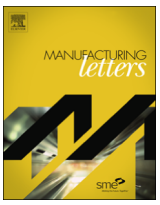
Available online at www.sciencedirect.com

Manufacturing Letters

Manufacturing Letters 44 (2025) 1456–1467



53rd SME North American Manufacturing Research Conference (NAMRC 53, 2025)

Input part shape representation/deep learning architecture and dataset analysis for additive manufacturing part quality predictions

Sara Shonkwiler[a,*], Tianshuang Qiu[a], Richard Ma[a], Chen Dai[a], Chang Li[a], Xiang Li[a], Hannah Budinoff[b], Sara McMains[a,]

[a] *University of California, Berkeley, United States*
[b] *University of Arizona, Tucson, United States*

Abstract

Deep learning (DL) models have revolutionized automation in fields such as image classification and segmentation. In traditional computer science fields, necessary training dataset size and quality, input resolution, and input shape representation/DL architecture pairings have been carefully selected for specific tasks. Predicting additive manufacturing (AM) part quality is increasingly important as more AM parts are made as end-use parts, but these predictions are often time and resource intensive. This research compares four DL pipelines' performance, across different dataset sizes and input resolutions, at predicting AM print quality. We train our DL pipelines on varied, real world data and systematically evaluate each model's predictive performance, training time, and sensitivity to hyperparameter tuning across different dataset sizes and input resolutions. We build and train voxel, depth image, and distance field 3D CNN and point cloud transformer pipelines that get far superior results to a baseline model. The distance field 3D CNN model achieves the best performance, 9.62% error, predicting AM print quality compared to 24.96% error for our baseline model. We find that dataset size and input resolution both impact model performance and hyperparameter sensitivity, but that dataset size has a greater impact on model performance than input resolution for the DL pipelines we test. We gain initial insight into what shape representation/DL pipelines are promising for improving AM part quality and performance predictions. Finally, this research demonstrates a systematic way to fairly compare multiple DL pipelines to a baseline model and evaluate the impacts of changing individual variables in the DL pipeline.

Keywords: Computer-Aided Design; Data Mining; Design Automation; Manufacturing; Neural Networks

1. Introduction

Deep learning (DL) models are powerful tools for a wide range of applications. For example, DL models enable precise image segmentation in medical diagnostics [30], optimize processes in manufacturing [17], and enhance decision-making in autonomous driving [4]. The success of DL models in these areas can be largely attributed to training DL models on enough high quality data, as well as the choice of appropriate input shape representations and corresponding DL architectures [21].

For deep learning based image classification, input images are represented as pixels using two-dimensional arrays. Convolutional neural networks (CNNs) and DL architectures that build on CNNs (e.g. ResNet50 [12]) are well-suited to finding patterns in images. Image classification models are trained on large, high-quality, labeled image datasets, such as MNIST [7], CIFAR-10 and CIFAR-100 [16], and ImageNet [6]. Deep learning has revolutionized image classification, but only because of careful decision making about training data and DL pipeline architectures.

The application of DL models in predicting the quality and performance under load of three-dimensional mechanical parts is an emerging and promising area of research [13, 33, 10, 31, 14, 1, 18, 29, 22, 11, 19]. Deep learning models have the poten-

* Corresponding authors. Email: mcmains@berkeley.edu, sara_shonkwiler@berkeley.edu

tial to efficiently learn patterns in part quality and performance data by exploiting similarities between analogous shapes where two additively manufactured parts have similar finite element analysis (FEA) manufacturing process distortion predictions. If successful, DL models will drastically reduce the computational and time requirements to run FEA for part quality and performance data. Speeding up FEA will enable engineers to iterate designs more quickly and discover better engineering designs. Reducing the computational requirements of FEA will make it possible for engineers and researchers without access to extensive GPU compute resources to generate FEA results. Early DL part quality and performance research has generated good results; however, current research is limited to training models on generally small to medium datasets of relatively simple, artificial or geometrically limited parts.

Existing DL part quality and performance research can be characterized by choices made for various aspects such as: (a) input dataset geometries, (b) dataset size, (c) input shape representation, (d) DL architecture, and (e) prediction type.

(a) Input dataset geometries: Nie et al. [22] and Wang et al. [29] simplify the input to two-dimensional geometries. Jin et al. [13], Nie et al., Wang et al., Dong et al. [10], Khadilkar et al. [14], Eranpurwala et al. [11], and Williams et al. [31] synthetically generate simple CAD parts (see example parts in Fig. 1) for the DL models to train on. Liang et al. [18] and Balu et al. [1] generate synthetic biological parts by parameterizing from an original real biological part thus keeping the input geometric variability quite limited.

(b) Dataset size: The amount of data generated for these research projects varies widely. Wang et al. [29] trains their model on 128 parts. Jin et al. uses 200 parts [13]. Wong et al. [33] uses 477 parts. Dong et al. [10] uses 420 parts rotated into a second orientation for 840 total parts. Liang et al. [18] uses 729 parts. Nie et al. [22], which makes 2D FEA predictions, has 120,960 common cantilevered geometries. Williams et al. [31], which predicts part mass, support material mass, and build time, uses 72,000 parts. Khadilkar et al. [14] uses 16,700 parts. Balu et al. [1] uses 90,941 heart valve parts.

Balu et al. uses a dataset of over 90,000 parts with a constricted range of geometries, all heart valves, while Liang et al. with a seemingly similar level of geometric diversity, all aortas, uses a dataset of 729 parts. Realistically, deep learning models (and even machine learning models) need a great deal of data to learn, but using more data than necessary wastes memory and computation time. In this research, we experimentally evaluate how much data is needed for the AM part quality and functional performance DL domain.

(c & d) Input shape representation & DL architecture: For the research projects that investigate three-dimensional geometries (research that simplifies the input to 2D geometries is omitted because it may not generalize well to 3D) shape representation/DL architectures used include voxel representation and 3D CNN models [31, 10, 11], point cloud and CNN-PointNet hybrid model [14], NURBS surface reformatted and convolutional autoencoder algorithm [1], and mesh representation paired encoder-decoder algorithm and/or graph neural network (GNN) [13, 33, 18]. There is a lack of clear guidance as to which shape representation/DL architecture combinations are most effective for DL part quality and performance models, including FEA surrogate models.

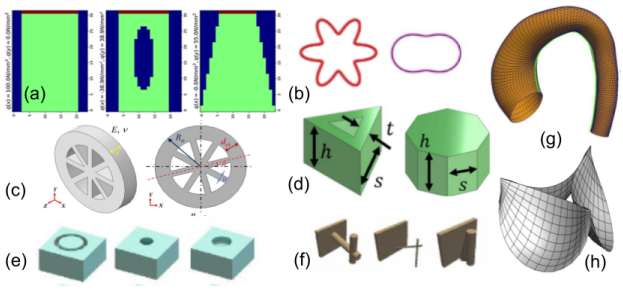


Figure 1: Input part geometries from related data-driven part quality and performance prediction research [10, 22, 11, 1, 13, 18, 29, 31]. All parts are synthetically generated for the specific purpose of doing research except in the case of the two biomedical part datasets where parts are created by artificially parameterizing biological parts. Our research uses parts from CAD repositories that more accurately reflect the true variety of mechancial parts than synthetically generated geometries. (a) & (b): two-dimensional geometry inputs [29, 22]. (c) - (f): "simple" synthetic geometries [13, 31, 10, 11]. (g) & (h): geometries based on parameterizing biological part [1, 18].

(e) Prediction type: Current DL models predict part mass and build time [31], stress [13, 10, 18, 22], deformation [1], bottom-up stereolithography separation stress [29], and best additive manufacturing (AM) orientation [11, 19]. Previous related research is summarized in Table 1.

The primary objective of this research is to systematically compare promising shape representation/DL architecture combinations and data requirements for predicting the AM print quality of engineering parts. We investigate the impact of dataset size and resolution on prediction results for AM print quality, which will help inform researchers wanting to make their own datasets or create benchmark datasets especially for additive manufacturing part quality predictions. To see the impact of dataset size on performance, we have collected a large, high quality dataset. Similar to Williams et al. [31] and Eranpurwala et al. [11], the AM print quality labels in this research are analytically generated and are not the main contribution of this research. We hypothesize that DL pipelines that work well on our AM print quality problem will transfer well to more computationally expensive FEA print quality and performance labels where the analytical approach has significant drawbacks, namely the time and computational resources required.

We evaluate the shape representation/DL architecture pairings based on the following criteria: (1) DL pipeline predictive performance over a range of dataset sizes, (2) time and computational resources required to train DL pipeline, and (3) DL pipeline performance's sensitivity to hyperparameter tuning.

Three-dimensional CAD data is memory intensive, and thus while it is important to have large datasets for training, it is also important not to store and train on more data than needed for good results. Another objective of this research is to advance understanding of the relationship between dataset size and DL performance in this domain.

Table 1: Deep learning part quality and performance research

Paper	Input data geometries	# of parts	Input shape representation	DL architecture	Prediction type
Nie et al. [22]	2D cantilevered geometries	120,960	pixel geometry, loading + boundary conditions	CNN	stress field
Wang et al. [29]	simple 2D geometries, chess pieces	128	2D shape context descriptor	NN	SLA separation stress
Dong et al. [10]	circular and square struts, walls	840	voxels	3D CNN, 3D-UNet	residual stress
Khadilkar et al. [14]	simple, "artificial" geometries	16,700	point cloud + image of bottom layer	CNN + PointNet	2D FEA
Balu et al. [1]	"artificial" heart valves	90,941	reformatted NURBS surface	convolutional autoencoder	deformation
Williams et al. [31]	simple, "artificial" geometries	72,000	voxels	3D CNN	part & support mass, build time
Liang et al. [18]	"synthetic" aortas	729	quadrilateral mesh	encoder, mapping, decoding via NNs	stress
Jin et al. [13]	parameterized wheels	200	mesh	GNN	3D stress
Eranpurwala et al. [11]	simple, "artificial" geometries	54,000	voxels	3D CNN	best orientation
Wong et al. [33]	circular and square struts, walls	1,505 (477)	mesh	GNN	pressure, coeffs (residual stress)
Our Work	mechanical parts	482,214	voxels, point clouds distance fields, depth images	3D CNN, Transformers	AM printability scores

In this research, four shape representation/DL architecture pairings and a baseline model are compared as models to predict AM print quality. Each DL model is trained on data from our real world engineering dataset. The models are systematically tuned and compared. Our main contributions include:

- Comparison of four shape representation/DL architecture pairings to a baseline model for AM print quality prediction:
  - Distance field 3D CNN DL model,
  - Depth image (from top and bottom) 3D CNN DL model,
  - Voxel 3D CNN DL model,
  - Point cloud Transformer DL model,
  - Mid-point baseline model;
- Analysis of impact of dataset size on DL pipeline predictive performance;
- Analysis of trade-offs between DL pipeline predictive performance, runtime, and hyperparameter sensitivity to give a full picture of each model's performance;
- Hundreds of thousands of varied, real world, labeled CAD parts for model training and comparison.

2. Background

AM (often colloquially referred to as 3D printing) processes are a group of relatively novel manufacturing processes that were originally used for prototyping, but are increasingly used to fabricate end-use parts [32], which requires higher quality manufacturing. AM processes give engineers and designers greater geometric design flexibility and shorter product lead times, but are generally more defect-prone than traditional manufacturing processes. AM allows companies to cut outsourcing costs, iterate designs more rapidly, speed up the product development cycle, generate parts with complex geometries, and customize parts. The flexibility in geometric design and potential for faster iterative design of AM processes give them the potential to revolutionize industries such as aerospace and medicine.

With the improvement of AM capabilities, these manufacturing processes will shape the way we design and manufacture a variety of goods. However, AM processes often have substandard mechanical properties and are prone to manufacturing, or build, failures [28]. Improving AM manufacturing quality and reliability is important if these processes are to be used for safety-critical end-use parts.

Most AM processes fabricate parts from the bottom up, layer-by-layer. A 3D CAD model is sliced into thin layers using a slicing software (e.g. Cura [8]). Then the AM process creates one layer of the part at a time, for example by depositing or sintering material on the 2D plane for that layer. Then the machine proceeds to the next layer and repeats until the part is complete. In the case of fused deposition modeling (FDM), one of the most common AM processes, the equipment heats up a plastic filament, which is fed through a nozzle and deposited onto the build platform layer by layer to form the object.

Evaluating if a CAD part in a particular orientation will be manufactured successfully is an ongoing challenge. Researchers have explored analytical approaches to optimizing 3D printing orientation (e.g. Tweaker3 [25], which is used as a plug-in for Cura). AM print orientation has been optimized for objectives such as surface quality, build time, support structures, build cost, part accuracy, post-processing time, and post-machining defects using a variety of non-machine learning approaches [3, 9, 24]. Researchers have used approaches that include exhaustive search, genetic algorithms, simulated annealing, particle swarm optimization, nonlinear programming solver, heuristic algorithms, and others [3]. These approaches tend to optimize a single objective (e.g. the amount of support volume required) in a particular use case and very infrequently include comparisons to other algorithms, making it challenging to assess algorithm success.

Eranpurwala et al. [11] converts "synthetically" generated mesh files (STL files) into 64^3 voxelized files and uses a 3D convolutional neural network (CNN) to predict optimal print orientation. Parts are labeled by the amount of support volume required to print. The support volume required is calculated using an analytical formula.

Our research builds on this research to predict AM print quality in a given orientation using DL pipelines. However, our main contributions are providing insights into the impact of shape representation/DL architecture and dataset size on AM print quality predictions. While the focus of this research is on one build quality metric for FDM, we hope the findings of this work can be used to help inform other AM print quality predictions with similar data distributions.

The next section (Methods) describes our research methodology including data collection, labeling, and pre-processing, DL pipeline details and selection rationale, and the DL pipeline comparison framework. The following section (Results) describes the experiments and results from the input shape representation/DL model pipelines including hyperparameter tuning, model complexity and runtime requirements, and dataset size sensitivity analysis. The final section discusses potential directions for future work and conclusions.

3. Methods

A major gap in research this study addresses is evaluating what shape representation and DL model pairings have enough expressivity to accurately predict AM print quality and performance metrics. This section explains why we selected particular shape representation/DL pipelines, how we collected and processed the part data and created the AM print quality labels, the details of our deep learning (DL) approaches, and finally the process used to systematically compare DL pipelines and evaluate dataset requirements.

We build a 64^3 voxel 3D CNN pipeline because it is a natural analog to CNN image recognition pipelines and as a comparison to other research groups that have used the voxel 3D CNN pipeline for similar research [11, 31, 10]. Related research uses

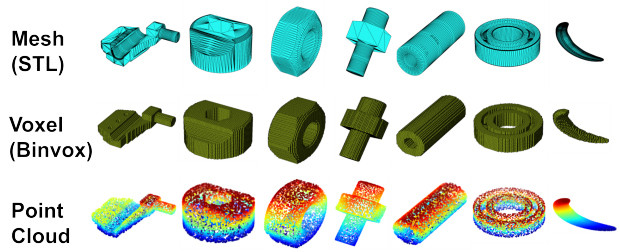


Figure 2: Seven example input parts in three shape representations we include in our dataset (original part representation: mesh/STL, voxel/Binvox, point cloud). One of the contributions of this research is training on a large dataset of real mechanical parts, instead of parts synthesized just for research, in multiple shape representations. We show our final two shape representations in the next Figure because they are more involved.

64^3 voxels with small to medium size datasets (840 – 72,000 parts).

3.1. Data collection, labeling, and shape representations

The part data we use are real world parts from the unlabeled FabWave data repository [2], which were in turn scraped from GrabCAD and Autodesk Gallery as mesh files (STL files). We collect over 80,300 parts. We rotate all parts into five additional orientations to place each other axis facing up, i.e., rotating the original part with the $+z$-axis oriented up to instead have the $\pm x$-axis, $\pm y$-axis, and the $-z$-axis oriented up for the five additional orientations. This gives us over 482,000 part-orientation combinations. Rotating parts into new orientations creates a more random distribution of part-orientation combinations and increases the dataset size.

The dataset is randomly shuffled and we first separate out a 10,000 part validation dataset (which we use to validate all models in this research). Then we randomly sample a 200,000 part training dataset from the parts left. We take the first 1,000, 10,000, and 50,000 parts as subsets to explore the effect of dataset size. Finally, we remove any part in these training datasets that exists in the validation dataset in any orientation, even though different orientations usually have different labels, to ensure complete isolation between the training and validation datasets. This gives us four final training datasets of sizes 911, 9,041, 45,091, and 180,294.

We explore different input shape representations to evaluate which input shape representation does the best job of predicting print quality and to try and find the most efficient input (i.e. an input shape representation that will get comparable results in less time or with less computational resources). We test the following shape representations, in addition to voxels, also with the 3D CNN DL architecture: (1) distance fields and (2) depth images. Finally, we tried a new shape representation/DL architecture pairing, a point cloud Transformer model.

Voxel 3D CNN represents perhaps the most intuitive and easy to understand pairing of shape representation and DL architecture. Different voxel 3D CNN architectures have been used in several related papers using 64^3 voxelized input. The depth image and distance field representations mimic the AM process. Point clouds have been used in a variety of DL applica-

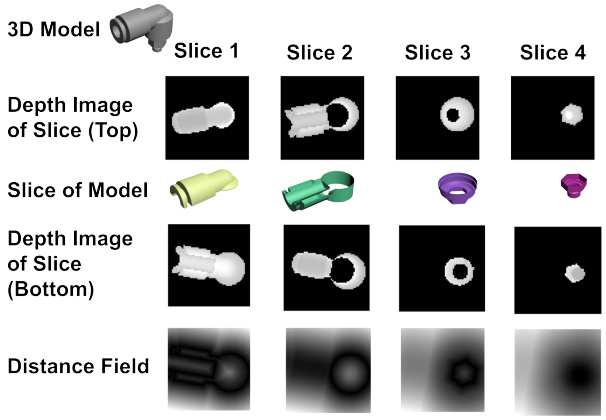


Figure 3: One example part, shown in the top left corner, in depth image (from top and bottom) and distance field shape representations. For the actual data representations, parts are split into 64 slices (or layers). For visualization purposes this part is only split into four slices.

tions, such as the Point Transformer architecture, which is designed to capture complex local and global relationships from point cloud inputs.

To evaluate input dataset requirements, we isolate input resolution and increase the input resolution from 64^3 to 256^3 to see the impact of the input resolution on performance.

We convert the mesh files (STL files) into 64^3 and 256^3 resolution voxelized files (Binvox files [20, 23]) and point clouds ($2,048 \times 3$ coordinates using `trimesh` [5] , where $2,048$ is the number of points in the point cloud). Example parts in each of these three shape representations are shown in Fig. 2.

We transform the parts into the depth image shape representation using the pinhole camera model: starting from a point in space (either the top or bottom of the part), we create one ray per pixel and return the distance traveled until the ray intersects with a surface of the part. If no intersections happen, we return zero. We evenly slice each mesh into 64 horizontal slices to match the resolution of the 64^3 voxel shape representation. Each slice is lowered so that its centroid lies on the X-Y plane. We compute two depth images for each slice, one taken from above and one taken from below. Each depth image has a resolution 64×64, for a total resolution of 64^3 for the depth image representation of the part.

Each part's distance field shape representation is created by computing the unsigned distance from each query point to the nearest face. Query points are sampled in a square grid and resized to fit each mesh. Each square grid plane is perpendicular to the z-plane and axis-aligned. These planes are positioned to evenly slice the mesh into 64 pieces, identical to the height of the cuts to generate distance fields. Each square grid contains 64×64 points, and thus the total resolution of the field is 64^3. The depth image and distance field shape representations (example in Fig. 3) input the part to the DL model as a series of layers, which is similar to how the part is fabricated and thus we predict an expressive input for the DL pipeline.

To label the full dataset with AM print quality, we calculate an unprintability score (UPS) for every part in our dataset using the analytical research software Tweaker3 [25], an open source slicer plug-in. Tweaker3 uses three factors to calculate the unprintability score (UPS): (1) area touching the print bed, (2) overhang area (meshes where the angle is greater than 45 degrees), and (3) contour length of the area touching the print bed. (These factors are related to stability, volume of support structure, and warping, respectively.)

Lower UPS scores represent a lower risk of AM print failure. The UPS Tweaker3 output is a description of the AM printability of the part. The labels range from $[0, \infty)$, with zero representing a highly printable orientation-part combination, and larger numbers representing more problematic part-orientation combinations. Using UPS as the labels allows us to label hundreds of thousands of parts and perform DL analysis on these parts.

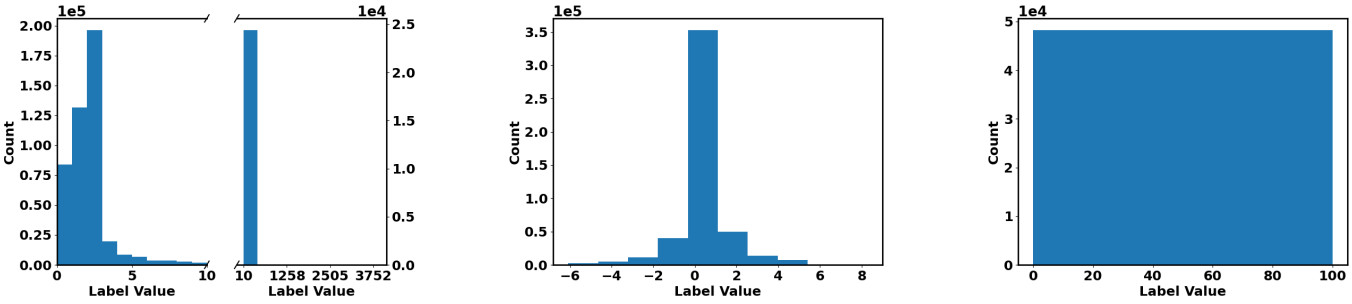
The UPS label data is heavily right-skewed, as illustrated in Fig. 4a: most of the parts have relatively small unprintability scores, but a small percentage of the parts have much larger scores. It is important to be able to predict these larger values because these scores represent parts that likely will have problems in the manufacturing process. It is more challenging to predict right-skewed data, but being able to predict the right-skewed UPS scores is a good simulator for being able to predict more computationally expensive and complicated manufacturing metrics, because other manufacturing quality data may also be skewed with a small percentage of problematic areas that are important to identify (i.e., FEA predictions of areas of high process-induced distortion).

It is reasonable to expect other types of AM manufacturing labels to be non-uniform because more parts will successfully print (or have low residual stress, distortion, etc.) than not. If that were not the case, AM manufacturing processes would not be as successful as they are. On the other hand, there are still a significant number of print failures and it is highly important that any DL model be able to successfully predict when prints will fail (or have issues) even though this may be a small proportion of the total data.

Label data can be processed in several ways to handle skewed data, including taking the log of the data (Fig. 4b) and converting scores to their percentiles (Fig. 4c). We chose to preprocess the label data into percentiles (i.e., calculate what percentage of the data a value is larger than) in order to make the data evenly distributed. The processed UPS scores range from zero to one.

3.2. *The importance of baseline models*

Generally, the goal of DL pipelines in this field is to learn some information about AM build quality. There are a variety of metrics to measure how much the DL model has learned (e.g. L1 loss), but it can be difficult to fairly evaluate how much a DL model has actually learned from performance metrics alone. For comparison, an appropriate baseline model should always be constructed, which is used to give the researcher an idea of how much the DL model(s) learned compared to a simple alternative. We construct a midpoint model that guesses that every model has a processed UPS score of 0.50, which is the midpoint of the uniformly distributed, processed UPS scores.



(a) Raw data distribution.

(b) Log data distribution.

(c) Percentile data distribution.

Figure 4: Histograms of the unprintability score (deep learning label) data: (a) raw data distribution, (b) log data distribution, and (c) percentile data distribution. The raw data, shown in (a), is heavily right skewed. For the raw data graph, the x-axis scale is split to make it possible to visualize the number of higher unprintability scores. We show the data processed by taking the log in (b) and using percentiles in (c). Ultimately we pre-process the data using percentiles so that the unprintability score (labels) are uniformly distributed. This makes the error results more meaningful. If we left the data skewed, we could get results that appeared excellent merely by guessing a low value unprintability score at all times, but actually did not indicate significant learning.

This comparison addresses the risk with research projects that use a limited variety of geometries (especially when the geometries are relatively simple) that a DL model may look like it's getting good results, when in reality it is guessing near the average or mode result for all parts. Note that had we used the log of the data (as visualized in Fig. 4b) instead of percentiles to pre-process our data, an appropriate baseline model would be approximately at the mean of a normal distribution fitted to the log data. Because of the narrow normal distribution of the logs, a DL model that learned to predict the mean log value, regardless of the input, would have appeared to have low error if error alone was reported without comparison to this appropriate baseline that could have obtained a similarly low error.

Alternatively if we had not pre-processed the highly skewed data at all, we could always guess low UPS values and get lower error than if we attempted to learn when the UPS value is high. However, it is important to be able to predict high UPS scores, which represent likely problematic 3D prints. This is why it is crucial to include an appropriate baseline model in DL applications. A baseline model makes it clear how much learning is taking place.

3.3. Deep learning pipelines

3.3.1. Voxel, distance field, and depth image 3D CNN models

Three-dimensional convolutional neural networks (3D CNNs) are a class of deep learning models designed to process, analyze, and learn from 3D input data. These networks extend the traditional 2D image-based CNN architecture by adding a third dimension to the input and performing convolutions in 3D space.

We have two closely related 3D CNN architectures: one for the 64^3 resolution voxel, distance field, and depth image inputs and one for the 256^3 resolution voxel input. The overall voxel 3D CNN pipeline is in Fig. 5. Each convolutional set contains the convolutional layer itself, followed by batch normalization, and then a non-linear activation function. The convolutional layer kernel size is a hyperparameter that we tune. The outputs from the convolutional layers are padded with zeros to retain their original dimensions. Each set is followed by a max pooling layer with kernel size 2x2x2, which compresses the data to half of the original resolution in all three dimensions with the goal of extracting the latent features from the convolution. The channel count of the convolutional layers also varies, with each layer outputting twice or four times the number of channels it takes in, as described next.

For the 64^3 voxel resolution inputs, our DL architecture has five convolutional sets. We choose five convolutional sets to reduce the resolution from 64^3 to 4^3 by halving the resolution at each layer. The number of channels in each hidden layer increases from two output channels (or nodes) for the first hidden layer to 32 for the final 3D CNN hidden layer. After the convolution sets, a final max pooling layer with kernel size 4x4x4 reduces the 32 channels into 32 numbers (a 1D array of numbers). Two fully connected linear layers each with weights for every neuron and a bias at the end to shift the output follow, the first mapping those 32 numbers to 8 numbers, then the second from 8 to 1. This final number is the predicted unprintability score.

For the 256^3 voxel resolution, our DL architecture has six convolutional sets. We choose six convolutional sets to reduce the resolution from 256^3 to 8^3 by halving the resolution at each layer. The number of channels in each hidden layer increases from two output channels (or nodes) for the first hidden layer to 256 for the final 3D CNN hidden layer. After the convolution sets, a final max pooling layer with kernel size 8x8x8 reduces the 256 channels into 256 numbers (a 1D array of numbers). Two fully connected linear layers each with weights for every neuron and a bias at the end to shift the output follow, the first mapping those 256 numbers to 16 numbers, then the second from 16 to the UPS.

For both architectures, the networks gradually compress the input data, and extract the latent features from within the convolutional layers. The channel count keeps increasing through the convolution sets in order for the 3D CNN to capture more features from the inputs while compressing the data. Finally, linear layers transform the spacial data we get from the convolutions into the prediction of the UPS.

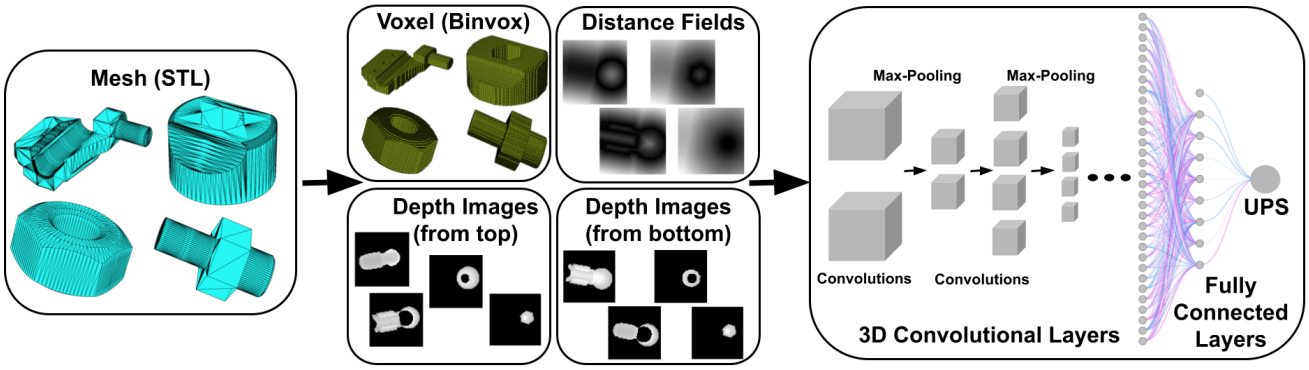


Figure 5: 3D CNN model pipelines. Compiled dataset of parts are in the STL file format. Parts are converted to 64^3 and 256^3 voxelized (Binvox) format, 64^3 distance field format, and two 64^3 depth image formats using an automated pipeline and then fed into the 3D CNN model which consists of five or six convolutional layers (depending on input voxel resolution) and two fully connected layers that reduce the output to a single score, the unprintability score (UPS).

3.3.2. Point Cloud Transformer Model

In the Transformer architecture [27], multi-headed attention allows the network to learn relational information in different parts of the input simultaneously, which is useful in our context because AM print quality is determined not by the existence of each individual point but by how interactions between points impacts the manufacturing process. An advantage of multi-headed attention is that it is inherently permutation invariant, which is a desired feature in this research since the ordering of individual points does not affect part shape.

Point Transformers [34] build on the PointNet work by using self-attention, which relates different parts of the input geometry to each other, to capture relationships between the input points. Point Transformers outperformed the next best architecture by almost 3% on the segmentation task measured by mean intersection over union (mIoU) as of 2021 when the paper was published [34]. Point Transfomers use vector attention instead of the more common scalar attention because scalar attention loses spatial information (vector attention can capture geometric relationships in 3D and scalar attention cannot) and is not permutation invariant.

Point Transformers use vector attention in a transformer layer. The prediction network has n repeating blocks, each block containing a *Transition Down* layer and a *Transformer Layer*. For the *Transition Down layer*, k-nearest neighbors (kNN) is used to pool local features that are processed with a multi-layer perceptron (MLP). The output is then locally max-pooled to reduce the number of features. We perform a hyperparameter search on this k as well as the n in our results section.

3.4. Methodology for Systematic Comparison of DL Models

To evaluate our DL pipelines, we compare performance (using validation L1 loss defined as $L(UPS, \hat{UPS}) = \frac{1}{n}\sum_{i=1}^{n}|UPS_i - \hat{UPS}_i|$), model efficiency (measured by runtime), and hyperparameter tuning sensitivity (by evaluating the average and standard deviation of our results). The DL pipelines are also directly compared to the baseline model. We contribute to the existing gap in knowledge by comparing and evaluating the DL pipelines and creating a framework for future researchers to evaluate additional DL pipelines, which will help determine what DL pipelines work best for predicting AM print quality. This framework can also be used to compare performance for new areas of DL application.

4. Results

We use our DL pipelines and baseline model to predict 3D printing part quality, defined by the UPS. Note that although we predict UPS, this is just one example of a manufacturing part quality or functional performance metric that our models could predict. Predicting UPS is a challenging research problem because the regression data is heavily right-skewed. We believe UPS may have similar data characteristics to other AM manufacturing process labels and hopefully some of our results can be used to inform other data-driven AM prediction research.

The DL pipelines are trained on a range of dataset sizes, from 911 to 180,294. All models are validated on a separate 10,000 part validation dataset.

4.1. Mid-point baseline model performance

We run the mid-point baseline model, which predicts that every part will have a processed UPS score of 0.5, on the validation dataset. The mid-point baseline has an L1 loss on the validation set of 0.2496, which is in line with the expected value of 0.25 when guessing 0.5 for every input.

4.2. Voxel 3D CNN model performance

We train and tune our 64^3 voxel 3D CNN model on four datasets of sizes 911, 9,041, 45,091 and 180,294, and our 256^3 voxel 3D CNN model on three datasets of sizes 911, 9,041, and 45,091.

We tune the following hyperparameters: kernel size (KS), learning rate (LR), batch size (BS), and non-linear activation function. Kernel size is tuned as either three or five. Learning rate is tuned as 0.001, 0.0001, or 0.00001. Batch size is tuned as 4, 8, or 32 for the 64^3 voxel models, but always 4 for the 256^3 voxel models because that is the largest batch size that can be loaded. Non-linear activation function was initially trained

as ReLU or Sigmoid, but we report final results for just ReLU because, as we show below, ReLU performs better across the board.

We do not train the 256^3 voxel 3D CNN model on the 180,294 part dataset because that would be highly resource intensive and we find that we already have enough data to show that increasing dataset size has more impact on model predictive performance than increasing input resolution. Similarly, we train the models for 20 epochs in all cases except for the 256^3 voxel 3D CNN model when trained on the 45,091 dataset. In this single case, we use 10 epochs because of how long it takes to run just one hyperparameter combination (i.e. over a day).

Both voxel 3D CNN models perform better when trained on more data, and the 256^3 voxel 3D CNN model performs better than the 64^3 voxel 3D CNN model when trained on the same size dataset. As a trade-off, larger dataset size seems to have a greater impact on performance than higher input resolution (see Tables 3 and 4). An example: increasing dataset size by a factor of approximately 20 (from 9,041 to 180,294) while decreasing input resolution by a factor of 64 (from 256^3 to 64^3) has a better best performing DL model and better average DL model performance.

Both voxel 3D CNN models' ability to generalize from the training data to the validation data differs based on how much data the model was trained on. For example, for the 64^3 3D CNN model trained on the 180,294 part dataset, the validation L1 loss can be minimized to 0.0975 with the best hyperparameters, compared to 0.1167 for the 45,091 dataset, 0.1383 for the 9,041 part dataset, and 0.1742 for the 911 part dataset with their best hyperparameters. The higher validation losses when training on the smaller datasets occurs mostly due to more overfitting to the training data for the smaller datasets. This can be inferred because the 180,294 part dataset has only a somewhat lower training L1 loss than the 911, 9,041, and 45,091 part datasets, but it has a much lower validation loss. This effect can be seen in Fig. 7, which compares training and validation losses for the models trained on the 911 and 45,091 part datasets. The model trained on the larger (45,091 parts) dataset does not overfit, which we can see from the similar training and validation losses, whereas for the smaller (911 parts) dataset, the model does seem to overfit, because the training loss is much lower than the validation loss.

The average performance (measured by average validation L1 loss) is significantly better when the 64^3 voxel 3D CNN model is trained on the 180,294 part training dataset or the 256^3 voxel 3D CNN on the 45,091 part training dataset compared to the smaller datasets for each resolution. For the 64^3 voxel 3D CNN model, the average L1 loss is 0.1158 for the 180,294 part dataset compared to 0.1349 for the 45,091 part dataset, 0.1619 for the 9,041 part dataset, and 0.2113 for the 911 part dataset (see Fig. 6 for full visualization).

Another advantage of training on the larger datasets is that it makes the model less sensitive to hyperparameter tuning. For example, on the 64^3 voxel 3D CNN model, the standard deviation of the loss is lower for the 180,294 part dataset, 0.0148 compared to 0.0171 for the 45,091 part dataset, and 0.0216 for the 9,041 part dataset.

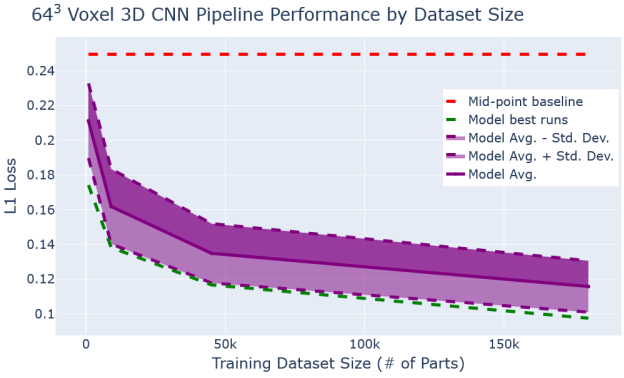


Figure 6: 64^3 voxel 3D CNN model performance results (measured using L1 loss) compared to the mid-point baseline model for 911, 9,041, 45,091, and 180,294 training datasets. Model performance improves significantly with increasing training dataset size; however, the effects of additional data decrease as dataset size increases.

For the 256^3 voxel 3D CNN model, the average L1 loss is 0.1185 for the 45,091 part dataset, compared to 0.1469 and 0.1936 for the smaller training datasets. For the 45,091 part dataset, the standard deviation in the L1 loss results is also smaller, 0.0099, compared to 0.0177 and 0.0237 for the smaller datasets.

For 64^3 and 256^3 voxel 3D CNN pipelines, increasing dataset size by approximately a factor of 10 (from 911 to 9,041) decreases L1 loss (thus improving model predictive performance) by 23.4% and 24.1%. Increasing dataset size again, this time by a factor of approximately five, decreases L1 loss by 16.7% and 19.3%. The final roughly four-fold increase in dataset size for the 64^3 input resolution decreases L1 loss another 14.2%. In contrast, increasing input resolution by a factor of 64 only decreases L1 loss by 8.4%, 9.3%, and 12.2% (for the 911, 9,041, and 45,091 dataset sizes).

4.3. Distance field and depth image 3D CNN model performance

We tune the 64^3 distance field and depth image 3D CNN models on the exact same hyperparameter combinations as the 64^3 voxel 3D CNN model. For the same dataset size (45,091) and input resolution (64^3), the average and best distance field 3D CNN model performs better than both depth image and voxel 3D CNN models. Furthermore, the average 64^3 distance field 3D CNN model trained on a dataset of 45,091 performs almost as well as the average 64^3 voxel 3D CNN model trained on a dataset of 180,294. A comparison between voxel, depth image, and distance field 3D CNN models, all trained on the 45,091 part dataset, is shown in Fig. 8. The performance results for the distance field 3D CNN, which performs better than either depth image 3D CNN model, trained on a range of dataset sizes, is illustrated in Fig. 9.

4.4. Point cloud Transformer model performance

The point cloud Transformer model is trained and tuned on three dataset sizes: 911, 9,041, and 45,091.

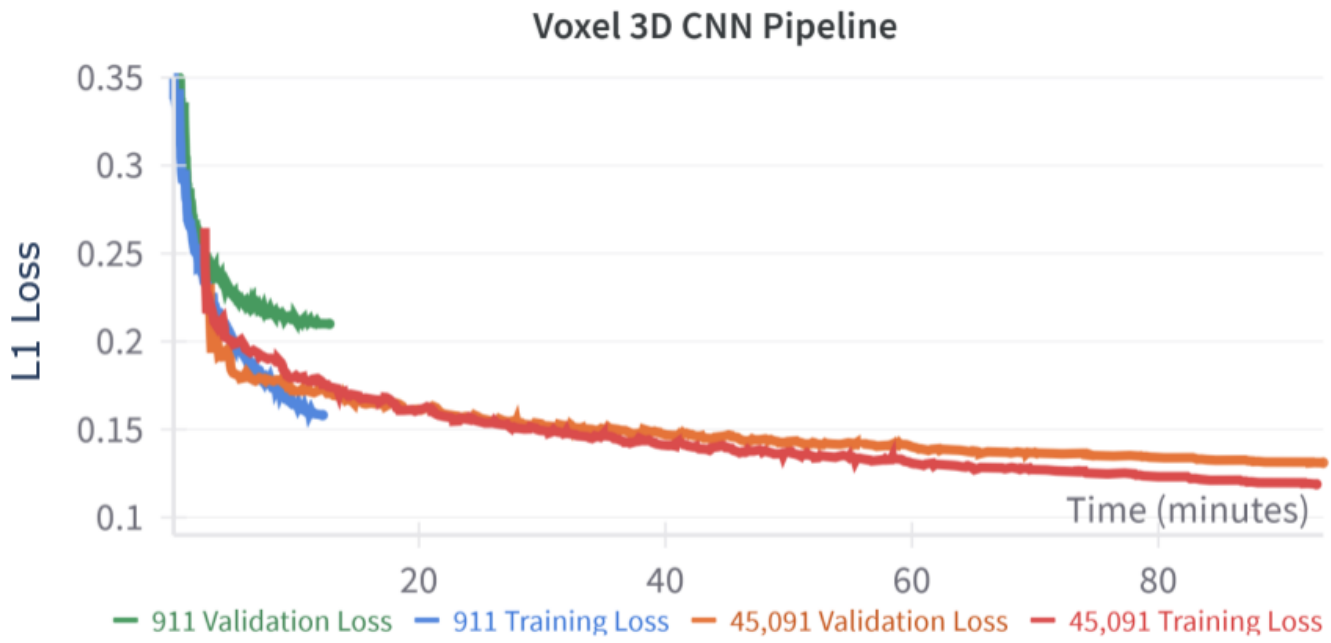


Figure 7: Comparison of training and validation losses for the 64^3 3D CNN models trained on 911 and 45,091 part datasets. Graph shows mean and one standard deviation in each direction for training and validation on each dataset. It can be seen that the model overfits more on the 911 part dataset than the 45,091 part dataset due to the larger gap between training and validation performance for the model trained on the 911 part dataset.

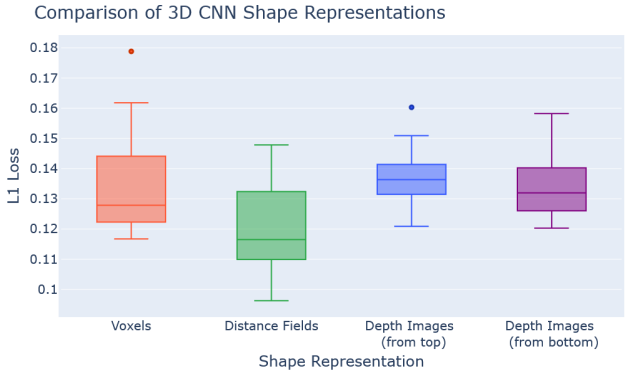


Figure 8: Comparison of 64^3 voxel, distance field, and depth image 3D CNN models' performance (measured using L1 loss) trained on the 45,091 part dataset.

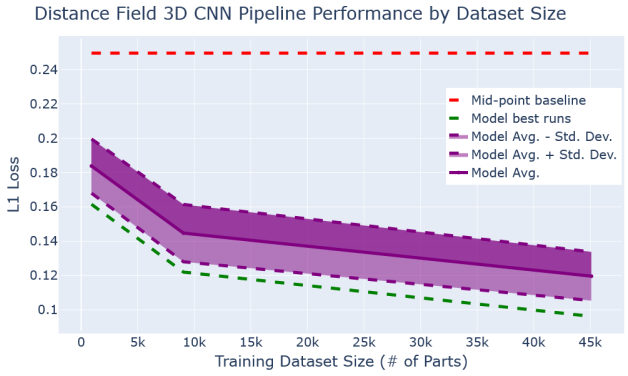


Figure 9: Graph showing the Distance Field 3D CNN pipeline's ability to predict additive manufacturing print quality (measured using L1 loss) compared to the mid-point baseline model for 911, 9,041 and 45,091 training datasets.

We tune the following hyperparameters: learning rate (LR), batch size (BS), number of nearest neighbors (k), number of transformer blocks (nb), and transformer embedding dimension (TD). Learning rate is tuned as 0.001 or 0.0001. Batch size is tuned as 4, 16, or 32. K-Nearest Neighbors is 8, 32 or 128. The number of transformer blocks is tuned as 2 or 4. The transformer embedding dimension is 32 or 64. Number of epochs is 20. Non-linear activation function is trained as ReLU, the same implementation used in the Point Transformer architecture [34].

Similar to the 3D CNN pipelines, the Transformer pipeline performs better when it trains on the larger datasets. We ran the point cloud Transformer model on the 180,294 part dataset and it was plateauing. Since computational resources are finite, we do not fully train and tune it on the 180,294 part dataset because we expect the results will be only marginally better, and not affect our conclusions.

Our results suggest that although point cloud Transformers perform well with tasks such as semantic classification and segmentation of point clouds, at the current dataset size and input resolution, they do not perform as well as 3D CNNs for AM part quality problems.

To test if point-cloud resolution could be limiting performance, we increase point cloud resolution, testing point cloud size ranges from 2,048 to 16,384 points. Point cloud Transformer performance actually worsens with increasing point cloud size. This may be because the larger input increases the overall model size and there may not be enough parts to properly train the larger model. Interestingly, while Transformers that are used in models such as LLaMA and SAM [26] [15] have huge numbers of parameters (13B for the former and 636M for the latter), our hyperparameter search shows that

smaller models (around 300K parameters when using lower resolution point cloud inputs) perform better than larger ones (around 1-10M parameters when using higher resolution point cloud inputs) for our task; however, LLaMA is trained on 1.4 trillion tokens and SAM is trained on 11 million images. It is possible that the higher resolution input point cloud Transformer models would perform better if trained on orders of magnitude more data, but our results suggest it is equally possible this architecture is not as well suited for AM applications as 3D CNNs.

4.4.1. Model comparison

We compare four DL pipelines across different dataset sizes and hyperparameters in the two tables below. The DL models are also compared to the baseline model. In both tables, Dataset size refers to the training dataset size. All models are validated on a separate validation dataset of size 10,000. Average and Standard Deviation L1 loss in Table 3 and L1 loss in Table 4 all represent the validation L1 loss. In Table 3, KS refers to kernel size, LR is learning rate, BS is batch size, and E is epoch. ReLU, or rectified linear unit, is the non-linear activation function used. We test both ReLU and sigmoid for the 64^3 and 256^3 voxel 3D CNN models, but ReLU almost exclusively performs better for both voxel resolutions (better on average, as well as lower standard deviations) and so we exclude sigmoid from the distance field and depth image hyperparameter tuning (Table 2 compares the difference for the 64^3 resolution). For point cloud Transformer tuning, nb is the number of Transformer blocks, TD is the transformer dimension, and k is the number of nearest neighbors checked.

Dataset size	Input resolution	Avg., Std. Dev. L1 Loss	
		ReLU	Sigmoid
45,091	64^3	0.1349, 0.0171	0.1609, 0.0397
9,041	64^3	0.1619, 0.0216	0.2086, 0.0478
911	64^3	0.2113, 0.0215	0.2446, 0.0206

Table 2: For Voxel 3D CNN DL pipelines, a comparison of average and standard deviation L1 Losses with using ReLU as the non-linear activation function or using Sigmoid as the activation function for different dataset sizes. ReLU performs almost exclusively better so we only use ReLU when training the depth image and distance field DL pipelines.

We used GTX 2080 Ti graphics cards for training the DL models. Each run uses 1 GPU and 4 Intel Xeon Skylake cores. For the mid-point baseline model, GPU acceleration is insignificant, so we perform this computation on an Intel Xenon CPU.

We summarize our DL models' results over all hyperparameters in Table 3. We show the best (measured by lowest validation L1 loss) models in Table 4. The training time reported in Table 4 is the time it takes to train the model on that particular combination of hyperparameters and validate the model on the validation dataset. The 64^3 voxel 3D CNN model trained on the 180,294 part dataset and distance field 3D CNN model trained on the 45,091 part dataset have the lowest L1 losses. The distance field 3D CNN model trained on the 9,041 part dataset and the 64^3 voxel 3D CNN model trained on 45,091 part dataset have relatively low L1 losses and take much less time to train.

The first takeaway of the model comparison is that the 3D CNN models are significantly better able to predict UPS than either the point cloud Transformer model or the mid-point baseline model. The best 3D CNN model has 9.62% error compared to 20.11% for the best point cloud Transformer model and 24.96% for the baseline model. The distance field 3D CNN model is the most promising model, although the voxel 3D CNN and depth image 3D CNN are not significantly worse. The distance field shape representation is the most promising because it has the best performing model with 9.62% error and a more than one standard deviation lower L1 loss on average than the corresponding voxel and depth image inputs (better at predicting AM print quality on average). One reason to consider another shape representation is timing. The 64^3 voxel 3D CNN model trained on 45,091 parts takes much less time to train than the 64^3 distance field 3D CNN model trained on 45,091 parts. If training time and resources are severely limited, then the 64^3 voxel shape representation is more promising.

Increasing dataset size significantly improves 3D CNN performance results and decreases hyperparameter sensitivity, mainly because the models overfit less when trained on larger datasets. Increasing data resolution also improves 3D CNN performance, but not at the same rate. For example, an approximately four times increase in dataset size improves model performance about as much as a 64 times increase in data resolution. However, there may be cases were researchers or industry engineers cannot collect more data, in which case increasing data resolution would be a valid second choice.

5. Future Work and Conclusion

This research creates a framework for comparing mechanical part DL pipelines to each other and a baseline model to improve manufactured part quality and functional performance predictions. Two natural extensions of this work are to use this framework to compare our results to additional DL architectures (e.g. deep neural operator models) and shape representations, as well as using the framework with improved label quality.

This research focuses on identifying and analyzing promising input shape representations and DL pipelines, as well as determining data requirements in this field. We find that the distance field shape representation yields the best results. However, if time or resources are severely limited, then voxels are more efficient. Increasing dataset size and input resolution both improve performance; however, dataset size has a bigger impact on performance. If collecting or using more data is not an option then it makes sense to increase input resolution.

In conclusion, we train and tune four deep learning pipelines on a range of dataset sizes and input resolutions and find that distance field and voxel 3D CNNs have the best performance. We are able to achieve an error of 9.62% compared to 24.96% from our baseline model.

Table 3: Deep learning pipeline & baseline model performance summary for each combination of pipeline, dataset size, and input resolution size.

DL pipeline	Dataset size	Input resolution	Avg. L1 loss	Std. Dev. L1 loss
Voxel - 3D CNN	180,294	64^3	0.1158	0.0148
	45,091	64^3	0.1349	0.0171
	9,041	64^3	0.1619	0.0216
	911	64^3	0.2113	0.0215
	45,091	256^3	0.1185	0.0099
	9,041	256^3	0.1469	0.0177
	911	256^3	0.1936	0.0237
Distance Field - 3D CNN	45,091	64^3	0.1196	0.0143
	9,041	64^3	0.1447	0.0167
	911	64^3	0.1838	0.0158
Depth Images (from top) - 3D CNN	45,091	64^3	0.1372	0.0096
	9,041	64^3	0.1626	0.0076
	911	64^3	0.2011	0.0155
Depth Images (from bottom) - 3D CNN	45,091	64^3	0.1341	0.0104
	9,041	64^3	0.1605	0.0104
	911	64^3	0.2043	0.0146
Point Cloud - Transformer	45,091	(2048, 3)	0.2284	0.0201
	9,041	(2048, 3)	0.2494	0.0323
	911	(2048, 3)	0.2548	0.0044
Mid-point Baseline	N/A	0	0.2496	N/A

Table 4: Best performing deep learning models (measured by lowest L1 loss) for each pipeline and dataset size

DL pipeline	Dataset size	Input res.	Hyperparameters	L1 loss	Train time
Voxel - 3D CNN	180,294	64^3	ReLU, KS=5, LR=0.001, BS=32	0.0975, E=20	4h 9m
	45,091	64^3	ReLU, KS=5, LR=0.001, BS=32	0.1167, E=20	1h 9m
	9,041	64^3	ReLU, KS=3, LR=0.001, BS=8	0.1383, E=20	19m
	911	64^3	ReLU, KS=5, LR=0.001, BS=4	0.1742, E=20	12m
	45,091	256^3	ReLU, KS=5, LR=0.0001, BS=4	0.1049, E=10	1d 19h 40m
	9,041	256^3	ReLU, KS=5, LR=0.001, BS=4	0.1239, E=20	22h 54m
	911	256^3	ReLU, KS=3, LR=0.0001, BS=4	0.1693, E=20	7h 43m
Distance Fields - 3D CNN	45,091	64^3	ReLU, KS=5, LR=0.001, BS=32	0.0962, E=20	14h 23m
	9,041	64^3	ReLU, KS=5, LR=0.001, BS=32	0.1220, E=20	25m
	911	64^3	ReLU, KS=5, LR=0.001, BS=4	0.1615, E=20	13m
Depth Image (top) - 3D CNN	45,091	64^3	ReLU, KS=5, LR=0.001, BS=32	0.1209, E=20	9h 3m
	9,041	64^3	ReLU, KS=5, LR=0.001, BS=32	0.1527, E=20	27m
	911	64^3	ReLU, KS=3, LR=0.0001, BS=4	0.1830, E=20	16m
Depth Image (bottom) - 3D CNN	45,091	64^3	ReLU, KS=5, LR=0.001, BS=4	0.1203, E=20	14h 52m
	9,041	64^3	ReLU, KS=5, LR=0.001, BS=8	0.1488, E=20	24m
	911	64^3	ReLU, KS=5, LR=0.0001, BS=4	0.1892, E=20	14m
Point Cloud - Transformer	45,091	(2048, 3)	nb=2, TD=64, k=8, LR=1e-4	0.2012, E=20	13h 44m
	9,041	(2048, 3)	nb=2, TD=32, k=8, LR=1e-4	0.2265, E=20	4h 6m
	911	(2048, 3)	nb=4, TD=32, k=8, LR=1e-3	0.2498, E=20	2h 14m
Mid-point Baseline	10,000	0	N/A	0.2496	<1m

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. Special thanks to Professor Anant Sahai, Jatearoon Boondicharern, Yifan Song, Adam Chiu, Wilson Wei, Noah Adhikari, Meshan Khosla, Nicole Lee, and Konrad Keihl for useful discussions. Finally, we would like to thank all the reviewers for their feedback, especially reviewer #3.

References

[1] Balu, A., Nallagonda, S., Xu, F., Krishnamurthy, A., Hsu, M.C., Sarkar, S., 2019. A deep learning framework for design and analysis of surgical bioprosthetic heart valves. Scientific Reports 9, 18560.

[2] Bharadwaj, A., Xu, Y., Angrish, A., Chen, Y., Starly, B., 2019. Development of a pilot manufacturing cyberinfrastructure with an information rich mechanical cad 3D model repository, in: International Manufacturing Science and Engineering Conference, American Society of Mechanical Engineers. p. V001T02A035.

[3] Bushra, J., Budinoff, H.D., 2021. Orientation optimization in additive manufacturing: Evaluation of recent trends, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers. p. V005T05A003.

[4] Chib, P.S., Singh, P., 2023. Recent advancements in end-to-end autonomous driving using deep learning: A survey. IEEE Transactions on Intelligent Vehicles 9, 103–118.

[5] Dawson-Haggerty et al., 2017-2025. trimesh. URL: https://trimesh.org/.

[6] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE. pp. 248–255.

[7] Deng, L., 2012. The MNIST database of handwritten digit images for machine learning research [best of the web]. IEEE signal processing magazine 29, 141–142.

[8] Dhore, G., Jagtap, R., Bhakad, S., Yadav, P., Sutar, M., Sawrate, M.S., 2021. Exploring 3D printing using cura: A slicing software. International Journal of Advance Scientific Research Engineering Trends 5, 212–222.

[9] Di Angelo, L., Di Stefano, P., Guardiani, E., 2020. Search for the optimal build direction in additive manufacturing technologies: a review. Journal of Manufacturing and Materials Processing 4, 71.

[10] Dong, G., Wong, J.C., Lestandi, L., Mikula, J., Vastola, G., Jhon, M.H., Dao, M.H., Kizhakkinan, U., Ford, C.S., Rosen, D.W., 2022. A part-scale, feature-based surrogate model for residual stresses in the laser powder bed fusion process. Journal of Materials Processing Technology 304, 117541.

[11] Eranpurwala, A., Ghiasian, S.E., Lewis, K., 2020. Predicting build orientation of additively manufactured parts with mechanical machining features using deep learning, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers. p. V11AT11A023.

[12] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778.

[13] Jin, Z., Zheng, B., Kim, C., Gu, G.X., 2023. Leveraging graph neural networks and neural operator techniques for high-fidelity mesh-based physics simulations. APL Machine Learning 1.

[14] Khadilkar, A., Wang, J., Rai, R., 2019. Deep learning–based stress prediction for bottom-up SLA 3D printing process. The International Journal of Advanced Manufacturing Technology 102, 2555–2569.

[15] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., et al., 2023. Segment anything, in: Proceedings of the IEEE/CVF international conference on computer vision, pp. 4015–4026.

[16] Krizhevsky, A., Hinton, G., 2009. Learning multiple layers of features from tiny images URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[17] Kumar, S., Gopi, T., Harikeerthana, N., Gupta, M.K., Gaur, V., Krolczyk, G.M., Wu, C., 2023. Machine learning techniques in additive manufacturing: a state of the art review on design, processes and production control. Journal of Intelligent Manufacturing 34, 21–55.

[18] Liang, L., Liu, M., Martin, C., Sun, W., 2018. A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis. Journal of The Royal Society Interface 15, 20170844.

[19] Malviya, M., Desai, K., 2019. Build orientation optimization for strength enhancement of fdm parts using machine learning based algorithm. Computer-Aided Design and Applications 17, 783–796.

[20] Min, P., 2004 - 2025. binvox. URL: http://www.patrickmin.com/binvox. Accessed: 2022-09-14.

[21] Najafabadi, M.M., Villanustre, F., Khoshgoftaar, T.M., Seliya, N., Wald, R., Muharemagic, E., 2015. Deep learning applications and challenges in big data analytics. Journal of Big Data 2, 1–21.

[22] Nie, Z., Jiang, H., Kara, L.B., 2020. Stress field prediction in cantilevered structures using convolutional neural networks. Journal of Computing and Information Science in Engineering 20, 011002.

[23] Nooruddin, F.S., Turk, G., 2003. Simplification and repair of polygonal models using volumetric techniques. IEEE Transactions on Visualization and Computer Graphics 9, 191–205.

[24] Pandey, P., Reddy, N.V., Dhande, S., 2007. Part deposition orientation studies in layered manufacturing. Journal of Materials Processing Technology 185, 125–131.

[25] Schranz, C., 2016. Tweaker-auto rotation module for FDM 3D printing. Salzburg Research Salzburg, Austria .

[26] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G., 2023. Llama: Open and efficient foundation language models. arXiv:2302.13971.

[27] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. Advances in Neural Information Processing Systems 30.

[28] Wang, C., Tan, X., Tor, S.B., Lim, C., 2020. Machine learning in additive manufacturing: State-of-the-art and perspectives. Additive Manufacturing 36, 101538.

[29] Wang, J., Das, S., Rai, R., Zhou, C., 2018. Data-driven simulation for fast prediction of pull-up process in bottom-up stereo-lithography. Computer-Aided Design 99, 29–42.

[30] Wang, R., Lei, T., Cui, R., Zhang, B., Meng, H., Nandi, A.K., 2022. Medical image segmentation using deep learning: A survey. IET Image Processing 16, 1243–1267.

[31] Williams, G., Meisel, N.A., Simpson, T.W., McComb, C., 2019. Design repository effectiveness for 3D convolutional neural networks: Application to additive manufacturing. Journal of Mechanical Design 141, 111701.

[32] Wohlers, T., International, A., 2024. Wohlers Report 2024: 3D Printing and Additive Manufacturing : Global State of the Industry. Wohlers Associates. URL: https://books.google.com/books?id=0j7i0AEACAAJ.

[33] Wong, J.C., Ooi, C.C., Chattoraj, J., Lestandi, L., Dong, G., Kizhakkinan, U., Rosen, D.W., Jhon, M.H., Dao, M.H., 2022. Graph neural network based surrogate model of physics simulations for geometry design, in: 2022 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE. pp. 1469–1475.

[34] Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V., 2021. Point transformer, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 16259–16268.