Proceedings of the ASME 2020 International Design Engineering Technical Conferences &
Computers and Information in Engineering Conference
IDETC/CIE 2020
August 18-21, 2020, St. Louis, MO, United States

IDETC2020-22518

DERIVING METAMODELS TO RELATE MACHINE LEARNING QUALITY TO DESIGN REPOSITORY CHARACTERISTICS IN THE CONTEXT OF ADDITIVE MANUFACTURING

Glen Williams
Mechanical Engineering
The Pennsylvania State University
University Park, PA, USA
gtw5020@psu.edu

Nicholas A. Meisel
Engineering Design
The Pennsylvania State University
University Park, PA, USA
nam20@psu.edu

Timothy W. Simpson
Mechanical Engineering
The Pennsylvania State University
University Park, PA, USA
tws8@psu.edu

Christopher McComb
Engineering Design
The Pennsylvania State University
University Park, PA, USA
mccomb@psu.edu

ABSTRACT

The widespread growth of additive manufacturing, a field with a complex informatic "digital thread", has helped fuel the creation of design repositories, where multiple users can upload distribute, and download a variety of candidate designs for a variety of situations. Additionally, advancements in additive manufacturing process development, design frameworks, and simulation are increasing what is possible to fabricate with AM, further growing the richness of such repositories. Machine learning offers new opportunities to combine these design repository components' rich geometric data with their associated process and performance data to train predictive models capable of automatically assessing build metrics related to AM part manufacturability. Although design repositories that can be used to train these machine learning constructs are expanding, our understanding of what makes a particular design repository useful as a machine learning training dataset is minimal. In this study we use a metamodel to predict the extent to which individual design repositories can train accurate convolutional neural networks. To facilitate the creation and refinement of this metamodel, we constructed a large artificial design repository, and subsequently split it into sub-repositories. We then analyzed metadata regarding the size, complexity, and diversity of the sub-repositories for use as independent variables predicting accuracy and the required training computational effort for training convolutional neural networks. The networks each predict one of three additive manufacturing build metrics: (1) part mass, (2) support material mass, and (3) build time. Our results suggest that metamodels predicting the convolutional neural network coefficient of determination, as opposed to computational effort, were most accurate. Moreover, the size of a design repository, the average complexity of its constituent designs, and the average and spread of design spatial diversity were the best predictors of convolutional neural network accuracy.

1. INTRODUCTION

Additive manufacturing (AM) has increased the diversity of geometries that can be designed and manufactured. A "digital thread", or information pathway, may be created when using an AM technology [1]. In this digital thread, a component's design, which begins as a concept and associated user needs and specifications, is represented by one or more 3D models before finally being transformed into machine motion commands transmitted to an AM system [1]. Along the way, rich and diverse data is produced at each step. These data, stored in various digital files (e.g. point clouds, CAD models, STL files, G-Code), may be saved in a design repository for further analysis. Prior to the rise of AM and other digitally-driven design and manufacturing techniques, learning from past designs was a predominantly manual process, often limited by what could be discerned from paper 2D engineering drawings. Digital design repositories, on the other hand, allow vast stores of geometric data to be rapidly searched, sorted, shared, and built upon [2].

These data, which are often geometric, can be useful for training machine learning (ML) tools to provide feedback related to AM components, such as in the automatic estimation of related build metrics [3–7]. ML-powered feedback tools may prove especially useful as accompaniments to AM heuristics [8,9], frameworks [10], educational tools [11], and methods [12,13] that seek to help designers make the most of AM's unique opportunities. These diverse areas of active AM design research illustrate the natural link between AM and ML.

Although the AM-ML intersection shows promise, popular ML algorithms have been increasing both in terms of complexity and computational cost [14], presenting an ongoing challenge to those seeking to apply ML to new areas. Additionally, AM design repositories vary widely in their size, the geometric complexity of their constituent 3D solid models, the nature of their relevant applications, and the skill level of their contributors. *There is currently no benchmark training design repository that meets the needs of AM-ML*, and thus researchers and practitioners will need to rely on diverse, real-world datasets. An improved theoretical understanding of how well design feedback algorithms are likely to perform when trained on diverse AM datasets will enhance the AM community's ability to develop more useful AM-ML tools. Such tools would help engineers design more manufacturable and cost-efficient AM parts by allowing them to use estimates about cost and manufacturability to inform design iterations. In this work, we research this relationship between AM datasets and design feedback algorithms. Specifically, we study a 3D convolutional neural network (CNN) designed to analyze AM build metrics of part mass, support material mass, and build time. Rather than focusing solely on the CNN performance, however, we create and refine a metamodel to predict how effective datasets with varying, quantifiable metadata attributes are when used as training data. With this metamodel, we can then investigate the following two research questions:

1. To what extent do the size, complexity, and diversity of a design repository affect the ability of a 3D CNN trained using that repository to predict part mass, build time, and support material usage for unseen designs?

2. To what extent does a derived metamodel accurately predict the performance of 3D CNNs using summary data related to the size, complexity, and diversity of the training and testing datasets?

The remainder of the paper is organized as follows. First, we discuss relevant literature regarding the use of artificial neural networks (ANN) and design repositories to analyze AM designs. Next, we describe our methodology to systematically create and analyze design repositories, build ML algorithms from their constituent designs, and use a linear regression metamodel to make quantitative predictions about the usefulness of different design repositories for CNN training. Finally, we discuss the results of our metamodel fitting and evaluation procedure and use those results to draw conclusions about how this metamodeling approach may be applied to industrial and academic design repositories in future work.

2. BACKGROUND

In this section we synthesize relevant prior contributions in the literature relating to artificial neural networks in AM (see Section 2.1) and design repositories used for AM research (see Section 2.2). We then highlight how the intersection of these fields creates an important research gap in the study of AM informatics and AM-ML applications.

2.1. Artificial Neural Networks for AM

The assorted files produced during AM design efforts present many opportunities for AM-ML research [15]. Specifically, artificial neural networks (ANN) are a popular type of ML algorithm that have been applied to AM in the literature.

Many AM-ML efforts aim to make predictions about new, arbitrary designs by learning generalizable patterns from repositories of existing designs [16]. It is important for AM analytical tools to learn patterns and solutions for general, diverse inputs because of the vast complexity of designs that are possible through AM [17]. These tools differ in input data type, ML algorithm choice, and analysis objective. In terms of input data, both design and in-situ processing measurements are commonly used. Design data typically includes either single metrics, such as part volume [18] or cost [3], geometric shapes (e.g. voxel-based model [5,6]), or machine toolpath instructions (e.g. G-code text [3,19]). In-situ processing data, which is gathered during a real or simulated fabrication, often includes thermal data from in-progress builds [20], such as infrared images [21], image data [22], video data [7,23], and/or surface topography scans [24]. The data format of choice and its dimensionality tends to influence which ML algorithms are most effective for a given task. For instance, data with fewer dimensions are often analyzed with techniques such as fully-connected neural networks [18], clustering, and regression [3], whereas higher dimensional data are typically analyzed with different techniques, particularly CNNs in recent years [5,6,25–27].

Generally, ANNs are advantageous for use in AM analysis due to the wide variety of data dimensionality and formats that may be used as inputs or outputs for ANN architectures [16]. In this study we specifically focus on the 3D CNN. The CNN is an advantageous baseline for AM design repository research because of its recent popularity in both AM-ML research and 3D ML research in general. Additionally, the majority of AM fabricated components are modeled in 3D at some point, allowing them to either be input layer-by-layer into 2D CNNs or as an entire volume into 3D CNNs [16]. CNNs, which are capable of learning spatially-relevant patterns within high dimensional inputs, such as pixel-based images or voxel-based volumes, have rapidly spread through many academic fields within the last decade [16,28]. Contemporary success in 3D fields was preceded by an explosion of highly performing image classification CNN architectures [28–31]. 2D and 3D CNNs, both of which have already been successfully used to analyze AM parts [5,25–27] are likely to continue to be used extensively in AM-ML.

2.2. AM Design Repositories

The growth of AM-ML research has been accompanied by similar advancements in AM design repositories. Repositories may be generally classified as those that are intentionally created and those that are produced as a byproduct of other work. Many design repositories that are intentionally created for or are applicable to AM draw influence from early design repository work that was completed in the 1990s. The NIST design repository, created by Regli et al. [2] and Szykman et al. [32], aimed to develop enhanced product design techniques that leveraged central repositories and were better-suited to handle design efforts spread over large geographic areas or long timelines, as occurs with modern outsourcing [32]. Despite many design repositories not being created specifically for AM, we believe they are still relevant to AM because of their tendency to contain 3D computer-aided design (CAD) data and AM's general ability to manufacture geometries that were not originally designed for it but are subjected to an AM redesign effort [33]. General design repository research has investigated diverse topics including but not limited to optimum data schema for mechanical components [34], the extent to which such design repositories impact design efforts and concept generation when used as a tool [35–37], and enhancing design repository systems to promote web-based search and networking activities [38].

Benchmark datasets are common in ML research, providing a consistent subset of the global data domains for researchers to compare machine learning results on [39] and eliminating the need to manually obtain data for each new research effort. For some fields, like image classification, one single type of data, such as ImageNet's annotated images [39], can be extensively useful for the long-term, providing a method for researchers to reliably and consistently compare the performance of their novel ML algorithms [39]. Many research efforts use such benchmarks to test image classification tools [40]. As mentioned in Section 2.1, some 2D classification techniques can be adapted to 3D classification algorithms [31]. These types of efforts have driven the formation of annotated 3D datasets, including ShapeNet [30] and ModelNet [41]. Such datasets provide ML researchers with geometric representations of common objects and hierarchical attribute labels that may be used to train ML classifiers [30,41].

Although hierarchically-labeled 3D object repositories have many beneficial research applications, they do not always contain sufficient data or organizational structure for 3D AM-ML applications. For instance, classification benchmarks tend to contain models designed for aesthetic representation [30], possibly making them less realistic for AM. Furthermore, lack of consistent scale and dimensional specification across designs [30] could be a challenge for conducting AM-ML analysis, which is often grounded by the build volume of the AM process. Also, lacking design intent data could hinder the automated understanding of what a shape's intended function is [42]. Other recent efforts have focused on improving the end-to-end CAD data gathering, storage, and access challenges associated with design repositories. FabWave is intended to support data-driven research of CAD parts by compiling web-scraped CAD models from free hosting websites and prototyping CAD tool add-ins to support crowd-sourced model submission [43]. Additionally, other researchers use artificial design repositories (ADR), or design repositories containing manufacturable components that were not originally intended for fabrication [5], to bypass data-gathering difficulties during early-stage ML research efforts [5,27]. As the drive to use ML for AM analysis grows, we anticipate further expansion, diversification, and improvement to AM-relevant design repositories.

2.3. AM Informatics and ML Research Gap

At first glance, using design repositories to train supervised ML tools may appear to be a straightforward process: data is split into one or more groups of training and validation sets which are used to iteratively optimize the adjustable parameters of a particular ML architecture, such as an ANN, until a desirable accuracy is reached [16]. Although this training process, once begun, may be largely automatic [44], many 3D analytical ML architectures require large training sets on the order of thousands of input geometries [5,31,42]. Manually compiling such large repositories for design may be time-consuming [45,46], and the computational and power consumption of training can be undesirably high [14]. Additionally, the final accuracy of an ANN may be hard to predict without computationally-intensive evaluation against an extensive validation dataset [47].

Lack of reliable design repository heuristics and theoretical understanding could cause inconsistency and suboptimality in the development of AM-ML. Also, if more industries begin investigating the use of ML to achieve increasingly specialized goals, then they may seek to use their existing design portfolios to drive the creation of useful AI tools. Compiling, selecting, and prioritizing datasets to use for AM-ML experimentation could prove inefficient and costly if done ad hoc, potentially hindering the success of those development efforts. However, this preparatory stage is critically important for downstream efforts. Being able to quickly and reliably predict the accuracy and training time of an AM-ML construct at the outset of a project could increase the efficiency of AM-ML development by ensuring computational resources are devoted only to attempts that are most likely to be successful. Knowing accuracy in advance is useful to anticipate if the overall goals of the trained AM-ML construct will be met. Estimating training time can provide decision makers with the knowledge needed to understand if their limited resources can handle the additional development cost. Although predicting both concurrently would be ideal, these predictions are also useful individually.

The rate and quality of AM-ML technological development will be increased through a better understanding of the design repositories to be used as training and testing datasets. In this study, we generate metamodels that relate design repository characteristics to measures of the quality of trained ML models. First, we implement an analytical methodology to model the design repositories using quantitative metadata that concisely describes the overall size, complexity, and diversity of the repositories. We then empirically test the performance of design repositories when used as training datasets for 3D CNN algorithms and use the results to train an ordinary least squares (OLS) metamodel. This metamodel can then be used to predict CNN training performance of a new design repository.

3. METHODOLOGY

We developed our design repository characterization metamodel using an analytical theory-building approach that leveraged synthetic data. This section details the artificial design repository (see Section 3.1), the AM build metrics used to evaluate each part in the artificial design repositories (see Section 3.2), the machine learning algorithm used to predict AM build metrics based on part geometry (see Section 3.3), and the OLS metamodel used to relate design repository characteristics to expected CNN accuracy and training time (see Section 3.4). Figure 1 shows a visualization of the overall methodology.

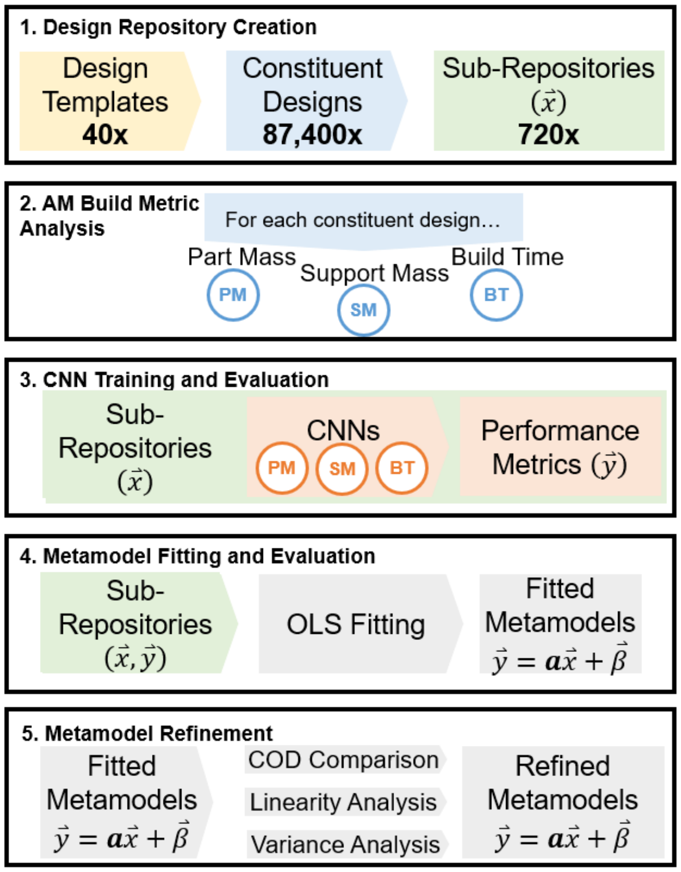
1. Design Repository Creation
Design Templates 40x
Constituent Designs 87,400x
Sub-Repositories $(\vec{x})$ 720x
2. AM Build Metric Analysis
For each constituent design…
Part Mass PM
Support Mass SM
Build Time BT
3. CNN Training and Evaluation
Sub-Repositories $(\vec{x})$
CNNs PM SM BT
Performance Metrics $(\vec{y})$
4. Metamodel Fitting and Evaluation
Sub-Repositories $(\vec{x},\vec{y})$
OLS Fitting
Fitted Metamodels $\vec{y} = \boldsymbol{a}\vec{x} + \vec{\beta}$
5. Metamodel Refinement
Fitted Metamodels $\vec{y} = \boldsymbol{a}\vec{x} + \vec{\beta}$
COD Comparison
Linearity Analysis
Variance Analysis
Refined Metamodels $\vec{y} = \boldsymbol{a}\vec{x} + \vec{\beta}$

Figure 1 Visualization of the methodology for the study.

3.1. Artificial Design Repository

The artificial design repository (ADR) used here was created as a representative analog of design repositories that exist today but with a more structured, controllable format that is well-suited for early-stage, AM-ML research. It emulates possible real-world repositories of 3D mechanical part CAD models in size, complexity, and diversity [48]. In order to represent the breadth of possible repositories, a single large repository with many unique geometries was assembled. This large repository was then strategically sliced to create sub-repositories which could be systematically analyzed for metadata attributes and performance as ML training sets. For the purposes of this research, we characterize each sub-repository according to the number of designs, the complexity of those designs, and the diversity across the set of designs in the sub-repository.

The constituent designs were created using a procedural, template-based CAD generation process (see Figure 2), similar to the one used by Williams et al. [5], but with the Autodesk Fusion 360 C++ API [49] . Although the number of dimensions required to specify a single design for a given design template was relatively low, the number of possible combinations is essentially infinite, enabling us to generate many diverse geometries relatively quickly.

Each design was first modeled as a parametric CAD file, then converted to an STL file, before being converted to a 64 × 64 × 64 voxel model, in which voxels may only be coded "1.0" to indicate material presence or "0.0" to indicate material absence. The STL file was used to analyze design complexity and the AM build metrics, and the voxel-based model was used for spatial diversity analysis and as an input to the 3D CNNs.

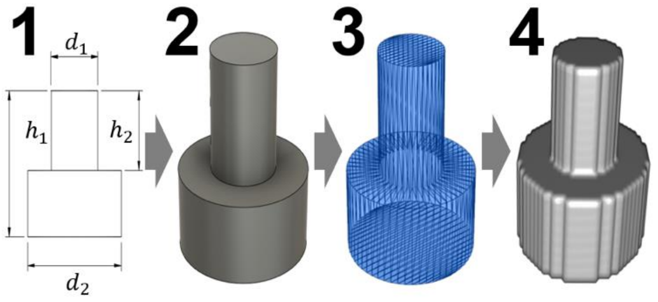

Figure 2 Visualization of the digital pathway from a set of dimensions specifying a particular embodiment of a two-stepped cylinder design template (1), a Fusion 360 CAD model (2), an STL file (3), and a voxel-based model (4).

In addition to the templates used by Williams et al. [5], templates representing shells, boxes, and hole patterns were included. Templates with more complex internal geometries were also added as such features are particularly important to DfAM, in which lattice structures and infill patterns are common. In total, 40 unique templates were equally represented across the 84,700 unique geometries that were created. Every geometry was also rotated to a random orientation per the impact of the orientation on the results found by Williams et al. [5]

Once a large design repository was created, sub-repositories were selectively assembled from the superset. A uniform random number generator was used to choose 20 to 5,000 individual designs to be included in each of the sub-repositories. Individual repositories were not limited to contain uniform distributions of particular design templates, in an attempt to simulate variety existing in human-curated repositories. In total, 720 different sub-repositories were created. These sub-repositories themselves served as data points for fitting and assessing metamodels that would predict attributes of other repositories.

In this study, we chose seven metadata values to represent the characteristics of the sub-repositories, providing varying measures of the sub-repository size, complexity, and diversity. The design repository size, s, is the min-max normalized number of unique geometries in the repository. We chose to include s because real-world design repositories can vary substantially in size, and prior work in the ML literature suggests that training dataset size is one of the most important factors when creating accurate CNNs [50].

In addition to size, we wanted to include design repository values that were specific to the geometric analysis task studied. We chose measures of geometric complexity, C_{PR} and C_{AR}, and spatial diversity D. These complexity measures were adapted from prior design complexity literature by Conner et al. [51]. The part volume ratio, C_{PR}, was calculated for each geometry j as

$$C_{PR,j} = 1 - \frac{V_p}{V_b}$$

where V_p is the part volume and V_b is the part bounding box volume. The part surface area ratio, C_{AR}, was calculated for each geometry j as

$$C_{AR,j} = 1 - \frac{A_s}{A_p}$$

where A_s is the surface area of a sphere with the same volume as the part and A_p is the part surface area. All part volumes and surface areas were calculated from STL approximations of the geometries (see Section 2.2).

Spatial diversity, D, was calculated using the voxel-based model instead of the STL. D was used to assess how much individual designs filled the cubic voxel space. Williams et al. [5] found that diversity of the voxel representations of 3D designs in a training dataset can affect the performance of a 3D regression CNN, leading us to study a version of that metric as a potential metadata attribute that might be critical to training dataset effectiveness. D was calculated by flattening the 64 × 64 × 64 voxel-based model into a 1D vector, then taking the squared Euclidean distance of that vector from the voxel space origin.

For each of these metrics, we computed a median value and interquartile range to represent the central-tendency and spread of that quantity for the given design repository, respectively. Combined with repository size, these seven quantities (s, $C_{PR,m}$, $C_{PR,r}$, $C_{AR,m}$, $C_{AR,r}$, D_m, D_r) effectively characterize a given design repository.

3.2. AM Build Metrics

After the large artificial design repository was created, all constituent geometries were analyzed for three AM build metrics: (1) part mass, (2) support material mass, and (3) build time assuming a material extrusion AM process. Part mass is the amount of material in grams present in the final, manufactured component after any support material has been removed. Support material mass is the amount of material in grams that is required to be deposited beneath overhanging structures to keep them from collapsing during fabrication. Build time is the amount of time in hours necessary for the AM machine to fabricate the part, approximated as a weighted linear combination of the part mass, support material mass, and part height. All build metric values were calculated based on STL files, and were calculated using the methods presented by Williams et al. [5], which assumed a polylactic acid (PLA) material and simplified AM processing speeds. These build metrics served as the ground-truth values when training 3D CNN models to analyze the same metrics based on voxel inputs (see Section 3.3).

The three metrics were specifically chosen because they may be quickly calculated using alternative algorithms and inputs to the 3D CNNs and have been studied in prior AM estimation and design repository literature [5,6,52]. Although build metric values will vary depending on the specific AM process, machine, and material used, we chose to assume these metrics would be calculated for an idealized, generic material extrusion process of the same properties as the one analyzed by Williams et al. [5]. Using a generalized AM process is acceptable for this study because we seek general theory related to AM-ML, and not particular machines, processes, or materials. The proposed method could be repeated with new ground truth data that is more closely related to specific AM processes (e.g. metal powder-bed fusion) to make results more relevant for use.

3.3. Machine Learning Algorithm

3.3.1. 3D CNN Architecture

For this study, we used a 10-layer neural network containing a convolutional section and a fully-connected section (see Figure 3), the same CNN from our previous work [5]. It accepts a 64 × 64 × 64 voxel model input, and the voxelized design space was 10 cm × 10 cm × 10 cm large, an arbitrary size that represents a build volume applicable to many desktop material extrusion AM systems. The CNN was developed and trained using TensorFlow version 1.12 [53], Keras version 2.2.4 [54], Python version 3.6.5, and the "Adam" Optimizer [55]. All training was completed on an NVIDIA P6000 Graphics Processing Unit.

For this study, the 3D CNNs were trained using an early stop condition. After three consecutive epochs of training in which the validation accuracy did not improve, the 3D CNN was considered trained, and its accuracy and the total number of epochs were recorded.

3.3.2. CNN Performance Metrics

Each individual CNN that was trained was capable of predicting only one of the three AM build metrics described in Section 3.2 at a time. All sub-repositories were thus used to train three separate CNNs to cover all build metrics. Each of these trained CNNs were then analyzed for two CNN performance metrics. The first performance metric was CNN coefficient of determination (CNN-COD), shortened here with the symbol $\hat{a}$.

CNN-COD describes the total fraction of a build metric's variation that is explained by the particular trained CNN for a given repository and is used here as an indication of accuracy. A 75%/25% train-test split was used to calculate the CNN-COD.

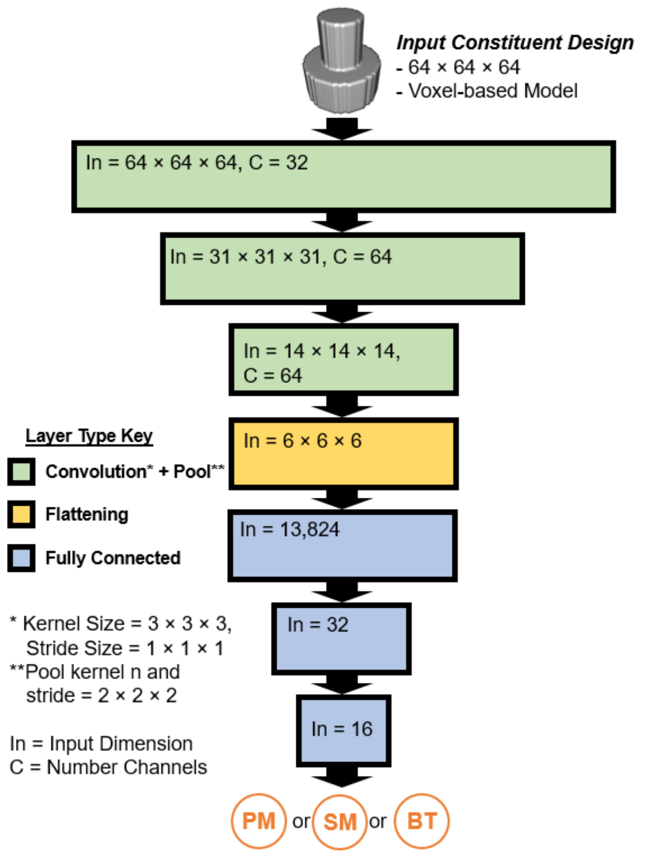

Figure 3 Diagram of the CNN architecture used in the study.

The second performance metric recorded was the number of epochs required to achieve convergence (ETC) as determined by our early stop condition. This value, abbreviated $\hat{t}$, was intended to be a surrogate measure for the amount of computational effort required to achieve convergence. Note that the raw data for both $\hat{a}$ and $\hat{t}$ are specific to the build metric predicted by the particular CNN for which they are reported. For instance, $\hat{a}_{PM}$ describes the COD of a CNN that predicts part mass based on a component's voxelized geometric input.

3.3.3. Aggregate CNN Performance Metrics

In addition to describing a trained CNN's accuracy and computation time, we also employed metrics to characterize a CNN's comparative quality in more general ways. Six aggregate CNN performance metrics were calculated to achieve this goal. By fitting metamodels that also predict these aggregate CNN performance metrics, we aim to study metamodels that offer more generalized predictions than those relating to only one combination of CNN performance metric and AM build metric at a time. The aggregate metrics were found by averaging normalized CNN performance metrics associated with a given design repository. The general form of the CNN performance metric aggregation formula was

$$\hat{p}_i = \left(\frac{1}{n}\right)\sum_{k=0}^{n}\frac{\hat{P}_{k,i}-\hat{P}_{k,min}}{\hat{P}_{k,max}-\hat{P}_{k,min}}$$

where k is the index of the particular performance metric, n is the total number of CNN performance metrics being aggregated, and $\hat{P}_k$ is either $\hat{a}$ or $\hat{t}$ for a given build metric. Variables subscripted min are the smallest values recorded across sub-repositories for that metric and variables subscripted max are the largest values recorded across all sub-repositories.

Two of the aggregate CNN performance metrics were found by taking the mean of the same CNN performance metrics across the three build metrics. These were the aggregate CNN-COD performance metric, $\hat{p}_a$, and the aggregate CNN epochs to convergence performance metric, $\hat{p}_t$. Each individual CNN performance metric was min-max normalized based on the overall sub-repository ranges prior to aggregation. For this metric, we chose to use larger aggregate scores to be indicative of CNNs that were higher performing overall. We used one minus min-max normalized $\hat{t}$ to align the direction of better CNN performance for both $\hat{a}$ and $\hat{t}$.

Three additional aggregate CNN performance metrics were found by taking the mean of the two original CNN performance metrics, $\hat{a}$ and $\hat{t}$, for the same build metric. Thus, individual aggregates specific to part mass ($\hat{p}_{PM}$), support material mass ($\hat{p}_{SMM}$), and build time ($\hat{p}_{BT}$) CNNs were calculated. Finally, we also investigated an overall aggregate CNN performance metric, $\hat{p}_{ALL}$, found by taking the mean of the other aggregate build metrics. These six values ($\hat{p}_a$, $\hat{p}_t$, $\hat{p}_{PM}$, $\hat{p}_{SMM}$, $\hat{p}_{BT}$, $\hat{p}_{ALL}$) characterize a design repository's performance at training CNNs.

3.4. Linear Metamodel Construction

One of the goals of the current study is to demonstrate models that could predict effectiveness of a design repository as an ML training dataset. The set of independent variables for this fit are the summative characteristics that describe each design repository (e.g., size, median part volume ratio, etc.). The dependent variables in this fit are the coefficient of determination (COD) of the CNN trained with the design repository and the number of epochs required to achieve convergence. To produce a metamodel linking these variables, we chose to use OLS regression.

The initial OLS metamodel is comprised of the twelve-response, multiple linear regression formula given by

$$\vec{y} = \boldsymbol{\alpha}\vec{x} + \vec{\beta}$$

where $\vec{y}$ is the vector of dependent variables predicted by the metamodel, $\vec{x}$ is the vector of independent variables that characterize a given design repository, $\boldsymbol{\alpha}$ are the coefficients found by the OLS fitting algorithm, and $\vec{\beta}$ are the intercepts. The independent and dependent variable vectors for a given sub-repository i was formed from the metadata attributes and CNN performance metrics as follows:

$$\vec{x}_i = [s_i \quad C_{PR,m,i} \quad C_{PR,r,i} \quad C_{AR,m,i} \quad C_{AR,r,i} \quad D_{m,i} \quad D_{r,i}]^T$$

$$\vec{y}_i = [\hat{a}_{PM,i} \ \hat{t}_{PM,i} \ \hat{a}_{SMM,i} \ \hat{t}_{SMM,i} \ \hat{a}_{BT,i} \ \hat{t}_{BT,i} \ \hat{p}_{a,i} \ \hat{p}_{t,i} \ \hat{p}_{PM,i} \ \hat{p}_{SMM,i} \ \hat{p}_{BT,i} \ \hat{p}_i]^T$$

where s is the repository size, $C_{PR,m}$ is the median part volume complexity, $C_{PR,r}$ is the interquartile range of part volume complexity, $C_{AR,m}$ is the median part area complexity, $C_{AR,r}$ is the interquartile range of part area complexity, D_m is the median squared Euclidean distance, and D_r is the interquartile range of squared Euclidean distance. All of the independent variable values were min-max normalized prior to use in the metamodel.

This initial metamodel was fit as eight individual OLS regression models, one response variable at a time. The Python Scikit-learn package version 0.22.1 was used for OLS fitting [56]. A 10-fold cross validation approach was used for fitting and evaluating each model. The individual, fitted linear regressions were then analyzed and compared to one another in terms of their mean coefficient of determination, averaged across all ten cross validation runs, when predicting their respective CNN performance metrics. The independent variables were also analyzed to determine whether they had a statistically significant linear correlation with each of the dependent variables through calculation of the Pearson correlation coefficients. Finally, the explained variance contribution of each independent variable was found using a sequential sum of squares approach in an ANOVA analysis. The statistical tests were completed using the Python statsmodels package version 0.9.0 [57]. The results are analyzed next.

4. RESULTS

This section describes the results from the metamodel fitting and evaluation process (see Section 4.1), the statistically-driven down selection process of the different metamodels investigated (see Section 4.2), and the final selection of the most effective, single metamodel.

4.1. Metamodel Data Gathering, Fitting, and Evaluation

All sub-repositories were used to train CNNs to predict part mass, support material mass, and build time as part of the metamodel fitting dataset creation process. We observed a wide range of CNN-COD (see Figure 4) and ETC results (see Figure 5), indicating that these data provide a good basis on which to fit a metamodel. In general, CNNs trained to predict part mass were the most accurate, with a median CNN-COD of 0.98. Build time CNNs were the second most effective on average, exhibiting a median COD of 0.92. Support material mass was consistently the most challenging build metric for the CNNs to predict, with a median COD of 0.47. This generally aligns with previous results found by Williams et al. [5].

Different output build metrics also resulted in different average CNN training durations. Part mass and build time were the longest training durations, with medians of 21 and 22 epochs. Support material mass CNNs converged faster, with a median of 14 epochs.

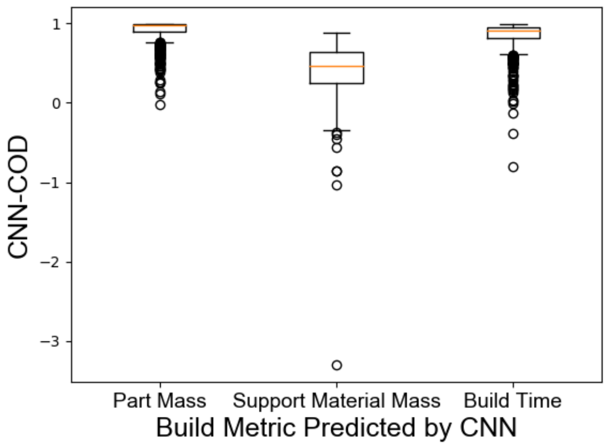

Figure 4 Box plots of CNN-COD results for all sub-repositories grouped by build metric predicted by the CNN.

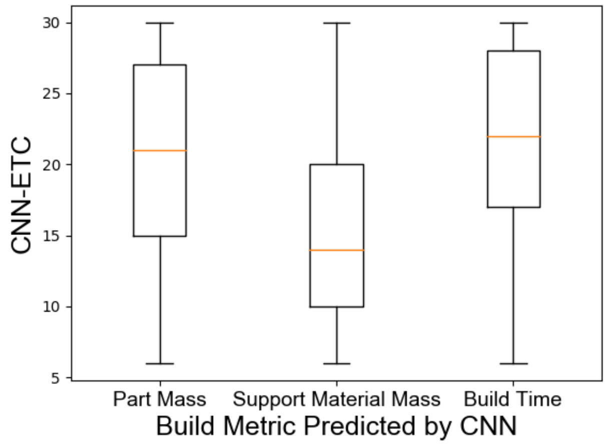

Figure 5 Box plots of ETC results for all sub-repositories grouped by build metric predicted by the CNN.

After training the CNNs with each of the sub-repositories and calculating aggregated CNN performance metrics, we analyzed the univariate Pearson linear correlation values between each of the metamodel independent variables and the CNN performance metrics. The associated p-values from the univariate Pearson analysis were then used to determine statistical significance of the linear correlations of each variable using an alpha value of 0.05 (see Table 1). Although this linearity estimate does not describe how well a trained linear model will perform, it does provide insight into whether fitting a linear model is likely to be sufficient.

Table 1 Univariate Pearson correlation values that were used to analyze the relationship between each independent variable and the dependent variables. Bolded, highlighted values indicate that the relationship was significantly linear (p < 0.05).

	$\vec{x}$ Variable Pearson Correlation Coefficient						
$\vec{y}$	s	$C_{PR,m}$	$C_{PR,r}$	$C_{AR,m}$	$C_{AR,r}$	D_m	D_r
$\hat{a}_{PM,i}$	**0.283**	**0.094**	0.037	-0.034	0.067	**-0.076**	**0.159**
$\hat{t}_{PM,i}$	**-0.323**	-0.015	-0.017	**-0.090**	0.009	0.020	0.016
$\hat{a}_{SMM,i}$	**0.504**	-0.018	-0.011	0.019	0.026	-0.052	0.062
$\hat{t}_{SMM,i}$	**0.393**	-0.041	0.022	-0.019	0.050	**-0.079**	0.034
$\hat{a}_{BT,i}$	**0.327**	**0.138**	0.041	-0.021	0.036	**-0.112**	0.041
$\hat{t}_{BT,i}$	-0.065	0.052	0.048	0.039	0.020	-0.009	-0.036
$\hat{p}_{a,i}$	**0.432**	**0.100**	0.033	-0.021	0.058	**-0.101**	**0.122**
$\hat{p}_{t,i}$	0.025	0.001	-0.031	0.045	-0.046	0.037	-0.008
$\hat{p}_{PM,i}$	**0.449**	0.058	0.033	0.073	0.022	-0.054	0.058
$\hat{p}_{SMM,i}$	**-0.281**	0.041	-0.029	0.028	-0.049	0.072	-0.018
$\hat{p}_{BT,i}$	**0.190**	-0.003	-0.035	-0.049	-0.007	-0.032	0.053
$\hat{p}_i$	**0.264**	0.057	-0.013	0.033	-0.014	-0.018	0.060

As shown in Table 1, repository size was frequently correlated with the CNN performance metrics. Size was significantly correlated in 10 out of 12 possible metamodel dependent variable types, and it had the largest correlation magnitudes when compared to other independent variables in all significant cases. Conversely, $C_{PR,r}$ and $C_{AR,r}$, the interquartile ranges of part volume ratio complexity and part area ratio complexity, were not significant in any of the correlations. Part mass and build time CNN-COD metamodels were also often linearly correlated with independent variables for the non-aggregate metamodels. In general, ETC metamodels tended to have lower numbers of statistically significant linear relationships than CNN-COD metamodels. Together, these results suggest that only a subset of the metamodel dependent variable outputs were likely to be well-predicted by the linear fit.

The k-fold cross validation results for the linear metamodel were averaged and are shown in Figure 6. Each of the first three rows of Figure 6 shows the metamodel predictive performances for a different build metric. The bottom row of Figure 6 shows the results for the individual normalized responses averaged across all three build metrics. The first two columns of Figure 6 detail the COD values for each CNN performance metric. The rightmost column of Figure 6 shows the COD values when the metamodel is predicting the aggregate, normalized CNN performance metrics.

A variety of metamodel performance levels were observed in this study. Overall, 9 out of 12 metamodel combinations exhibited CODs greater than zero, indicating that they were able to explain some of the variability present in the data. In general, CNN-COD, $\hat{a}$, was more easily predicted by the metamodels than ETC, $\hat{t}$. This trend is evident from the fact that all four metamodel combinations predicting $\hat{a}$ were above zero, whereas only two out of four of those predicting $\hat{t}$ were below zero, and therefore worse than using a mean model. Of the three build metrics, metamodels that were fitted to predict support material mass CNN performance explained the most variability for both non-aggregated CNN performance metrics. The support material metamodel combination predicting CNN-COD had the highest individual average COD, at 0.27. Part mass was the second most effective, non-aggregate build metric, with an average COD of 0.12. Build time had the least effective metamodels, averaging a COD of 0.02.

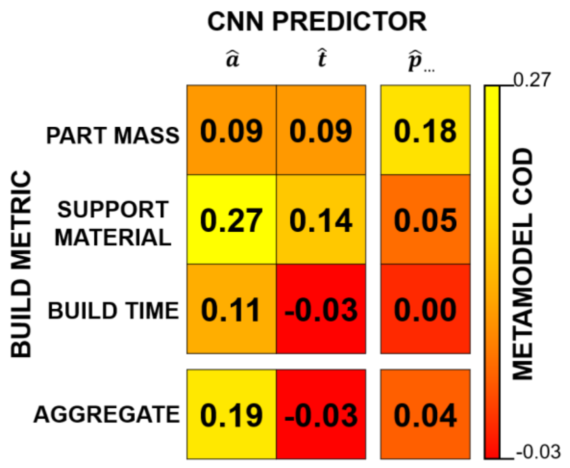

Figure 6 Mean COD values for metamodels predicting each of the independent variables and each build metric.

The most effective aggregate metamodel was the aggregate CNN-COD performance metric, $\hat{p}_a$. It had a COD of 0.19. The $\hat{p}_a$ metamodel was selected for further study and refinement due to its relatively high COD and the high number of significant correlations (see Table 1). This metamodel was also particularly interesting because it provided predictions that were applicable to all three build metrics, making it a more generalized metamodel than some of the others.

4.2. Metamodel Refinement

To further analyze the $\hat{p}_a$ metamodel, the proportion of variability explained by each independent variable was estimated using an ANOVA sequential sum of squares (see Table 2).

Table 2 Explained variance breakdown of the metamodel predicting $\hat{p}_a$ CNN performance metric.

Independent Variable	Proportion of Overall Variance	Proportion of Explained Variance
s	18.6%	84.9%
$C_{PR,m}$	0.6%	2.7%
$C_{PR,r}$	0.1%	0.5%
$C_{AR,m}$	0.6%	2.7%
$C_{AR,r}$	0.0%	0.0%
D_m	0.7%	3.2%
D_r	1.1%	5.0%

As shown in Table 2, size was clearly the largest portion, accounting for 84.9% of the variance explained by the model. This provides substantial predictive power. However, there is a set of additional variables that account for an additional 13.6% when combined. These included median part volume ratio complexity ($C_{PR,m}$), median part area ratio complexity ($C_{AR,m}$), median squared Euclidean distance diversity (D_m), and interquartile range of squared Euclidean distance diversity (D_r). The interquartile ranges of part volume ratio ($C_{PR,r}$) and part area ratio ($C_{AR,r}$) accounted for only 0.5% and 0.0% of the explained variance, respectively.

Since our results indicated that some independent variables were likely more influential than others, we decided to create a refined metamodel that relied only on the most effective predictors. Pursuing a parsimonious predictive model is important to avoid overfitting and encourage comprehension and utility. The proportions of explained variance from each independent variable, alongside those from the independent variable linearity significance test results shown in Table 1 provide statistical bases for a trimming approach to refine the metamodel. Furthermore, the linearity results suggest that certain independent variables are less likely to be well-modeled by a linear metamodel. For the case of the $\hat{p}_a$ metamodel, the insignificantly correlated variables were interquartile range of part volume ratio complexity ($C_{PR,r}$), median part area ratio complexity ($C_{AR,m}$), and interquartile range of part area ratio complexity ($C_{AR,r}$). Heuristically, one might use this information to suggest trimming those independent variables from the metamodel. However, the empirical evidence of variance explained by each variable shown in Table 2, suggests including $C_{AR,m}$ in the refined metamodel may be more effective. An alternative approach could be to remove all variables other than size, since size was clearly the dominant predictor of aggregate CNN-COD in the $\hat{p}_a$ metamodel. We call these two respective approaches the "size-only trimming approach" and "variance trimming approach".

We tested both of these metamodel trimming approaches. This step involved refitting the metamodels with trimmed subsets of the original independent variable vector, evaluating the metamodel performances, and comparing the results. The results are summarized in Tables 3 and 4.

The best-performing refined metamodel was that trimmed with the variance approach, with mean COD equal to 0.22. This COD value was slightly greater than the mean COD of its predecessor, untrimmed metamodel (which had a COD of 0.19). Conversely, the size-only trimming approach resulted in a lower COD of 0.18, indicating that there is value in considering additional metrics.

Based on these results, we selected the refined metamodel based on the variance trimming approach as our final, best metamodel. This metamodel was shown to be able to predict trained CNN performance for multiple build metric applications, as evidenced by its nonzero COD value when predicting the CNN-COD aggregate response. This trait makes it relatively general and more widely applicable than the non-aggregate metamodels. Additionally, the metamodel's mean COD value of 0.22 was relatively large, the second largest of all metamodels attempted. We refit the metamodel using all available data to obtain coefficients and a final explained variance breakdown of a single, fitted OLS metamodel (see Table 4).

Table 3 Coefficients and explained variance of each independent variable in the size only linear metamodel.

Independent Variable	Metamodel Coefficients	Proportion of Overall Variance	Proportion of Explained Variance
β	0.8932	N/A	N/A
s	0.1736	18.6%	100.0%

Table 4 Coefficients and explained variance of each independent variable in the refined linear metamodel.

Independent Variable	Metamodel Coefficients	Proportion of Overall Variance	Proportion of Explained Variance
β	0.8775	N/A	N/A
s	0.1685	18.6%	85.8%
$C_{PR,m}$	0.0844	0.6%	2.9%
$C_{AR,m}$	-0.0551	0.6%	2.6%
D_m	-0.0573	0.7%	3.3%
D_r	0.0630	1.2%	5.4%

5. DISCUSSION

Our results suggest several interesting trends regarding design repository effectiveness when training ML constructs for AM build metrics that may be used to inform development of new heuristics related to design repository assessment. First, we found that the predictive power of a metamodel assessing a design repository can vary substantially depending on the CNN performance metric it attempts to predict. We found that CNN-COD was more accurately predicted than ETC. Additionally, of the aggregate CNN performance metrics, only metamodels predicting $\hat{p}_a$ and $\hat{p}_{PM}$ were of relatively high COD, namely, 0.19 and 0.18. This result suggests that aggregating CNN performance metrics, although potentially useful, must be done with care and should be supported by empirical evidence of effectiveness on a case-by-case basis. In this study, it appears that aggregating too many measures can result in undesirable predictive capabilities, as is the case with the overall aggregate, $\hat{p}$, which only achieved an average COD of 0.04. Additionally, these results differed depending on which build metric was being predicted by the CNNs. This observation is crucial to the expansion of this work to other ML applications within AM. In general, we found that more challenging CNN outputs, such as support material mass, were easier for the metamodel to predict. Understanding the specific causality of this observation is reserved for future work, and may be achieved with a more complex experimental method examining the interactions between these factors.

Our data also shows useful trends related to the effectiveness of individual design repository metadata metrics when used as metamodel independent variables. These results suggest that choosing which metadata metrics will correlate with CNN performance may be best done empirically. We chose the size, complexity, and diversity metadata measures used in this study based on intuition gathered from the experience of training thousands of 3D CNNs in prior work. Ultimately, only some of our intuitively chosen metadata metrics were empirically found to predict CNN performance accurately based on statistical analyses. Some, such as size (s), median part volume ratio complexity ($C_{PR,m}$), and median squared Euclidean distance (D_m) were significantly linear and contributed nonzero explained variance in multiple metamodels. Others, specifically interquartile range of part volume ratio complexity ($C_{PR,r}$) and interquartile range of part area ratio complexity ($C_{AR,r}$) were not significantly linear in any attempted metamodels. These results further illustrate the motivation for this type of study, in which quantitative metamodels are used to predict ML performance. Intuitive experience alone is not likely sufficient to make accurate projections about whether design repositories are effective for use in deep ML training. Further study of the utility of summative design repository attributes may reveal that these complexity metrics are more useful for different predictive models. Similarly, entirely different complexity metrics may be found that improve predictive performance.

6. CLOSING REMARKS

In this study we introduced and demonstrated a novel metamodeling approach to predict CNN performance of AM build metric estimation given a design repository. We created an artificial design repository and split it into sub-repositories with varying size, complexity, and diversity of constituent designs. The sub-repository attributes were then used as independent variables to predict CNN performance using a linear metamodel. We then produced a refined metamodel using only 5 of the original 7 independent variables that was selected as the most capable.

Our first research question investigated the extent to which design repository attributes influence CNN performance. CNN-COD predictors were more frequently significant than ETC. ETC may have more subtle or nonlinear relationships with the independent variables. Also, part mass and build time CNNs were able to explain more variability than support material mass CNNs. However, support material CNNs converged faster. This result indicates that the suitability of particular AM response variables for ML prediction may be dependent on the most important ML performance metric for specific problems. We found that size, median part volume ratio, and squared Euclidean distance interquartile range were positively correlated with CNN-COD. Therefore, design repositories that are larger, contain more complex designs, and are characterized by a higher spatial diversity spread are most effective at training CNNs.

Regarding our second research question, which aimed to determine the extent to which the design repository metadata variables predicted CNN performance, we found that repository size was the dominant predictor. Squared Euclidean distance interquartile range was the second most explanatory independent variable. Although size certainly dominated, the metamodel refinement process showed that including other variables, such as median complexity and diversity values, was worthwhile in making incremental metamodel improvements. Furthermore, metamodels tended to be more accurate at predicting CNN performances if those performances were relatively high. This trend suggests that modeling nonlinear relationships may increase accuracy for low CNN performance cases.

Although the refined metamodel and the metamodel development method serve their purpose in responding to our specific research questions, future work could overcome the limitations of this study and seek to provide a stronger causative pathway for the present results. Investigating real-world design repositories such as design challenge datasets [6] or FabWave [43], a dataset specifically curated for digital design, instead of an artificial design repository may provide additional insight regarding the applicability of our metamodel to such data. These new datasets could also be analyzed for different and more numerous summative metadata attributes, such as metrics that describe the complexity of individual parametric features. Additionally, the current work could be expanded to study different build metrics, AM processes, or AM fabrication performance criteria, such as dimensional distortion due to thermal effects in small features, bridged features, or other challenging geometric attributes. Finally, the design of the metamodel itself could be expanded to explore nonlinear relationships between design repository metadata and ML performance metrics and the choice of particular metadata attributes studied as independent variables could be further investigated.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. CMMI-1825535. The authors acknowledge the generous support of the NVIDIA Corporation through the donation of the Quadro P6000 GPU used in this work. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] Bonnard, R., Hascoët, J. Y., Mognol, P., Zancul, E., and Alvares, A. J., 2019, "Hierarchical Object-Oriented Model (HOOM) for Additive Manufacturing Digital Thread," J. Manuf. Syst., **50**(May 2017), pp. 36–52.

[2] Regli, W. C., and Gaines, D. M., 1997, "A Repository for Design, Process Planning and Assembly," Comput. Des., **29**(12), pp. 895–905.

[3] Chan, S. L., Lu, Y., and Wang, Y., 2018, "Data-Driven Cost Estimation for Additive Manufacturing in

Cybermanufacturing," J. Manuf. Syst., **46**, pp. 115–126.

[4] Tapia, G., Elwany, A. H., and Sang, H., 2016, "Prediction of Porosity in Metal-Based Additive Manufacturing Using Spatial Gaussian Process Models," Addit. Manuf., **12**, pp. 282–290.

[5] Williams, G., Meisel, N. A., Simpson, T. W., and McComb, C., 2019, "Design Repository Effectiveness for 3D Convolutional Neural Networks: Application to Additive Manufacturing," J. Mech. Des. Trans. ASME, **141**(11), pp. 1–12.

[6] McComb, C., Murphey, C., Meisel, N., and Simpson, T. W., 2018, "Predicting Part Mass, Required Support Material, and Build Time via Autoencoded Voxel Patterns," 29th Annu. Int. Solid Free. Fabr. Symp., pp. 1–15.

[7] Arul Prakash, S. K., Mahan, T., Williams, G., McComb, C., Menold, J., and Tucker, C. S., 2020, "Detection of System Compromise in Additive Manufacturing Using Video Motion Magnification," J. Mech. Des., **142**(3), pp. 1–11.

[8] Bloesch-Paidosh, A., and Shea, K., 2018, "Design Heuristics for Additive Manufacturing Validated Through a User Study," J. Mech. Des., (c), pp. 1–40.

[9] Mellor, S., Hao, L., and Zhang, D., 2014, "Additive Manufacturing: A Framework for Implementation," Int. J. Prod. Econ., **149**, pp. 194–201.

[10] Kumke, M., Watschke, H., and Vietor, T., 2016, "A New Methodological Framework for Design for Additive Manufacturing," Virtual Phys. Prototyp., **11**(1), pp. 3–19.

[11] Lynn, R., Saldana, C., Kurfess, T., Reddy, N., Simpson, T., Jablokow, K., Tucker, T., Tedia, S., and Williams, C., 2016, "Toward Rapid Manufacturability Analysis Tools for Engineering Design Education," Procedia Manuf., **5**, pp. 1183–1196.

[12] Prabhu, R., Miller, S. R., Simpson, T. W., and Meisel, N. A., 2018, "Teaching Design Freedom: Exploring the Effects of Design for Additive Manufacturing Education on the Cognitive Components of Students' Creativity," Proc. ASME Des. Eng. Tech. Conf., **3**, pp. 1–13.

[13] Sinha, S., Chen, H. E., Meisel, N. A., and Miller, S. R., 2017, "Does Designing for Additive Manufacturing Help Us Be More Creative? An Exploration in Engineering Design Education," Proc. ASME Des. Eng. Tech. Conf., **3**(August).

[14] Yang, T., Chen, Y., and Sze, V., 2017, "Designing Energy-Efficient Convolutional Neural Networks Using Energy-Aware Pruning," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 6071–6079.

[15] Razvi, S. S., Feng, S., Narayanan, A., Lee, Y.-T. T., and Witherell, P., 2019, "A Review of Machine Learning Applications in Additive Manufacturing," *Volume 1: 39th Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, pp. 1–10.

[16] Schmidhuber, J., 2015, "Deep Learning in Neural Networks: An Overview," Neural Networks, **61**, pp. 85–117.

[17] Meisel, N., and Williams, C., 2015, "An Investigation of Key Design for Additive Manufacturing Constraints in Multimaterial Three-Dimensional Printing," J. Mech. Des. Trans. ASME, **137**(11), pp. 1–9.

[18] Munguía, J., Ciurana, J., and Riba, C., 2009, "Neural-Network-Based Model for Build-Time Estimation in Selective Laser Sintering," Proc. Inst. Mech. Eng. Part B J. Eng. Manuf., **223**(8), pp. 995–1003.

[19] Roy, M., and Wodo, O., 2020, "Data-Driven Modeling of Thermal History in Additive Manufacturing," Addit. Manuf., **32**(December 2019), p. 101017.

[20] Gaikwad, A., Yavari, R., Montazeri, M., Cole, K., Bian, L., and Rao, P., 2019, "Toward the Digital Twin of Additive Manufacturing – Integrating Thermal Simulations, Sensing, and Analytics to Detect Process Faults," IISE Trans., **0**(0), pp. 1–22.

[21] Mahmoudi, M., Ezzat, A. A., and Elwany, A., 2019, "Layerwise Anomaly Detection in Laser Powder-Bed Fusion Metal Additive Manufacturing," J. Manuf. Sci. Eng., **141**(3), p. 031002.

[22] Liu, C., Law, A. C. C., Roberson, D., and Kong, Z. (James), 2019, "Image Analysis-Based Closed Loop Quality Control for Additive Manufacturing with Fused Filament Fabrication," J. Manuf. Syst., **51**(October 2018), pp. 75–86.

[23] Gobert, C., Reutzel, E. W., Petrich, J., Nassar, A. R., and Phoha, S., 2018, "Application of Supervised Machine Learning for Defect Detection during Metallic Powder Bed Fusion Additive Manufacturing Using High Resolution Imaging.," Addit. Manuf., **21**(May 2017), pp. 517–528.

[24] Özel, T., Altay, A., Kaftanoğlu, B., Leach, R., Senin, N., and Donmez, A., 2020, "Focus Variation Measurement and Prediction of Surface Texture Parameters Using Machine Learning in Laser Powder Bed Fusion," J. Manuf. Sci. Eng., **142**(1), pp. 1–12.

[25] Caggiano, A., Zhang, J., Alfieri, V., Caiazzo, F., Gao, R., and Teti, R., 2019, "Machine Learning-Based Image Processing for on-Line Defect Recognition in Additive Manufacturing," CIRP Ann., pp. 3–6.

[26] Cui, W., Zhang, Y., Zhang, X., Li, L., and Liou, F., 2020, "Metal Additive Manufacturing Parts Inspection Using Convolutional Neural Network," Appl. Sci., **10**(2), p. 545.

[27] Khadilkar, A., Wang, J., and Rai, R., 2019, "Deep Learning–Based Stress Prediction for Bottom-up SLA 3D Printing Process," Int. J. Adv. Manuf. Technol., **102**(5–8), pp. 2555–2569.

[28] Rawat, W., and Wang, Z., 2017, "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review," Neural Comput., **29**(9), pp. 2352–2449.

[29] Ioannidou, A., Chatzilari, E., Nikolopoulos, S., and

Kompatsiaris, I., 2017, “Deep Learning Advances in Computer Vision with 3D Data,” ACM Comput. Surv., **50**(2), pp. 1–38.

[30] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F., 2015, “ShapeNet: An Information-Rich 3D Model Repository.”

[31] Maturana, D., and Scherer, S., 2015, “VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition,” *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 922–928.

[32] Szykman, S., Sriram, R. D., Khanh, T., Jean-Francois, H., Cuilhem, A., and Sylvain, A., 1999, “The NIST Design Repository Project: Project Overview and Implementational Design,” Adv. Soft Comput., pp. 5–19.

[33] Vayre, B., Vignat, F., and Villeneuve, F., 2012, “Designing for Additive Manufacturing,” Procedia CIRP, **3**(1), pp. 632–637.

[34] Bohm, M. R., Stone, R. B., Simpson, T. W., and Steva, E. D., 2008, “Introduction of a Data Schema to Support a Design Repository,” CAD Comput. Aided Des., **40**(7), pp. 801–811.

[35] Song, H., and Fu, K., 2019, “Design-by-Analogy: Exploring for Analogical Inspiration with Behavior, Material, and Component-Based Structural Representation of Patent Databases,” J. Comput. Inf. Sci. Eng., **19**(2).

[36] Tensa, M., Edmonds, K., Ferrero, V., Mikes, A., Soria Zurita, N., Stone, R., and DuPont, B., 2019, “Toward Automated Functional Modeling: An Association Rules Approach for Mining the Relationship between Product Components and Function,” Proc. Des. Soc. Int. Conf. Eng. Des., **1**(1), pp. 1713–1722.

[37] Bohm, M. R., Vucovich, J. P., and Stone, R. B., 2008, “Using a Design Repository to Drive Concept Generation,” J. Comput. Inf. Sci. Eng., **8**(1), pp. 0145021–0145028.

[38] Bohm, M. R., Stone, R. B., and Szykman, S., 2005, “Enhancing Virtual Product Representations for Advanced Design Repository Systems,” J. Comput. Inf. Sci. Eng., **5**(4), p. 360.

[39] Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li, and Li Fei-Fei, 2010, “ImageNet: A Large-Scale Hierarchical Image Database,” *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, pp. 248–255.

[40] Egmont-Petersen, M., de Ridder, D., and Handels, H., 2002, “Image Processing with Neural Networks—a Review,” Pattern Recognit., **35**(10), pp. 2279–2301.

[41] Brock, A., Lim, T., Ritchie, J. M., and Weston, N., 2016, “Generative and Discriminative Voxel Modeling with Convolutional Neural Networks.”

[42] Dering, M. L., and Tucker, C. S., 2017, “A Convolutional Neural Network Model for Predicting a Product’s Function, Given Its Form,” J. Mech. Des., **139**(11), p. 111408.

[43] Bharadwaj, A., Xu, Y., Angrish, A., Chen, Y., and Starly, B., 2019, “Development of a Pilot Manufacturing Cyberinfrastructure with an Information Rich Mechanical Cad 3D Model Repository,” ASME 2019 14th Int. Manuf. Sci. Eng. Conf. MSEC 2019, **1**, pp. 1–8.

[44] Sibi, P., Allwyn Jones, S., and Siddarth, P., 2013, “Analysis of Different Activation Functions Using Back Propagation Neural Networks,” J. Theor. Appl. Inf. Technol., **47**(3), pp. 1344–1348.

[45] Welinder, P., and Perona, P., 2010, “Online Crowdsourcing: Rating Annotators and Obtaining Cost-Effective Labels,” 2010 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. - Work. CVPRW 2010, pp. 25–32.

[46] Baldridge, J., and Osborne, M., 2004, “Active Learning and the Total Cost of Annotation,” Proc. Empir. Methods Nat. Lang. Process., pp. 9–16.

[47] Rao, R. B., Fung, G., and Rosales, R., 2008, “On the Dangers of Cross-Validation. An Experimental Evaluation,” Soc. Ind. Appl. Math. - 8th SIAM Int. Conf. Data Min. 2008, Proc. Appl. Math. 130, **2**, pp. 588–596.

[48] Szykman, S., Sriram, R. D., Bochenek, C., Racz, J. W., and Senfaute, J., 2000, “Design Repositories: Engineering Design’s New Knowledge Base,” IEEE Intell. Syst. Their Appl., **15**(3), pp. 48–55.

[49] 2019, “Cloud Powered 3D CAD/CAM Software for Product Design | Fusion 360” [Online]. Available: https://www.autodesk.com/products/fusion-360.

[50] Cho, J., Lee, K., Shin, E., Choy, G., and Do, S., 2015, “How Much Data Is Needed to Train a Medical Image Deep Learning System to Achieve Necessary High Accuracy?”

[51] Conner, B. P., Manogharan, G. P., Martof, A. N., Rodomsky, L. M., Rodomsky, C. M., Jordan, D. C., and Limperos, J. W., 2014, “Making Sense of 3-D Printing: Creating a Map of Additive Manufacturing Products and Services,” Addit. Manuf., **1**, pp. 64–76.

[52] Kechagias, J., and Chryssolouris, G., 1997, “Estimation of Build Times in Rapid Prototyping Processes,” *Proceedings of the 6th European Conference on Rapid Prototyping and Manufacturing. University of Nottingham, UK.*

[53] Abadi, M., Paul, B., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X., 2016, “TensorFlow: A System for Large-Scale Machine Learning,” 12th USENIX Symp. Oper. Syst. Des. Implement. (OSDI ’16), pp. 265–283.

[54] Chollet, F., and others, 2015, “Keras.”

[55] Kingma, D. P., and Ba, J., 2015, “Adam: A Method for Stochastic Optimization,” 3rd Int. Conf. Learn. Represent., **1631**, pp. 58–62.

[56] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., 2011, "Scikit-Learn: Machine Learning in Python," J. Mach. Learn. Res., **12**, pp. 2825–2830.

[57] "StatsModels - Statistics in Python" [Online]. Available: https://www.statsmodels.org/0.9.0/index.html.