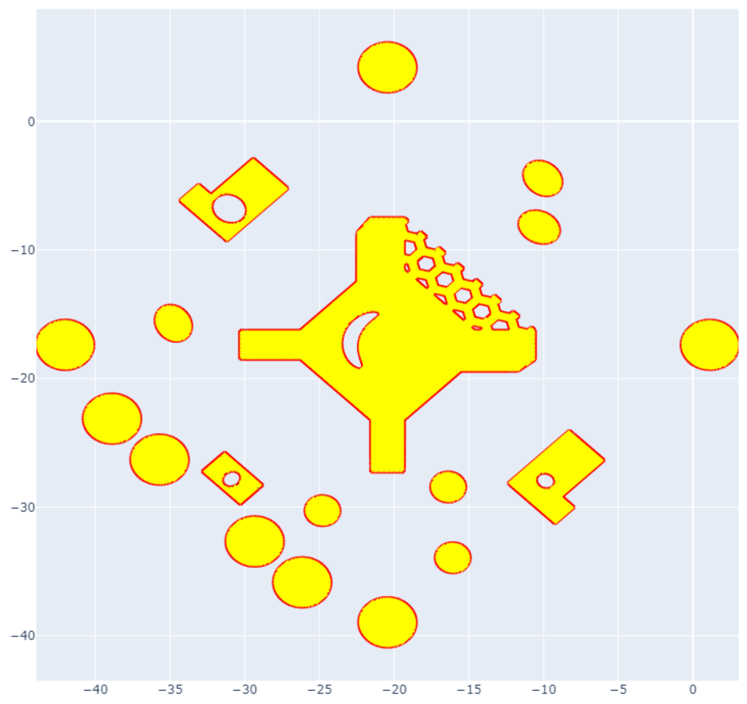


Figure 6: Visualization of Layer 30.0 from the CLI File

of string in the CLI file, are converted into Comma Separated Values (CSV) Format where a vector is represented as rows having four values as start coordinate (Xs, Ys) and end coordinate (Xe, Ye). The vectors in each HATCH can be of two types (see Figure 7) :

1. Single Vector
2. Polyline Vector

A single vector, as the name suggests, is an individual vector and is not directly connected with another vector by either of its edges. The Polyline vector is a set of consecutive vectors that are connected with each other by either one of the edges of

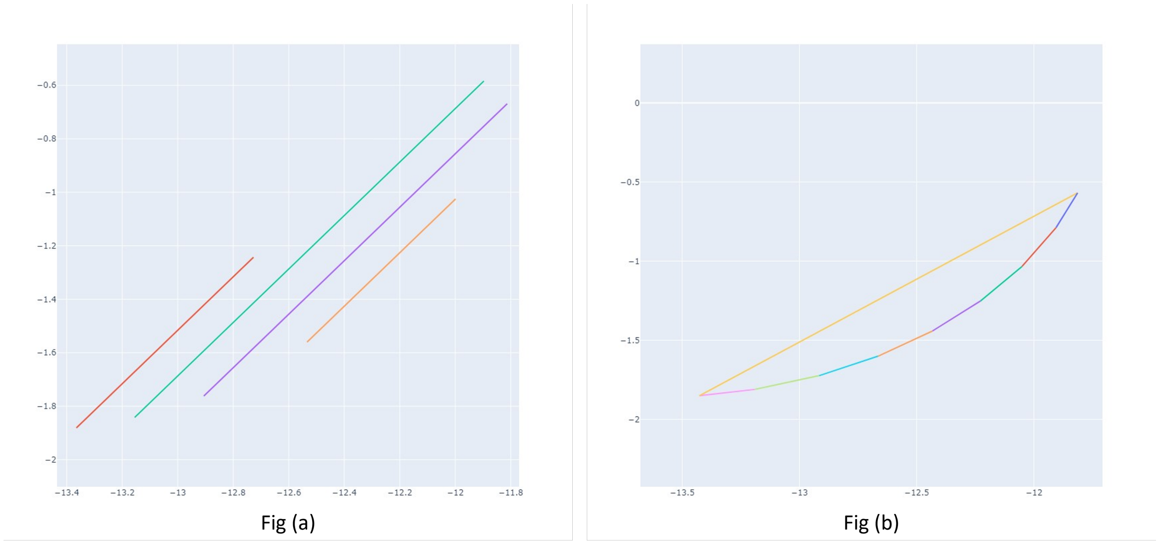


Figure 7: Visualization of a slice of one layer of the Command File (CLI) (a) Single Vectors, (b) A Polylines

another vector. The number of vectors in a Polyline can be as few as two vectors. The pre-processing of the CLI files generates a set of CSV files where each CSV file corresponds to a layer of the artifact.

4.3.2 Point Cloud Data

Point Cloud Data is stored in the .pcd file format in which the top few lines of the file include meta information about the machine and printing parameters, which are required during the analysis of the data on application software provided by the machine manufacturer. The metadata for the proposed purpose includes axes offsets and scaling factors, as the values given in the PCD file are stored in a format understandable by the manufacturer software, so the offsets and scaling factors are used to convert these values into standard units (millimeters). The machine generates one file for one layer of the artifact. Each PCD file records the coordinates of the

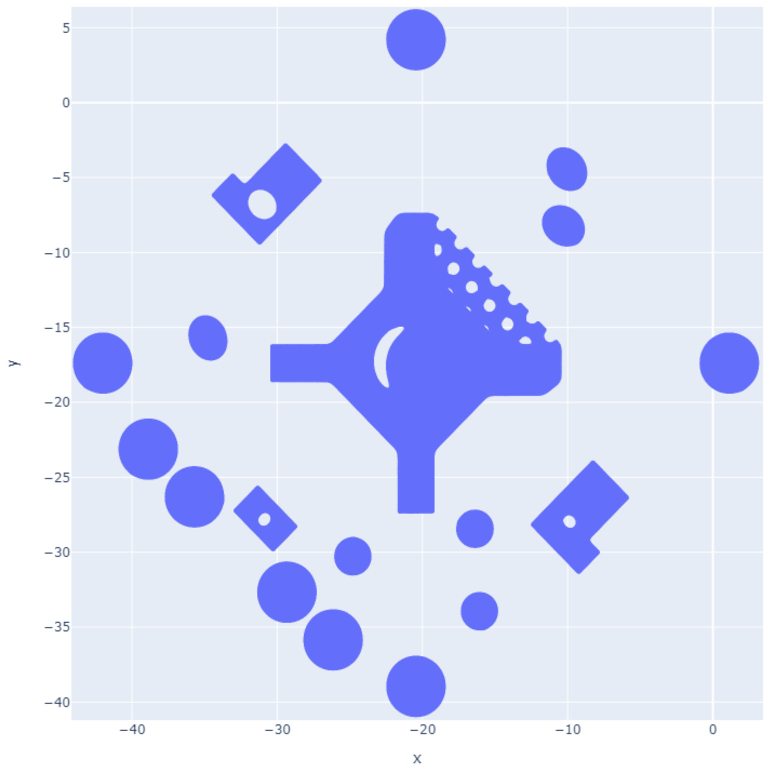


Figure 8: Visualization of Actual data-points of Layer 30.0 of QTA UTEP 43.1

LASER position on the build plate every 10 microseconds. The processed PCD is saved in Comma Separated Values (CSV) format. Visualization of one layer of the PCD File is shown in Figure 8.

4.4 Data Labeling

Proposed methods are supervised machine learning algorithms that require a labeled data set to train the model. To label the input dataset, the commanded dataset (CLI) is compared with the actual dataset (PCD). One vector command

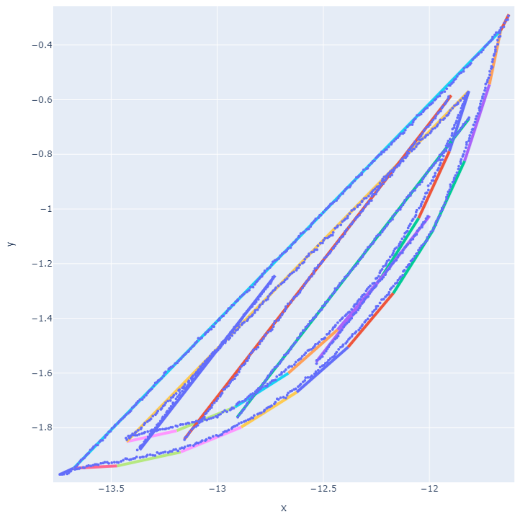


Figure 9: A slice of Actual data points on top of Commanded vectors

$((X_{scli}, Y_{scli}), (X_{ecli}, Y_{ecli}))$ is created by a set of consecutive actual data points(X_i, Y_i), where i represents number of points in the set. To find the actual data points associated with one vector, an exhaustive comparison of vector coordinates with the actual data point coordinates is performed. This comparison includes a search of the points in the PCD file that are associated with the commanded vectors. A magnified plot of the actual data points on top of the commanded vectors can be seen in Figure 9. The search process starts with finding the closest points in the PCD file to the start and end coordinates of the vector. All the actual data points between the start and end coordinates are associated with the subjected vector. Figure 10 is a representative example of a CLI vector $((X_{scli}, Y_{scli}), (X_{ecli}, Y_{ecli}))$ overlapped on actual data points (X_i, Y_i). To identify if the CLI vector is deviated, the perpendicular distances (d_i) of the actual data points are calculated. If the average of distances is greater than a deviation threshold distance(τ), then the vector is labeled as deviated. The distances

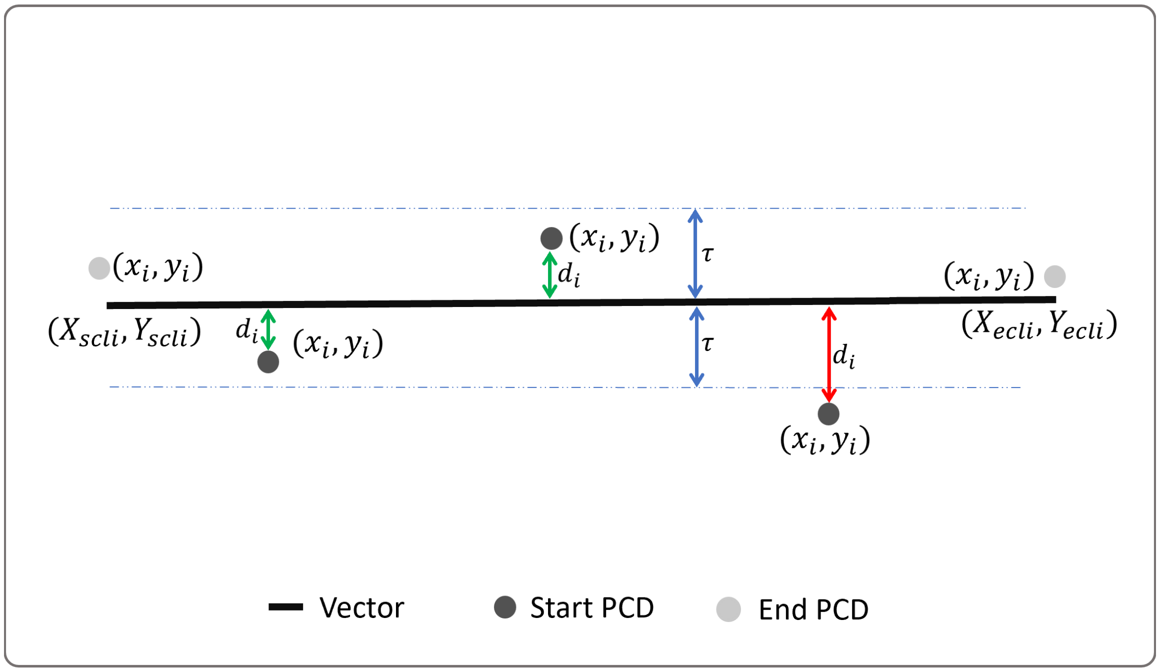


Figure 10: Representation of Actual Data-points Associated with One Commanded Vector

of the actual data points from the CLI vectors are calculated using the Cartesian distance calculation equation, which is as follows.

$$d_i = \frac{|(x_i - X_{scli})(Y_{ecli} - Y_{scli}) - (y_i - Y_{scli})(X_{ecli} - X_{scli})|}{\sqrt{(X_{ecli} - X_{scli})^2 + (Y_{ecli} - Y_{scli})^2}} \tag{6}$$

For experimentation of the proposed methods, deviation distance threshold τ for the used data-set has been set to 25μ. The labeled dataset generated using the aforementioned labeling method showed that over 30% of the points deviated from the given commands. The description of the independent variable in the final labeled dataset is presented in Table 2

Features	**Description**
layer_height	Height of the Layer in millimeter
$((X_{scli}, Y_{scli}), (X_{ecli}, Y_{ecli}))$	Start and End Coordinates of the vector
c_power	Commanded Power for the vector
c_speed	Commanded Speed for the feature of the layer
vector	The set number of the vector
deviation	The label associated with each vector (0 or 1)

Table 2: Description of the Data-set after the Labeling process

5 Experiments and Results

We tested the three algorithms mentioned in Section 3 on the same dataset to determine the accuracy of each algorithm. To achieve a good trade-off between training time and the accuracy of the final model, all the algorithms were trained and compared on different sets of hyperparameters. The training dataset has 481,847 vectors, among which 152,565 are labeled as 'Deviated (0)' and 329,282 are labeled as 'Not Deviated (1)'. The size of the train, validation, and test datasets are kept constant for all the algorithms. The train, validation, and test datasets are divided in the ratio of 0.8, 0.1, and 0.1 respectively. The accuracy of each of the tests was calculated using the following equation:

$$Accuracy(\%) = \frac{P_{correct}}{P_{correct} + P_{incorrect}} \times 100 \tag{7}$$

Where $P_{correct}$, and $P_{incorrect}$ represent correct and wrong predictions respectively. The experiments were performed on Simple Linux Utility for Resource Management (SLURM) provided by Ohio Super Computer (OSC)[21] on Pitzer Cluster[22].

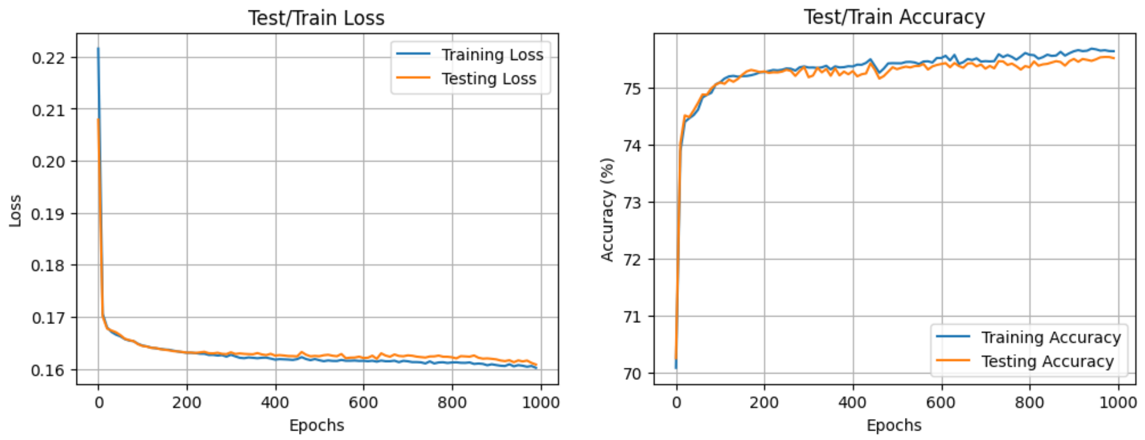


Figure 11: Convergence of Loss and Accuracy in MLP neural network model with the number of epochs

5.1 Multi-Layer Perceptron

The multi-layer perceptron model contains three hidden layers. The input layer size is 8, while the hidden layers are 16, 32, and 16 respectively. ReLU[23] (Rectified Linear Unit) activation function is used by the hidden layers where, as a sigmoid function is used at the output layer of the network. The Multi-Layer Perceptron has been tested on four training batch sizes. While loss and test accuracies did not change significantly, the convergence of the model was slower or faster depending on the batch size used. The learning rate was kept constant at 0.01 for all the tests, as it was the most optimal for the model structure used. The most optimal training batch size was 10,000. The results are shown in the figure 11. The maximum accuracy achieved by the model was 0.756, and the recorded convergence time was 108.48 sec.

5.2 Adaptive Boosting for Decision Tree

The AdaBoost algorithm was trained on different sets of parameters to find the most optimal hyperparameters. In the AdaBoost algorithm, the parameters that can affect the model's accuracy and training time are the learning rate and the depth of the tree being used as a weak learner. For experimentation, the model was tested with four learning rates: 0.1, 0.2, 0.3, and 0.4. These learning rates were tested with nine tree depths, ranging from 2 to 10. The AdaBoost algorithm produced accuracy similar to that of the multi-layer perceptron, with the training time of the model slightly lower. The maximum accuracy (0.7608) generated by the model was at a learning rate of 0.3 and a tree depth of 7. The recorded model convergence time was 174.14 seconds, higher than MLP, but AdaBoost produced slightly better accuracy. Figure 12 shows the accuracies and time taken by each hyperparameter set.

5.3 Extreme Gradient Boost (XGBoost)

The XGBoost algorithm underwent testing across various scenarios involving changes in the number of estimators or weak learners. The number of estimators tested varied from 200 to 500, and each specific number of estimators was paired with nine different tree depths ranging from 2 to 10. It's important to note that the learning rate was held constant throughout all tests at its default value of 1. As depicted in Figure 13, the model's convergence time was notably faster, achieving similar accuracies compared to the other two previously mentioned models. The most favorable outcome was obtained when employing 500 estimators in conjunction with a tree depth of 4, demonstrating efficient training completed in a mere 1.65 seconds. This configuration proved highly effective in achieving the desired result, highlighting

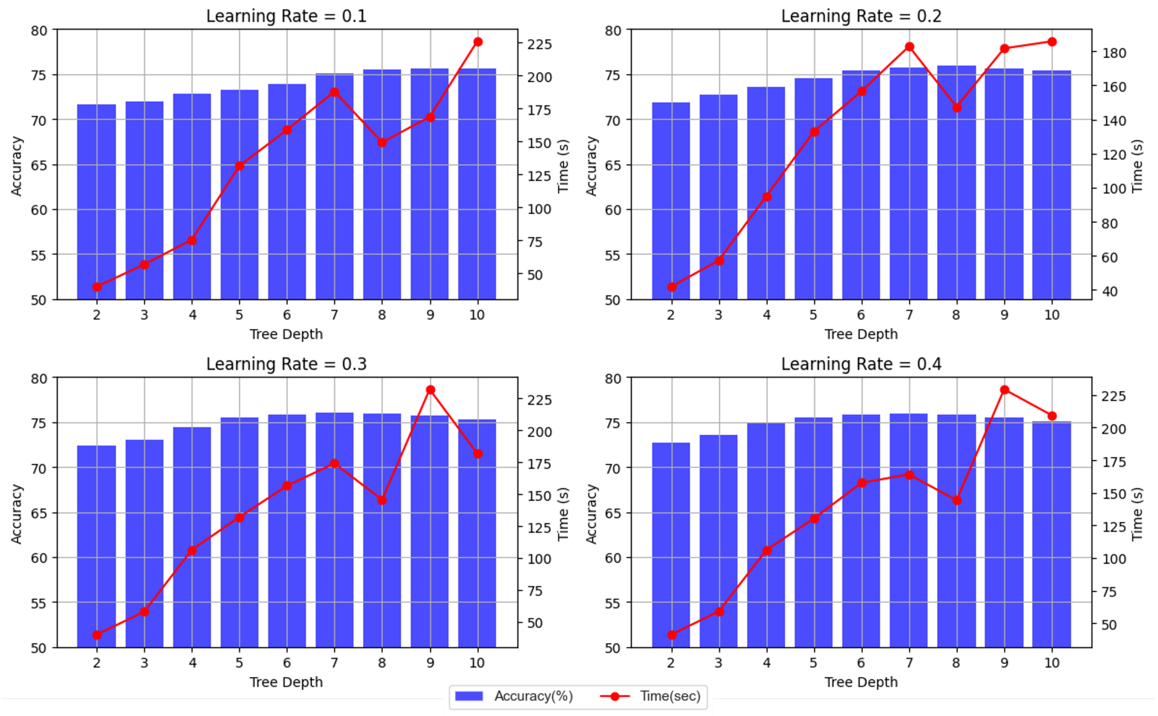


Figure 12: Accuracy and Convergence time of AdaBoost with different Learning Rates and Tree Depths

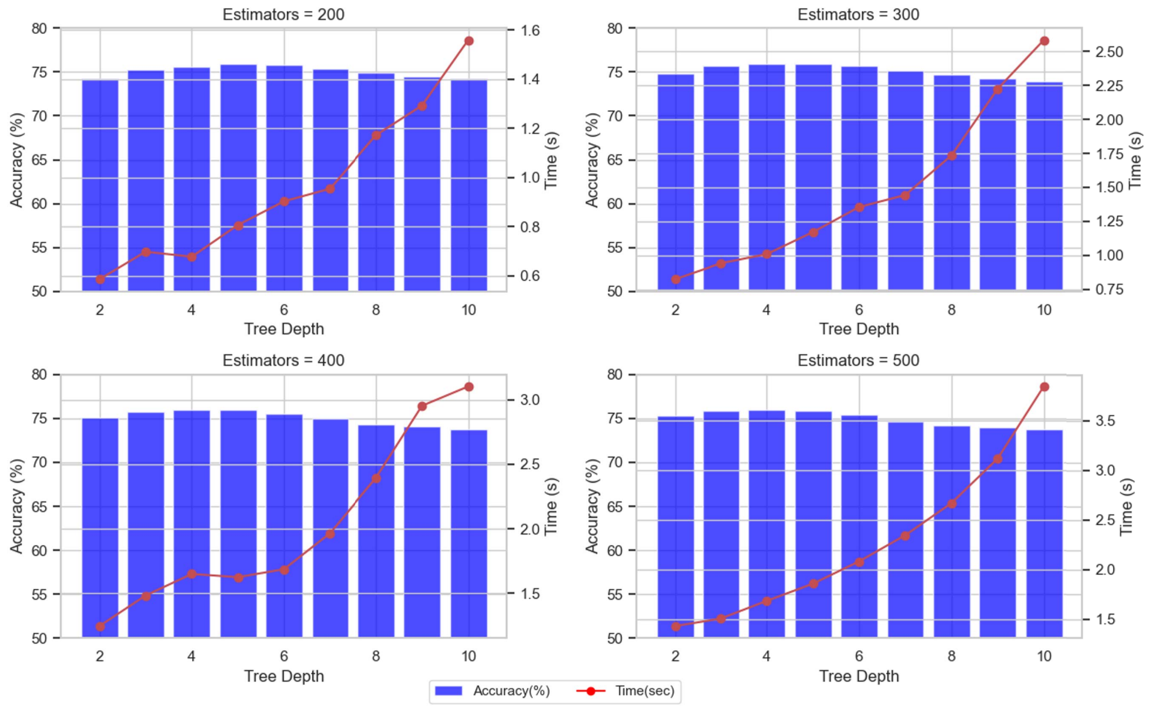


Figure 13: Accuracy and Convergence time of XGBoost with different Number of Estimators and Tree Depths

the synergy between the substantial number of estimators and the moderate depth of the decision trees.

6 Discussion

The experiments on all three algorithms revealed that the accuracy generated from the size of the dataset we utilized is comparable. Although the accuracy fluctuates between 0.75 and 0.77, the speed at which the model converges during the training process varies significantly. Table 3 presents the mean absolute error (MAE), root mean squared error (RMSE), score, and training time for each algorithm with different sets of hyper-parameters. The optimal trade-off between training time, er-

Model	Criterion		MAE	RMSE	Score	Time (s)
MLP	Batch Size	5000	0.2462	0.4692	0.7538	253.64
		10000	0.244	0.494	**0.756**	108.48
		15000	0.2461	0.4961	0.7539	134.02
		20000	0.2459	0.496	0.754	76.45
AdaBoost	learning rate	0.1	0.247	0.497	0.753	137.28
		0.2	0.2436	0.4935	0.7564	158.5
		0.3	0.2406	0.4905	0.7593	154.62
		0.4	0.2387	0.4885	**0.7613**	169.49
XGBoost	Estimators	200	0.2405	0.4904	0.7594	0.9272
		300	0.2379	0.4878	0.762	1.05
		400	0.2382	0.488	0.7617	1.42
		500	0.2377	0.4875	**0.7622**	1.65

Table 3: Comparison of the Algorithms on Different Criterion

ror, and accuracy for the MLP occurs with a batch size 10,000. The best trade-off is attained at a learning rate of 0.3 for the AdaBoost algorithm. Similarly, for XGBoost, the optimal trade-off is achieved with 500 estimators. Among the three algorithms, XGBoost has proven to be the most efficient for this task, delivering the highest accuracy in the shortest training time, which is only 1.65 seconds.

Augmenting the size of the dataset may lead to enhanced accuracy and faster convergence of the model. The three models used in this study do not account for the relationship between a vector and its consecutive vectors, which could significantly enhance model accuracy. These limitations can be addressed in our future work by incorporating a graph neural network model or sequence processing model to gain a deeper understanding of the relationships among vectors

7 Conclusion

In this study, we propose a method for conducting a comparative analysis of 3D design models printed using LPBF with the resultant data generated during the printing process. We demonstrate that this comparative analysis is valuable for creating a labeled dataset, which can be employed by machine learning and artificial neural network models to comprehend the relationships among various parameters of the printing process. The outcomes of these methods demonstrate their accuracy in predicting the printability of a new design or identifying commands in the design that the machine may struggle to print. Looking ahead, we aim to leverage the findings from this thesis to develop a method capable of analyzing the root causes of issues in print and modifying the design to address these challenges.

8 References

[1] Joran W Booth, Jeffrey Alperovich, Pratik Chawla, Jiayan Ma, Tahira N Reid, and Karthik Ramani. The design for additive manufacturing worksheet. *Journal of Mechanical Design*, 139(10):100904, 2017.

[2] Iwona Jasiuk, Diab W Abueidda, Christopher Kozuch, Siyuan Pang, Frances Y Su, and Joanna McKittrick. An overview on additive manufacturing of polymers. *Jom*, 70:275–283, 2018.

[3] Y Lakhdar, C Tuck, J Binner, A Terry, and R Goodridge. Additive manufacturing of advanced ceramic materials. *Progress in Materials Science*, 116:100736, 2021.

[4] Dirk Herzog, Vanessa Seyda, Eric Wycisk, and Claus Emmelmann. Additive manufacturing of metals. *Acta Materialia*, 117:371–392, 2016.

[5] Terry T Wohlers, Tim Caffrey, et al. Wohlers report 2013: additive manufacturing and 3d printing state of the industry: annual worldwide progress report. *(No Title)*, 2013.

[6] Yuanbin Wang, Robert Blache, Pai Zheng, and Xun Xu. A knowledge management system to support design for additive manufacturing using bayesian networks. *Journal of Mechanical Design*, 140(5):051701, 2018.

[7] Swee Leong Sing, CN Kuo, CT Shih, CC Ho, and Chee Kai Chua. Perspectives of using machine learning in laser powder bed fusion for metal additive manufacturing. *Virtual and Physical Prototyping*, 16(3):372–386, 2021.

[8] Mohammadhossein Amini and Shing I Chang. Mlcpm: A process monitoring framework for 3d metal printing in industrial scale. *Computers & Industrial Engineering*, 124:322–330, 2018.

[9] Zhongshu Ren, Lin Gao, Samuel J Clark, Kamel Fezzaa, Pavel Shevchenko, Ann Choi, Wes Everhart, Anthony D Rollett, Lianyi Chen, and Tao Sun. Machine learning–aided real-time detection of keyhole pore generation in laser powder bed fusion. *Science*, 379(6627):89–94, 2023.

[10] Bodi Yuan, Gabriel M Guss, Aaron C Wilson, Stefan P Hau-Riege, Phillip J DePond, Sara McMains, Manyalibo J Matthews, and Brian Giera. Machine-learning-based monitoring of laser powder bed fusion. *Advanced Materials Technologies*, 3(12):1800136, 2018.

[11] Eckart Uhlmann, Rodrigo Pastl Pontes, Abdelhakim Laghmouchi, and André Bergmann. Intelligent pattern recognition of a slm machine process and sensor data. *Procedia Cirp*, 62:464–469, 2017.

[12] Ying Zhang, Sheng Yang, Guoying Dong, and Yaoyao Fiona Zhao. Predictive manufacturability assessment system for laser powder bed fusion based on a hybrid machine learning model. *Additive Manufacturing*, 41:101946, 2021.

[13] Ying Zhang and Yaoyao Fiona Zhao. Hybrid sparse convolutional neural networks for predicting manufacturability of visual defects of laser powder bed fusion processes. *Journal of Manufacturing Systems*, 62:835–845, 2022.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[15] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017.

[16] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[17] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

[18] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[19] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[20] HC Taylor, EA Garibay, and RB Wicker. Toward a common laser powder bed fusion qualification test artifact. *Additive Manufacturing*, 39:101803, 2021.

[21] Ohio Supercomputer Center. Ohio supercomputer center, 1987.

[22] Ohio Supercomputer Center. Pitzer supercomputer, 2018.

[23] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.