

## RESEARCH ARTICLE

# Neural network-based build time estimation for additive manufacturing: a performance comparison

Yosep Oh<sup>1</sup>, Michael Sharp<sup>2</sup>, Timothy Sprock<sup>2</sup> <sup>2,\*</sup> and Soonjo Kwon<sup>3</sup> <sup>3,\*</sup>

<sup>1</sup>Department of Industrial and Management Engineering, Kyonggi University, Suwon-si, Gyeonggi-do 16227, Republic of Korea; <sup>2</sup>Engineering Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA and <sup>3</sup>Department of Mechanical System Engineering, Kumoh National Institute of Technology, Gumi-si, Gyeongsangbuk-do 39177, Republic of Korea

\*Current Affiliation: Applied Research Laboratory for Intelligence and Security (ARLIS), University of Maryland, College Park, MD, USA.

\*Corresponding author. E-mail: [soonjo.kwon@kumoh.ac.kr](mailto:soonjo.kwon@kumoh.ac.kr)  <http://orcid.org/0000-0003-1578-9262>

## Abstract

Additive manufacturing (AM) has brought positive opportunities with phenomenal changes to traditional manufacturing. Consistent efforts and novel studies into AM use have resolved critical issues in manufacturing and broadened technical boundaries. Build time estimation is one of the critical issues in AM that still needs attention. Accurate build time estimation is key for feasibility studies, preliminary design, and process/production planning. Recent studies have provided the possibility of *neural network (NN)*-based build time estimation. In particular, *traditional artificial NN (ANN)*- and *convolutional NN (CNN)*-based methods have been demonstrated. However, very little has been done on the performance comparison for build time estimation among the different types of NNs. This study is aimed at filling this gap by designing various NNs for build time estimation and comparing them. Two types of features are prepared as inputs for the NNs by processing three-dimensional (3D) models: (1) *representative features (RFs)* including dimensions, part volume, and support volume; and (2) the set of voxels generated from designating the cells occupied by the workpiece in a mesh grid. With the combination of NN types and input feature types, we design three NNs: (1) ANN with RFs; (2) ANN with voxels; and (3) CNN with voxels. To obtain large enough label data for reliable training, we consider simulation build time from commercial slicing applications rather than actual build time. The simulation build time is calculated based on a *material extrusion* process. To address various cases for input models, two design factors (scale and rotation) are considered by controlling the size and build orientation of 3D models. In computational experiments, we reveal that the CNN-based estimation is often more accurate than others. Furthermore, the design factors affect the performance of build time estimation. In particular, the CNN-based estimation is strongly influenced by changing the size of 3D models.

**Keywords:** additive manufacturing; artificial neural network; 3D convolutional neural network; build time estimation; 3D printing

## 1 Introduction

Additive manufacturing (AM), commonly referred to as *three-dimensional (3D) printing*, has spurred manufacturing industry (Khorrarn Niaki & Nonino, 2017; Ceruti et al., 2019; Stolt &

Elgh, 2020). AM enables to produce customized and personalized products, allowing for expanded focus toward mass customization in the manufacturing community (Oh, 2019; Oh et al., 2021). While the flexibility of AM enables the production of a

Received: 25 November 2020; Revised: 19 May 2021; Accepted: 24 May 2021

© The Author(s) 2021. Published by Oxford University Press on behalf of the Society for Computational Design and Engineering. This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited. For commercial re-use, please contact [journals.permissions@oup.com](mailto:journals.permissions@oup.com)

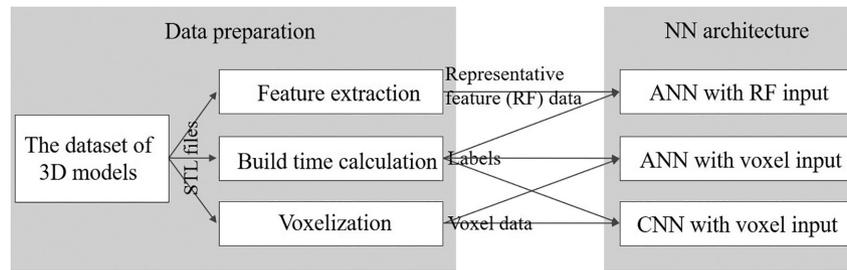


Figure 1: NN models depending on inputs (RF, voxel and label data).

variety of products, it also creates challenges in managing them in terms of production planning and control. To efficiently handle a number of heterogeneous parts for AM, the accurate prediction of cost, time, and quality for products before production is critical (Zhang & Bernard, 2013; Seo et al., 2021). In particular, manufacturers need to accurately estimate build time to make capacity and scheduling decisions in a timely manner. The often long build times for each build process make the issue more challenging by exaggerating errors between estimated time and actual time.

Meanwhile, *deep learning* (or *deep neural network*, DNN; Antescu et al., 2019; Guo et al., 2019) has paved the way for build time estimation. Once a NN model is well trained, it provides relatively high accuracy (or acceptable accuracy) of prediction time (Zhang et al., 2015). This can be beneficial when quickly and roughly estimating build time with limited information. In particular, it can be effective in the early design stage where specific process parameters are not available and the geometric information is the only input for the build time estimation. *Artificial neural network* (ANN)-based estimation showed relatively high performance with a few representative geometric features including the size and volume of parts (Munguía et al., 2009). Recently, *3D convolutional neural network* (CNN) was introduced to investigate part mass, support material mass, and build time (Williams et al., 2019). Extracting the spatial features from the 2D or 3D data is one of the advantages of CNN (Vania et al., 2019).

Although a variety of NNs have been designed and applied to build time estimation, there have been very few attempts at comparing the performances among the NNs. Moreover, it is hard to conclude which method is superior by directly comparing the experiment results (e.g. error rate and accuracy) of the previous studies. This is mainly because the studies were conducted based on different conditions depending on AM processes, input geometries, and machine parameters. As a step toward addressing this research gap, we design different types of NNs for build time estimation and compare them under the same experimental conditions. The main contributions of this study are summarized as follows:

1. In this paper, we compare performances of NN-based build time methods, which has been less addressed in the literature.
2. With experimental results for a variety of cases, we show how much the CNN-based build time estimation is more accurate than the ANN-based estimation.
3. Based on a statistical result, we show that the size of 3D models is critical in the CNN-based estimation while the amount of support structure is less significant.

Figure 1 presents the overall procedure of this study. Since build time estimation is a regression problem and categorized into supervised learning, features and labels should be paired for

training and testing. For data preparation, features and labels are obtained from the public dataset of 3D models. We extract two types of features from 3D models: (1) *representative features* (RFs) including dimensions, part volume, and support volume and (2) the set of voxels that are the cells occupied by the workpiece or the empty space in a build space. For label data, we consider simulation build time calculated by slicing software. With the combination of NN types and input feature types, three NNs are designed: (1) ANN with RFs; (2) ANN with voxels; and (3) CNN with voxels. Then, these three NNs are trained and the prediction results are compared. To specifically investigate the various geometries of 3D models, input models are created by controlling two design factors, scale and rotation. The two design factors change the size and build orientation of 3D models.

In this study, instead of actual build time, simulation build time is considered as label data for three reasons. First, we consider a lot of label data for 11,792 geometries (2,948 models for each case  $\times$  4 cases) to analyse a variety of cases. It will be extremely costly and time consuming to obtain actual build time as label data for 11,792 geometries since a build process running a 3D printer takes a long time from a couple of hours to even days. Second, since *machine learning* (ML) and *deep learning* (DL) require a significant number of data, simulation data are often used as label data in build time estimation (Williams et al., 2019) as well as other research areas [e.g. computational mechanics (Samaniego et al., 2020) and stress analysis (Khadilkar et al., 2019)]. Lastly, since the main objective of our study is to provide the performance comparisons of NN-based methods, whether actual build time is used as label data is not essential.

The rest of the paper is organized as follows: Section 2 provides literature reviews for build time estimation studies and CNN for AM. Section 3 details how we obtain the dataset of features and labels for ML. Section 4 describes how we design the architecture of the three NNs. In Section 5, computational experiments are conducted to compare the three NNs. Section 6 concludes this research and discusses future work.

## 2 Literature Review

This section provides literature reviews. Section 2.1 summarizes previous studies about build time estimation for AM. Section 2.2 briefly introduces CNN and explains how CNNs have been applied to AM studies.

### 2.1 Build time estimation

In AM, there are mainly three approaches to estimate the build time (Zhang et al., 2015): (1) parametric, (2) analytical, and (3) analogical. Figure 2 roughly represents that the accuracy is improved depending on the number of input parameters. In the parametric approach, only a few key parameters (e.g. part

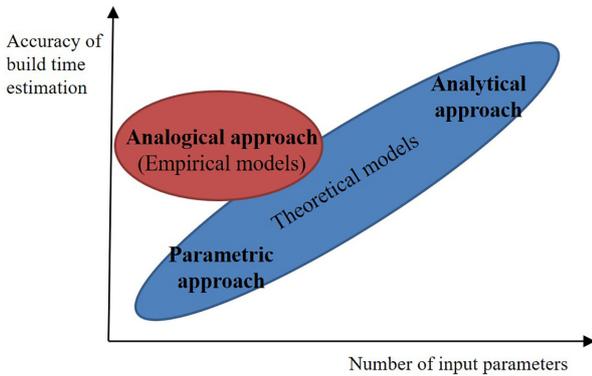


Figure 2: Build time estimation approaches: parametric, analytical, and analogical.

volume, height, and surface roughness) could be considered to estimate the build time (Choi & Samavedam, 2002). Since the mathematical model of the parametric approach is simple and even crude by ignoring many key parameters for the machine setup and processing details, the prediction accuracy is low. On the other hand, if a number of input parameters are considered based on the analytical approach, the estimation model becomes complicated, although the high accuracy is guaranteed. Commercial slicing software for 3D printing, a *simulator*, is based on the analytical approach to accurately estimate build time and cost. For instance, PrusaSlicer (<https://www.prusa3d.com/prusa-slicer/>) considers almost 180–200 input parameters to estimate build time for a *material extrusion (ME)* process (Zhang & Moon, 2021). The input parameters cover not only part information (e.g. geometries, location, and build orientation) but also kinematic factors (e.g. a laser velocity; Chen & Sullivan, 1996), driving factors for tool path (e.g. a contouring area and a hatching length; Giannatsis et al., 2001), and support structure-related factors (e.g. a self-supporting angle; Alexander et al., 1998). In short, there is a trade-off between the parametric and analytical approaches. It results in that people look for a mathematical estimation model that is placed in the grey area of parametric and analytical approaches, considering a “proper” number of parameters and ensuring the “acceptable” accuracy. The acceptable accuracy could be different depending on when the build time estimation is applied. For example, the low accuracy is often acceptable at the early design stage. This is because the main reason applying the build time estimation at the design stage is to recognize which design candidate is better in terms of build time rather than to accurately estimate build time. Moreover, some parameters may not be available at the design stage since machine setup parameters (e.g. laser power, laser speed, etc.) are usually determined at the process planning stage following the design stage.

Meanwhile, researchers have proposed the analogical (or experimental; Medina-Sanchez et al., 2019) approach that is a different way from the parametric and analytical approaches. The analogical approach is estimating build time based on an empirical model created by experiments and input datasets. A regression model (Ruffo et al., 2006), a NN model (Munguía et al., 2009), and an adaptive model based on the Grey theory (Zhang et al., 2015) are classified into the analogical approach. The analogical methods have been suggested to reduce the computation cost and model complexity of the analytical approach (Zhang et al., 2015). As shown in Fig. 2, the analogical approach shows relatively high accuracy (or acceptable accuracy) of build time estimation based on a small number of input parameters. As such,

the analogical methods have been suggested to reduce the computation cost and model complexity of the analytical approach (Zhang et al., 2015). In particular, it can be effective in the design stage rather than the process planning stage. To be specific, the analogical approach could be adopted when build time is required at the preliminary design stage or the early stage of production plan where slicing data from a CAD file or process parameters including machine setup are not available.

A comparison of the parametric, analytical, and analogical approaches has been provided in the literature (Zhang et al., 2015; Kadir et al., 2020). However, there have been very few attempts at comparing between the analogical methods, especially NN models. As one of the analogical methods, an ANN technique has been applied to estimate build time. Munguía et al. proposed ANN-based build time estimation for *Selective Laser Sintering* and error rates were observed from 2 to 15% (Munguía et al., 2009). Angelo et al. suggested another ANN-based build time estimation and error rates were from 3 to 16% for a *fused deposition modeling (FDM)* machine and from 6 to 20% for a *3D printing (3DP)* machine (Di Angelo & Di Stefano, 2011). However, given the accuracy numbers that the previous studies provide, it is hard to tell which method is better than others. This is because the accuracy or error rate of the previous studies depends on AM process types, the geometry of models, and process parameters. To provide the fair comparisons of multiple NNs, we prepare the same dataset of 3D models and focus on an ME process. Furthermore, when actual build time is considered as label data, it is challenging to obtain a large enough number of samples since a build process running a 3D printer takes a long time from a couple of hours to even days. To obtain enough data for training, we adopt commercial slicing applications to calculate build time.

## 2.2 CNN for AM

DL is a part of ML based on a NN model with multiple hidden layers (Francis & Bian, 2019). In the literature, ANN and CNN for DL are often named as DNN (Kwon et al., 2020; Samaniego et al., 2020) and *deep convolutional neural network (DCNN)* (Caggiano et al., 2019), respectively. In this paper, although more than two hidden layers are considered, we use simple terms, ANN and CNN, without “deep” in the remaining sections.

In ML, CNN is categorized into a class of DL (Valueva et al., 2020). CNN has been widely applied to image processing domains (e.g. image and video recognition, image classification, and medical image analysis; Williams et al., 2019). Extracting the spatial features (e.g. edges) from the data using a kernel in a convolution layer is one of the main advantages of CNNs (Spruegel et al., 2021). This characteristic of CNN provides translation equivariance, which means that translating input data does not affect the recognition of the key geometric features (Worrall et al., 2017). Therefore, CNN is usually adopted when facilitating the identification and classification of 2D images or 3D models containing spatial properties.

Recently, CNN has been applied to the AM domain in various ways. Most of the CNN studies focused on 2D CNN for laser powder bed fusion by collecting 2D images in process to identify defects (Caggiano et al., 2019; Zhang et al., 2019a), flaws (Imani et al., 2019), anomaly (Scime & Beuth, 2018), and porosity (Zhang et al., 2019b) and to predict laser power (Kwon et al., 2020). Additionally, 2D CNNs were applied to FDM for defect classification (Wang et al., 2020) and *Stereolithography* for stress prediction (Khadilkar et al., 2019). While a 2D CNN usually uses 2D images (e.g. cross-section and melt-pool images), a 3D CNN uses 3D models.



Figure 3: 3D models from Bauman's dataset.

Eranpurwala *et al.* applied 3D CNN to determine the build orientation of parts (Eranpurwala *et al.*, 2020). Williams *et al.* adopted 3D CNN-based build time estimation (Williams *et al.*, 2019). While their work is relevant to the 3D CNN of our study, there are some critical differences. First, to obtain label data (a.k.a. target estimations) for build time, they used a simple equation with some key factors by assuming a constant material deposition rate by volume. On the other hand, we generate our labeled data from slicing applications to obtain estimations as close as possible to the real build time. Additionally, Williams *et al.* arbitrarily generated dataset repositories from 18 design templates by putting different parameters for translation and rotation. While they adopted relatively simple geometries for the design templates, we use practical geometries for 3D printing from a validated public dataset (Baumann, 2018). It should be noted that we adopt 3D CNN rather than 2D CNN since 3D models are inputs to the NNs. Therefore, in the remaining sections, we name 3D CNN as CNN for simplicity.

### 3 Dataset Preparation

This section details how our datasets are prepared for ML. The datasets are composed of feature and label data. In a regression problem, features (i.e. independent variables) are used to predict a label (i.e. a dependent variable). Section 3.1 explains how validated 3D models are obtained. Section 3.2 presents how features are extracted from the 3D models. Section 3.3 describes how label data (i.e. build time in this study) are obtained.

#### 3.1 3D models

For 3D models, we adopt the STL files of Baumann's dataset (Baumann, 2018). The dataset includes 30,000 models that are collected from Thingiverse (<https://www.thingiverse.com/>). The dataset covers a variety of geometries as shown in Fig. 3. In the dataset, we only consider the small-sized files that are less than 10 MB for computational efficiency. Then, the geometries are normalized into a unit size that the maximum dimension is 1 mm. To include validated 3D models, they go through a manifold-check process. Moreover, too thin or long geometries are excluded to avoid some extreme cases. After the filtering, normalization, and validation processes, 2,948 models are chosen as the base geometries.

The size and orientation of 2,948 base models are redefined to consider various cases. By controlling the scale and rotation

Table 1: Design control combinations for four design treatments.

|       |      | Rotation |      |
|-------|------|----------|------|
|       |      | Low      | High |
| Scale | Low  | N        | R    |
|       | High | S        | SR   |

of the base models, we generate redefined geometries for two levels as follows:

1. Low scale: randomly scaling from 30 to 90 mm.
2. High scale: randomly scaling from 90 to 150 mm.
3. Low rotation: no rotating (the initial orientation).
4. High rotation: randomly rotating from  $-90$  to  $90$  degrees for the X- and Y-axes.

When rotating for an axis, we only consider the X- and Y-axes since rotating for Z-axis is usually not critical to build time (Gogate & Pande, 2008). We consider scale and rotation for design factors to investigate a variety of models, which is detailed in Section 3.2.1. The combination of two levels, high and low, for two design factors, scale and rotation, generates four design treatments as shown in Table 1. Since each design treatment includes 2,948 models, we consider 11,792 ( $4 \times 2,948$ ) geometries as input models.

#### 3.2 Feature data

##### 3.2.1 Representative geometric features

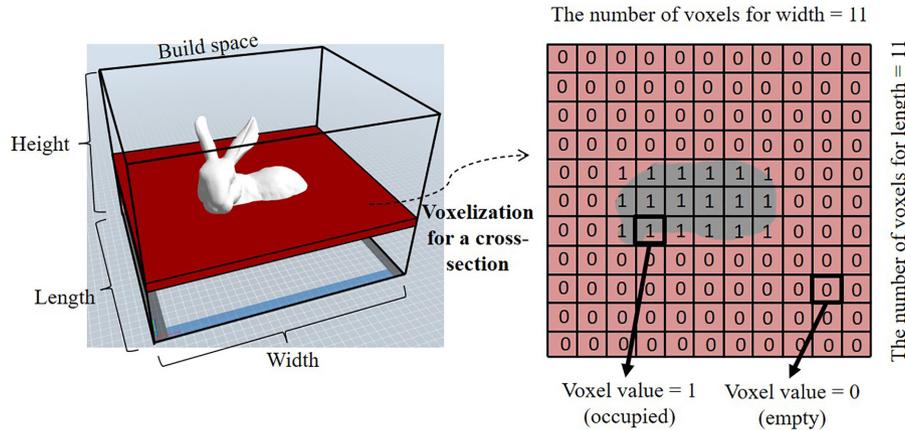
For RFs, we consider the width (dimension X), length (dimension Y), and height (dimension Z) of a bounding box, the volume of a 3D model, and the volume of support structure. These features are used as independent variables to estimate build time. To calculate the volume of support structure, we partially refer to a method of Williams *et al.* (2019). The calculation is represented by

$$V_s = \sum_{i \in M_s} d_i A(i), \quad (1)$$

where  $V_s$  is the calculated support volume,  $M_s$  is a set of meshes with an overhang angle (less than 45 degrees),  $A(i)$  is the area of mesh  $i$ , and  $d_i$  is a distance in the negative Z-direction between the center of mass of Mesh  $i$  and either another surface of the model or the build platform.

**Table 2:** The average of dimensions, part volume, and support amount for each case.

| Design treatments | Dimension X (mm) | Dimension Y (mm) | Dimension Z (mm) | Part volume (mm <sup>3</sup> ) | Support amount (mm <sup>3</sup> ) |
|-------------------|------------------|------------------|------------------|--------------------------------|-----------------------------------|
| N                 | 51.75            | 46.37            | 33.03            | 28,414.84                      | 725.15                            |
| R                 | 55.95            | 47.86            | 56.18            | 28,636.83                      | 2,259.54                          |
| S                 | 104.08           | 93.51            | 65.88            | 192,375.22                     | 4,520.99                          |
| SR                | 111.15           | 94.36            | 111.70           | 193,798.60                     | 14,131.13                         |

**Figure 4:** Voxel data structure.

As discussed in Section 3.1, each case includes 2,948 models. The average of four RFs for each case is represented in Table 2. As shown in the table, dimensions, part volume, and support amount depend on the cases. The average of dimensions and part volume shows that design treatments S and SR include relatively large geometries. In addition, the geometries of design treatments R and SR generate more support structure compared to design treatments N and S. This is because it is likely to have more support structure by rotating parts. Usually, parts with an initial orientation have less support structure since they are stably placed. Since RFs have different units, they are normalized based on the standard score with the mean and standard deviation of the data population.

RFs are used as input features for an ANN model, which is named as ANN with RFs and detailed in Section 4.2. The idea of build time estimation based on ANN with RFs is originated from the work of Munguía *et al.* (2009). To differentiate from the previous work, we consider the amount of support structure as an RF that represents a characteristic of AM.

### 3.2.2 Voxel data

The set of voxels is another type of feature data. A voxel set is obtained from a voxelization process<sup>1</sup>. In computer graphics, the voxelization process is to convert a 3D model into a set of cubes (i.e. voxels). In this paper, however, an object for voxelization is not just a 3D model but a build space including the model. If a certain space of a voxel is occupied by some part of a 3D model, the binary value for the voxel becomes 1. Otherwise, it becomes 0, which means that the space of the voxel is empty. For the voxelization process, we assume that every model is a solid model. Therefore, the voxel value of inner part of a 3D model is 1 since the inside of a solid model is filled. Along the X-, Y-, and Z-axes, every voxel within a build space is checked whether

the binary value is 1 or 0. Finally, the voxelization process generates the data of 1D array consisting of binary values for voxels. Figure 4 represents the voxel data structure for a certain cross-section. In the top view, the build space is represented by a grid structure (11 × 11 voxels). As shown in the figure, the voxel value of occupied spaces for the rabbit model is 1.

In the voxelization approach, a voxel size is critical in determining the degree of resolution. The voxel size refers to the dimension of a voxel. Figure 5 presents the resolution of voxelization depending on voxel sizes. If voxelization is conducted with a small voxel size, high resolution can be achieved. Otherwise, it generates low resolution that the surface of a 3D model becomes coarse.

There is a trade-off between resolution and computation efficiency (Zhang *et al.*, 2018). If high-resolution data are used for ML, it would take a lot of processing time and require a large storage capacity while the build time estimation is accurate. This is shown in Fig. 6 that illustrates the performance of an estimation model and the difficulty of data handling depending on voxel sizes. To be specific, Fig. 6a represents *mean absolute percent error (MAPE)* and  $R^2$  that are indicators for the performance of a regression model detailed in Sections 5.1 and 5.4.1. Figure 6b presents processing time for the voxelization process and the file size of voxel data. As shown in the figures, if the voxel size gets smaller, process time and data file size become increased while the prediction for build time becomes accurate. While the performance gets better linearly (see Fig. 6a), the difficulty is dramatically increased from 6 to 3 mm (see Fig. 6b). We discarded to obtain the data for a voxel size of 2 mm since it would take so much processing time for voxelization and require a lot of data size. To compromise on a certain level, we decide to proceed computation experiments with a voxel size of 3 mm.

In this study, a build space with 250 × 250 × 250 mm is voxelized based on a voxel size with 3 mm. As such, 592,704

<sup>1</sup> The source code is available at <https://github.com/soonjokwon/VOX4AM>

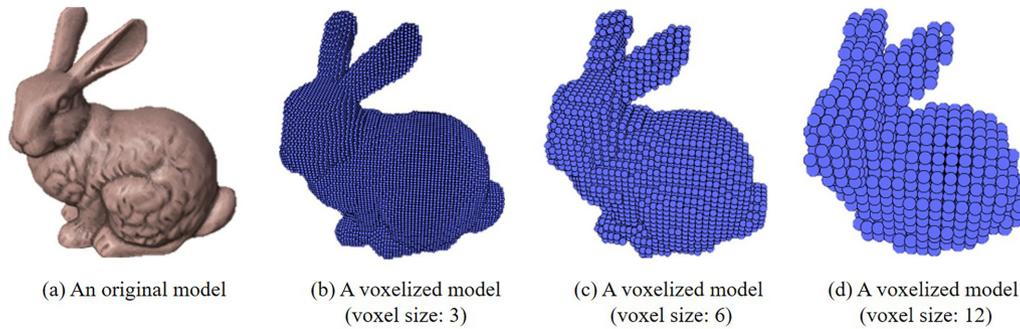


Figure 5: The resolution of voxelization depending on voxel size.

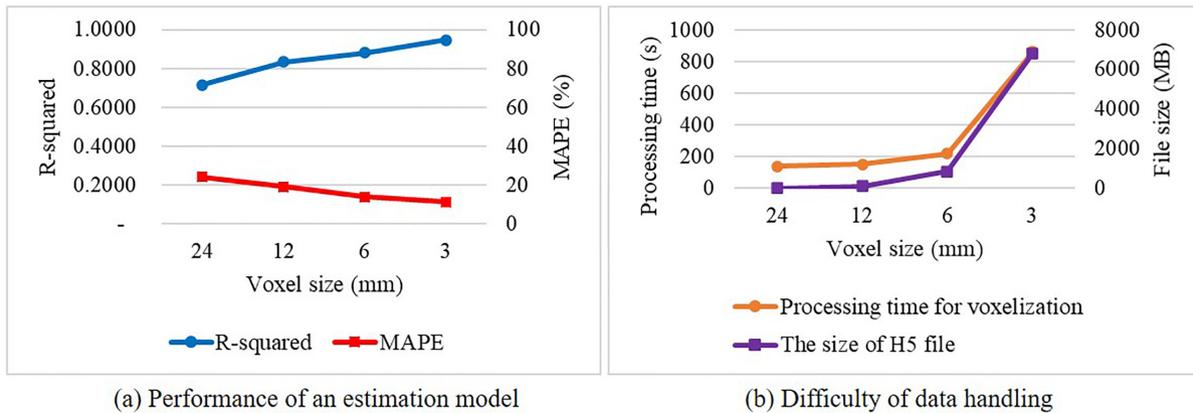


Figure 6: Computation efficiency depending on voxel sizes: (a) performance of an estimation model and (b) difficulty of data handling.

voxels,  $84 \times 84 \times 84$ , are generated for each build space including a 3D model. Since each voxel has a binary value, each build space is converted into a 1D array consisting of 592,704 binary values. After the voxelization process for each case, an H5 file (<https://support.hdfgroup.org/>) is created and its data structure is a 2D array consisting of 2,948 models  $\times$  592,704 values. The size of an H5 file for each case is about 6.5 GB. Since we deal with four cases, four H5 files are needed for an experiment.

### 3.3 Label data

To obtain label data, collecting actual build time for 11,792 models (2,948 models for each case  $\times$  4 cases) is costly and time consuming. As such, instead of actual build time, we consider the simulation build time calculated from slicing applications.

To calculate build time, two slicing applications, PrusaSlicer 2.2.0 (<https://www.prusa3d.com/prusaslicer/>) and UltimakerCura 4.5 (<https://ultimaker.com/software/ultimaker-cura>), are employed. We consider multiple slicing applications to avoid that an experiment result is biased for a certain application. The applications calculate build time by slicing a 3D model layer by layer and extracting tool path. Virtual 3D printers are embedded in the slicing applications. In this study, Ultimaker Original and Original Prusa i3 MK3S are used as virtual 3D printers for an ME process. In addition, except for the following items, process parameters are set as the default values of the virtual printers.

1. Build space:  $250 \times 250 \times 350$  mm
2. Material type: PLA
3. Overhang threshold for support structures: 45 degrees

4. Layer thickness: 0.2 mm
5. Infill rate: 30%

## 4 The Design of NNs

Sections 4.1 and 4.2 detail the architecture of CNN and ANN, respectively. While Section 4.1 describes a CNN structure that voxels are input features, Section 4.2 describes the two types of input features (i.e. RFs and voxels) for ANN. Section 4.3 provides how hyperparameters are determined for the three NNs.

### 4.1 CNN architecture

Figure 7 presents the architecture of CNN for estimating build time. For CNN, a pair of feature data and label data is provided. The feature data are 592,704 voxel values ( $84 \times 84 \times 84$ ) for 2,948 models (see Section 3.2.2). As a regression problem, the output layer has only one output neuron and its activation function is linear since an output variable takes a continuous value. In our CNN model, two pairs of convolutional and max-pooling layers are included. The primary objective of a convolutional layer is to extract specific features from input data (e.g. a 3D model) by sliding a kernel (or a filter) and creating a feature map. A pooling layer reduces the spatial size (or dimensionality) of the feature map. Then, in a fully connected layer, the features are flattened into neurons. The activation function of the neurons in the fully connected layer is *rectified linear unit activation (ReLU)*. Additionally, we include dropout for output values from the fully connected layer to reduce overfitting (Aggarwal, 2018).

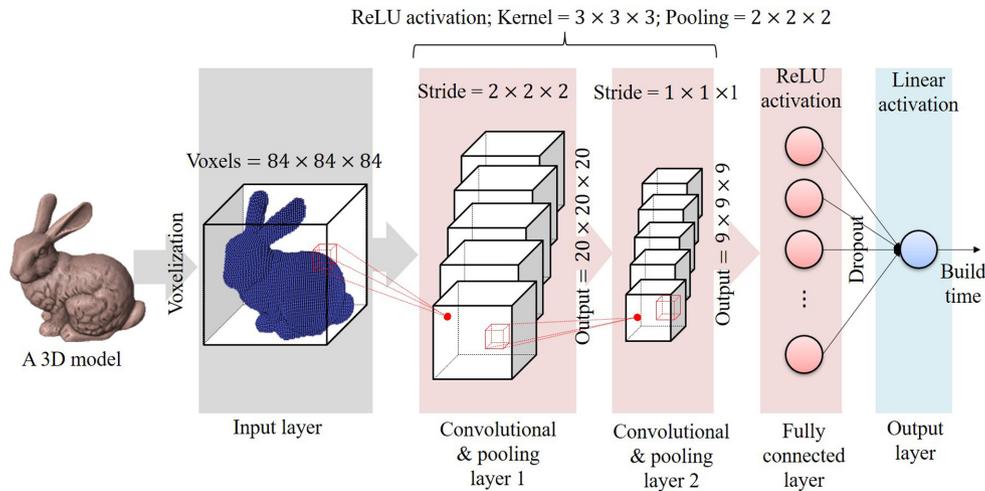


Figure 7: The architecture of CNN to estimate build time.

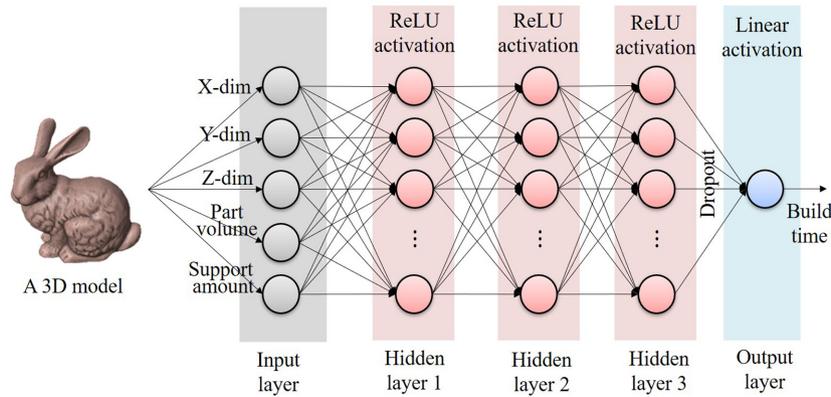


Figure 8: The architecture of ANN to estimate build time.

## 4.2 ANN architecture

Depending on the type of input features, we design two ANNs: ANN with RFs and ANN with voxels. In the remaining sections, we simply name them as ANN<sup>r</sup> and ANN<sup>v</sup>, respectively. Figure 8 illustrates the architecture of ANN<sup>r</sup> for estimating build time. For the ANN<sup>r</sup>, the input layer consists of the five normalized RFs, dimensions X, Y, and Z, the volume of a part, and the amount of support structure. While the overall ANN<sup>r</sup> design comes from the work of Munguía *et al.* (2009), we added a new input feature, the amount of support structure, that is a critical factor in build time estimation. Similar to the CNN structure, the output layer of ANN<sup>r</sup> also has only one output neuron in which the activation function is linear. While Munguía *et al.* considered only one hidden layer, we deploy three layers for ANN<sup>r</sup> to improve the prediction performance based on accumulated experimental data. The neurons in the hidden layers have ReLU for their activation function.

The architecture of ANN<sup>v</sup> has the input layer for voxel data. As such, instead of the five RFs, 592,704 voxels for each model are entered to ANN<sup>v</sup>. Therefore, the design of the input layer is the same as CNN. Except for the input layer, others including hidden layers and the output layer are the same as ANN<sup>r</sup>.

## 4.3 Hyperparameters for NNs

In ML, hyperparameters (e.g. dropout rate and the number of neurons) are parameters that are used to control a learning process. Depending on hyperparameters, the model of NNs is differently trained, which affects the prediction outcome. In other words, the prediction performance of ANN<sup>r</sup>, ANN<sup>v</sup>, and CNN depends on their hyperparameters. In this paper, the three NNs are compared based on their best performance. To obtain the best performance for each NN, we look for the optimal combination of hyperparameters to minimize the loss function by altering hyperparameters. To tune hyperparameters, we adopt TensorFlow's HParams plugin. ([https://www.tensorflow.org/tensorboard/hyperparameter\\_tuning\\_with\\_hparams](https://www.tensorflow.org/tensorboard/hyperparameter_tuning_with_hparams))

In this paper, we tune the number of neurons and the dropout rate for the ANN<sup>r</sup> and ANN<sup>v</sup> models. In the CNN model, the number of filters in convolutional layers 1 and 2 and the dropout rate are tuned. Table 3 represents options that can be chosen for each hyperparameter. To reduce computation load for processing a significant number of voxels, ANN<sup>v</sup> and CNN have only two options for the number of neurons and filters while ANN<sup>r</sup> has three options. Depending on the four cases and the two slicing applications, we differently choose the hyperparameter options that minimize a loss function. In the

Table 3: Hyperparameter options for ANN<sup>r</sup>, ANN<sup>v</sup>, and CNN.

|                  | Hyperparameters                                       | Options        |
|------------------|---|----------------|
| ANN <sup>r</sup> | The number of neurons in hidden layers 1, 2, and 3    | [16, 128, 256] |
|                  | Dropout rate (%)                                      | [0.1, 0.5]     |
| ANN <sup>v</sup> | The number of neurons in hidden layers 1, 2, and 3    | [32, 256]      |
|                  | Dropout rate (%)                                      | [0.1, 0.5]     |
| CNN              | The number of filters in convolutional layers 1 and 2 | [16, 64]       |
|                  | The number of neurons in fully connected layer        | [32, 128]      |
|                  | Dropout rate (%)                                      | [0.1, 0.5]     |

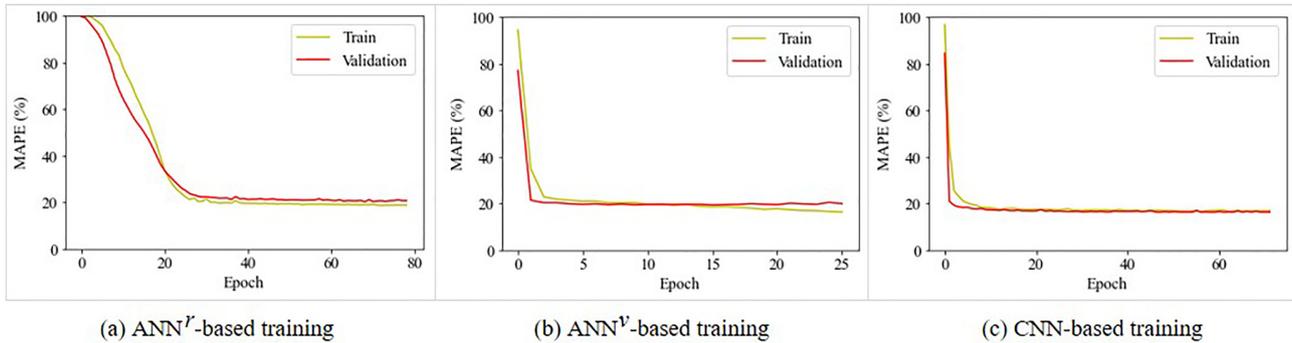


Figure 9: Training results for design treatment SR: MAPE for train and validation loss (%).

CNN design, some hyperparameters are referred from the work of Williams et al. (2019) such as pooling size (2, 2, 2) and filter (or kernel) size (3, 3, 3).

## 5 Computational Experiment

In this section, we provide computational experiments to compare the performances of CNN, ANN<sup>r</sup>, and ANN<sup>v</sup>. The design of the three NNs explained in Section 4 is implemented in Tensorflow 2.1.0 (<https://www.tensorflow.org/>) Python 3.7.7 (<https://www.python.org/>) and Keras 2.2.4 (<https://keras.io/>) Sections 5.1 describes how the NNs are trained and tested. In Section 5.4, we discuss about test results.

### 5.1 Training NN models

An input dataset includes a variety of samples depending on the four design treatments (N, S, R, and SR). As explained Section 3.1, each design treatment includes 2,947 samples. For label data, the build time for each sample (a 3D model) is simulated by the two slicing applications, PrusaSlicer and UltimakerCura. Moreover, we consider three types of NNs (CNN, ANN<sup>r</sup>, and ANN<sup>v</sup>). 2,947 samples for each design treatment are trained and tested as an experiment. Therefore, we have 24 experiment sets (4 design treatments × 3 NN types × 2 slicing applications).

The input dataset of 2,947 samples is split into a train set and a test set with a ratio of 9:1. As such, the train and test sets include 2,652 and 295 samples, respectively. When splitting it, the last 10% of samples in the dataset are included in a test set. This is to make sure that the three NN models have the same test set for a fair comparison. For the train set, 10% of the 2,652 samples are randomly chosen for validation and then the samples are shuffled for each training epoch. The number of epochs and training batch size are 150 and 20, respectively. During training, *mean absolute percentage error* (MAPE; see equation 2) is considered for a loss function. Note that *mean squared*

*error* is not considered in this paper since it generates too large numbers, which results from the build time is measured by unit seconds and the difference between actual and predicted values is squared. This makes it difficult for people to recognize the meaning of the numbers. In equation (2),  $Y_i$ ,  $\hat{Y}_i$ , and  $n$  are an actual value, a predicted value, and the number of data, respectively. Training is conducted with the Adam optimizer (Kingma & Ba, 2014) used in the previous study (Williams et al., 2019).

$$\text{MAPE} = \frac{100}{n} \sum_{i=1}^n \left| \frac{Y_i - \hat{Y}_i}{Y_i} \right| \quad (2)$$

In ML, learning curves consisting of training and validation scores help us to identify whether a NN model is well trained. Figure 9 represents learning curves for design treatment SR. In these figures, if a training curve goes under a validation curve and the gap between the two curves is large, it means that a NN model is too dependent on training data and it is likely to have a higher error rate on test data, which is known as *overfitting*. On the other hand, if a training curve goes over a validation curve with a large gap between the two curves, it indicates that a NN model is too simple to train the given dataset, which is known as *underfitting*. To avoid overfitting during training, we adopt two methods: dropout and stop condition. The dropout is a technique to randomly drop some units in a NN during training (Srivastava et al., 2014). The stop condition is to stop a training process if there is no improvement for certain epochs. In the experiments, the dropout rate is tuned for each NN as discussed in Section 4.3 and the stop condition is set as 10 epochs. Figure 9 shows that the NN models for design treatment SR are well trained without overfitting or underfitting. This is also shown in the learning curves of other design treatments.

### 5.2 Test with a new dataset

After the training process, 295 samples in the test dataset are used for the prediction test. Figure 10 represents the relation

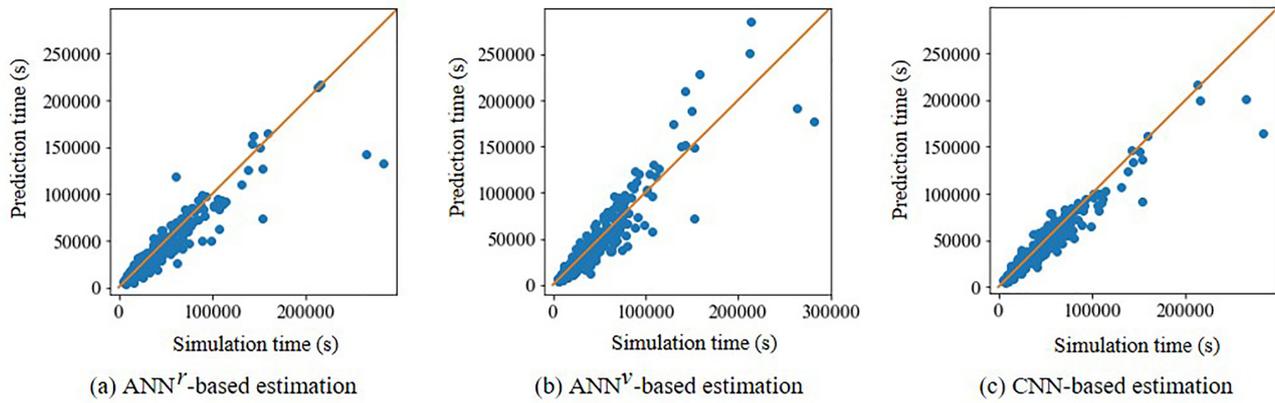


Figure 10: Test results: the plotting graph of prediction and simulation build time (BT) for design treatment SR.

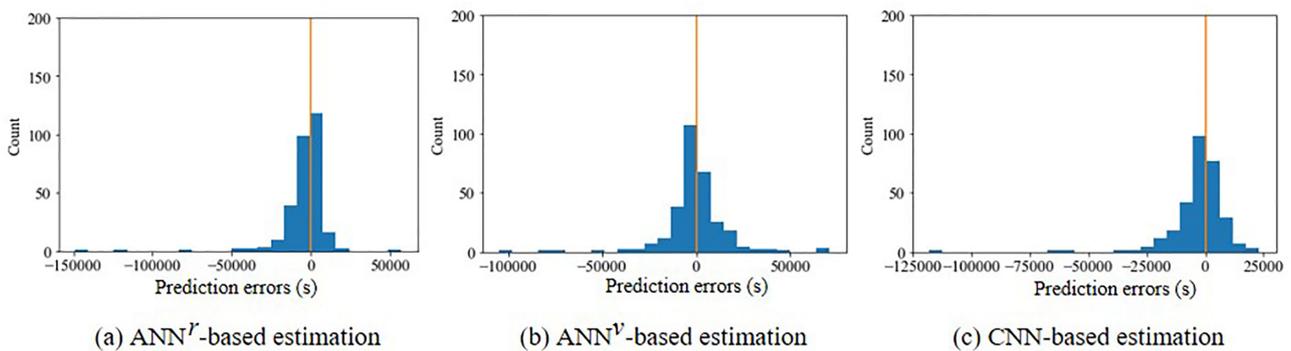


Figure 11: Test results: the error distribution of prediction and simulation build time (BT) for design treatment SR.

between prediction and simulation build times. The simulation build time is label data estimated by slicing applications. The three figures are the test results of ANN<sup>r</sup>, ANN<sup>v</sup>, and CNN-based estimation for design treatment SR with labels from PrusaSlicer. The closer the plotting dots on the diagonal, the less error they cause. The gaps between the prediction and simulation values are illustrated as error distributions with a bin size of 25 in Fig. 11. The Gaussian distributions show the variation of errors. In this case, the CNN-based estimation provides less error variation compared to others.

### 5.3 Part fabrication

As shown in Fig. 12, our paper focuses on two types of build time: build time calculated by slicing software (simulation time) and build time predicted by a NN model (prediction time). The NN model is trained based on the simulation time. Thus, if the NN model is well trained, the prediction time will be close to the simulation time. To validate how the NN model accurately predicts, a new dataset of simulation time is used, which is addressed in Section 5.2 (see Figs 10 and 11). It should be noted that actual build time is not necessary in the training and test processes for our study.

However, to show a gap between the virtual and real worlds, we chose three samples from the test dataset (295 samples) and fabricated the three parts based on an ME 3D printer. As shown in Fig. 13, when comparing simulation and prediction time, error rates are 2.93% of Lion, 5.22% of Banana, and 1.16% of Angel. For the comparisons of actual and simulation times, error rates vary from 0.35% of Banana to 15.43% of Angel. For Angel, the

large difference of actual and simulation times could be affected by the geometry of a model or the status of a 3D printer. However, the comparison between actual and simulation times is not deeply addressed in this paper since it is not our focus. Note that directly comparing prediction time with actual time is meaningless since our NN model is trained based on simulation time.

## 5.4 Result and discussion

### 5.4.1 Comparison of NN-based regression models

Coefficient of determination, denoted  $R^2$ , is a widely used measure to judge the adequacy of a regression model (Montgomery & Runger, 2013).  $R^2$  is based on the proportion of total variation of outcomes explained by the model. This measure is useful to find the likelihood of future events falling within the predicted outcomes (Williams *et al.*, 2019).  $R^2$  normally ranges from 0 to 1 and, if  $R^2$  is equal to 1, it is a perfect prediction. In addition, to judge the adequacy of a regression model, we consider one more measure, MAPE, that is based on the difference between actual and predicted values. The MAPE represents an error rate as the 100 scales.

Tables 4 and 5 show  $R^2$  and MAPE of the three NNs (ANN<sup>r</sup>, ANN<sup>v</sup>, and CNN) depending on the four design treatments (N, R, S, and SR). The tables include outcomes for PrusaSlicer and UltimakerCura and the averages of the two slicing applications. Note that we have 24 experiment sets (4 design treatments  $\times$  3 NN types  $\times$  2 slicing applications) and each set is repeated 5 times. In Tables 4 and 5,  $R^2$  and MAPE are the means of five repeats of training and testing processes. Figure 14 visually presents  $R^2$  and MAPE that are the averages of the two

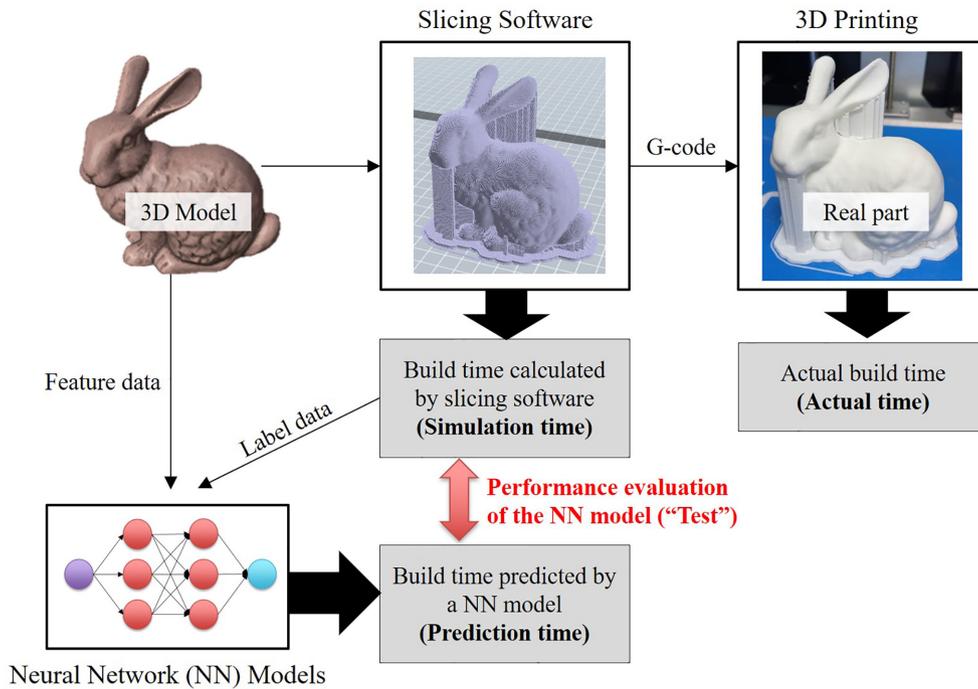


Figure 12: Definition of simulation time, prediction time, and actual time.

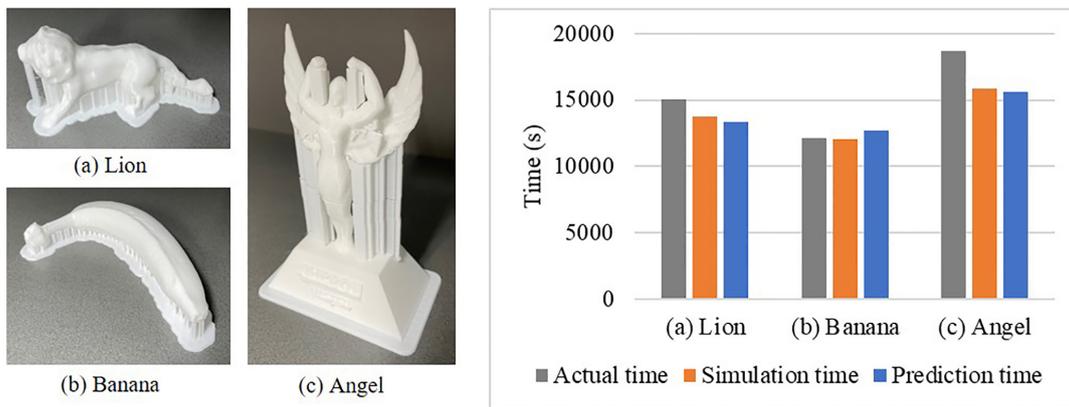


Figure 13: Part fabrication: simulation time, prediction time, and actual time.

Table 4:  $R^2$  of the three NN models ( $ANN^r$ ,  $ANN^v$ , and CNN) depending on the design treatments (N, R, S, and SR).

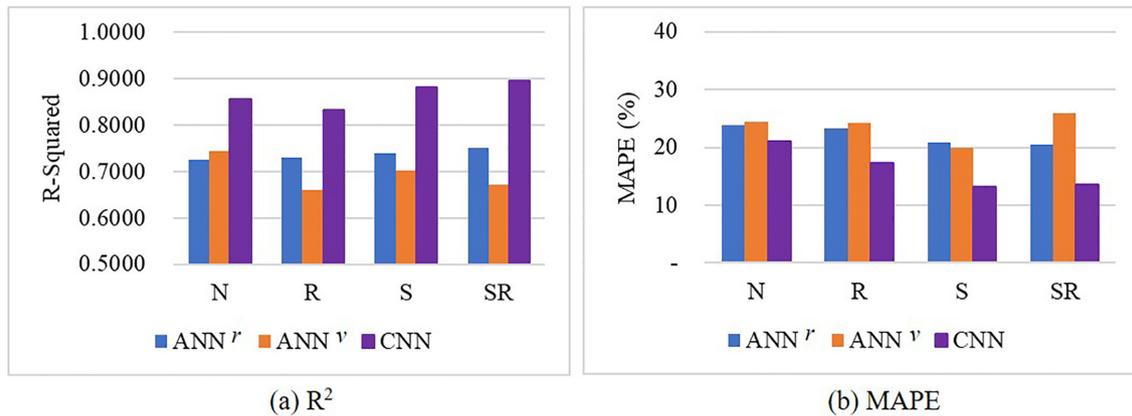
|    | PrusaSlicer |         |        | UltimakerCura |         |        | Average |         |        |
|----|-------------|---------|--------|---------------|---------|--------|---------|---------|--------|
|    | $ANN^r$     | $ANN^v$ | CNN    | $ANN^r$       | $ANN^v$ | CNN    | $ANN^r$ | $ANN^v$ | CNN    |
| N  | 0.5797      | 0.6014  | 0.7606 | 0.8696        | 0.8895  | 0.9545 | 0.7246  | 0.7455  | 0.8575 |
| R  | 0.6437      | 0.5322  | 0.7435 | 0.8169        | 0.7887  | 0.9249 | 0.7303  | 0.6605  | 0.8342 |
| S  | 0.6132      | 0.5323  | 0.8108 | 0.8684        | 0.8724  | 0.9555 | 0.7408  | 0.7023  | 0.8832 |
| SR | 0.6578      | 0.5195  | 0.8443 | 0.8451        | 0.8268  | 0.9473 | 0.7515  | 0.6731  | 0.8958 |

applications from Tables 4 and 5. With the figure, we can recognize which NN model is superior for a certain design treatment. It shows that the CNN-based estimation provides higher  $R^2$  and smaller MAPE than the other NN models, which means that we can obtain more accurate build time prediction outcomes based on the CNN model.

In Fig. 14a, when the CNN model is applied to design treatment SR,  $R^2$  is 0.8958, which is 0.2227 larger than the  $ANN^v$  model. Moreover, in Fig. 14b, when considering the CNN model for design treatment SR, MAPE is 13.55% that is 12.41% smaller than the  $ANN^v$  model.

**Table 5:** MAPE (%) of the three NN models (ANN<sup>r</sup>, ANN<sup>v</sup>, and CNN) depending on the design treatments (N, R, S, and SR).

|    | PrusaSlicer      |                  |       | UltimakerCura    |                  |       | Average          |                  |       |
|----|------------------|------------------|-------|------------------|------------------|-------|------------------|------------------|-------|
|    | ANN <sup>r</sup> | ANN <sup>v</sup> | CNN   | ANN <sup>r</sup> | ANN <sup>v</sup> | CNN   | ANN <sup>r</sup> | ANN <sup>v</sup> | CNN   |
| N  | 29.05            | 31.21            | 27.79 | 18.58            | 17.64            | 14.11 | 23.82            | 24.43            | 20.95 |
| R  | 26.02            | 29.86            | 20.65 | 20.50            | 18.50            | 13.99 | 23.26            | 24.18            | 17.32 |
| S  | 23.87            | 24.85            | 16.37 | 17.77            | 15.01            | 10.00 | 20.82            | 19.93            | 13.19 |
| SR | 22.68            | 31.57            | 15.82 | 18.20            | 20.35            | 11.28 | 20.44            | 25.96            | 13.55 |

**Figure 14:** Comparison of the three NN models (ANN<sup>r</sup>, ANN<sup>v</sup>, and CNN): (a) R<sup>2</sup> (the averages from Table 4) and (b) MAPE (the averages from Table 5).

#### 5.4.2 Significance of differences among the NN models

To identify that the difference among the NN models is significant, we apply one-way *analysis of variance* (ANOVA) test with a 0.05 significance level. For the one-way ANOVA, we use percent errors as a dataset. A percent error is an element of MAPE. As explained in equation (2), MAPE is the mean of percent errors and a percent error is based on the difference between actual and predicted values for each sample. The dataset is comprised of 885 percent errors (295 percent errors  $\times$  3 NN types). With the dataset, we apply the ANOVA test for each design treatment. Note that each percent error is the average for the two slicing applications, the PrusaSlicer and UltimakerCura.

Table 6 presents the mean,  $\mu$ , and standard deviation,  $\sigma$ , of 295 samples for each NN model and the P-value for each ANOVA test. Herein,  $\mu$  is MAPE explained in Section 5.4.1. As shown in Table 6, the difference is significant in most cases. Furthermore, we adopt Fisher's least significant difference (LSD) method to compare all pairwise differences. The grouping information for all pairwise differences is presented in Table 7. If a difference between two models is not significant, the models are classified into the same group. As a result, the CNN-based estimation is classified into a different group in design treatments R, S, and SR. This means that the CNN-based estimation for build time gives significantly less error rate compared to the other ANN models in most cases. We hypothesize that the high performance of the CNN model is due to the ability of a CNN extracting spatial features.

#### 5.4.3 The effect of design factors

As discussed in Section 3.1, the four design treatments are distinguished by the combination of two design factors (i.e. scale and rotation) and two levels (i.e. low and high). To identify the effect of the alterations of scale and rotation on constructed 3D models, we apply a two-way ANOVA test with a significance level

of 0.05. For the two-way ANOVA, we use the percent errors as a dataset as explained in Section 5.4.2. The dataset is comprised of 1,180 percent errors (295 percent errors  $\times$  4 design treatments). With the dataset, we apply the two-way ANOVA test for each NN model.

Table 8 presents the P-value of the ANOVA test for each NN model. This table shows that scale is clearly a significant design factor for CNN. In other words, if the size of 3D models is large, the CNN-based estimation tends to show small errors. This is mainly because the CNN model requires large enough a 3D model to extract spatial features and accurately estimate build time. In short, to accurately estimate build time based on CNN, large parts would be more effective. The tendency is also shown in Tables 6 and 7 that the CNN model has a small MAPE in design treatment S or SR when 3D models are large. However, in design treatment N when 3D models are small, the MAPE of the CNN model is not much different from the other ANN models. Before the experiments, we expected that rotation would be significant for the CNN model. This is because a CNN can recognize the rotation features of 3D models and support structure generated by rotation affects build time. However, Table 8 shows that rotation for the CNN model does not have a significant influence on reducing errors for build time estimation. This could be caused by the fact that the effect of support structure on build time is not significant. Although rotating a model from the initial orientation generates more support structure, it is hard to tell that the amount of support structure has a significant effect on estimating build time.

#### 5.4.4 Relations between training and prediction design treatments

In the previous sections, we assumed that the training and test sets are the same. In this section, we differently apply the training and test sets. This is to investigate relations between training and prediction design treatments, which is shown in the work

**Table 6:** MAPE (%) for the three estimation models and P-value of one-way ANOVA test.

| Design treatments | ANN <sup>r</sup> |          | ANN <sup>v</sup> |          | CNN   |          | P-value |
|-------------------|------------------|----------|------------------|----------|-------|----------|---------|
|                   | $\mu$            | $\sigma$ | $\mu$            | $\sigma$ | $\mu$ | $\sigma$ |         |
| N                 | 23.82            | 17.70    | 24.62            | 24.05    | 20.95 | 23.50    | 0.1030  |
| R                 | 23.26            | 15.97    | 24.12            | 15.83    | 17.32 | 11.84    | <0.0001 |
| S                 | 20.82            | 16.11    | 19.98            | 14.14    | 13.19 | 10.04    | <0.0001 |
| SR                | 20.44            | 14.59    | 25.88            | 16.28    | 13.55 | 9.18     | <0.0001 |

**Table 7:** Grouping information using the Fisher LSD method.

| Design treatment | ANN <sup>r</sup>      | ANN <sup>v</sup>      | CNN                   |
|------------------|-----------------------|-----------------------|-----------------------|
| N                | Group A <sup>N</sup>  |                       | Group B <sup>N</sup>  |
| R                | Group A <sup>R</sup>  |                       | Group B <sup>R</sup>  |
| S                | Group A <sup>S</sup>  |                       | Group B <sup>S</sup>  |
| SR               | Group A <sup>SR</sup> | Group B <sup>SR</sup> | Group C <sup>SR</sup> |

**Table 8:** P-values of two-way ANOVA test for percent errors.

|                    | ANN <sup>r</sup> | ANN <sup>v</sup> | CNN     |
|--------------------|------------------|------------------|---------|
| Scale              | 0.0020           | 0.1709           | <0.0001 |
| Rotation           | 0.6184           | 0.0101           | 0.0586  |
| Interaction effect | 0.9258           | 0.0023           | 0.0208  |

of Williams *et al.* (2019). However, we have two different points from their work. First, our design treatments are different from the previous work by considering scale as a design factor. Moreover, we adopt MAPE as a measure for confusion matrices while the previous work used  $R^2$ . This is mainly because  $R^2$  could be inflated so that it may not be useful to determine the appropriateness of a regression model (Montgomery & Runger, 2013).

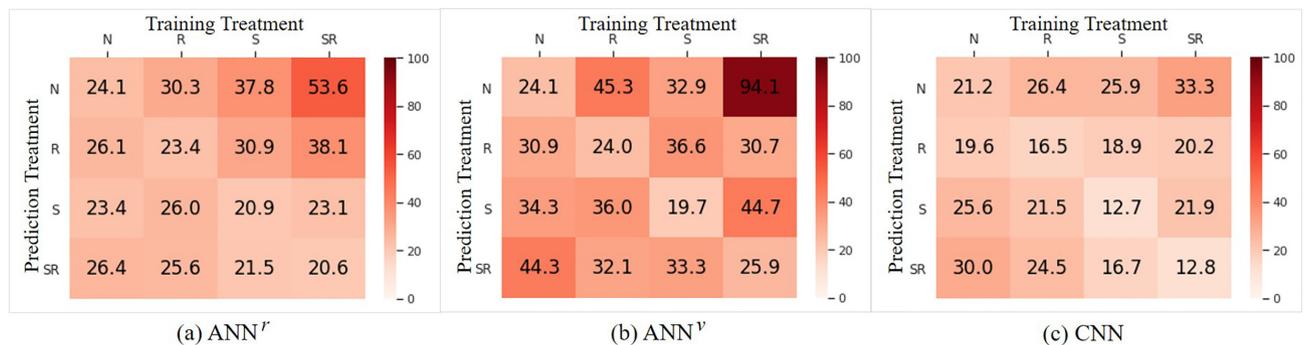
Figure 15 presents confusion matrices for the three NN models. Note that a value in the confusion matrices is the average of MAPEs for the two applications. For most cases in Fig. 15, CNN-based estimation provided smaller MAPEs compared to

the other NN models. In addition, we mostly obtained smaller MAPEs when prediction and training treatments are the same. When it comes to the CNN model (see Fig. 15c), the result can be compared to the work of Williams *et al.* (2019). Our result shows a similar tendency with the previous work in terms of having mostly positive outcomes when prediction and training treatments are the same. In our result, however, this tendency is more clearly represented by showing that the most positive outcomes are focused on the diagonal numbers in the confusion matrix.

We hypothesize that this difference is due to the label data. In our study, label data, namely build time, is created by slicing applications considering a variety of factors including process parameters and a tool path. However, the previous study calculated build time for label data based on a couple of parameters. This could result in that the calculated build time of the previous work is dominated by some major factors (especially the volume of a part). In this case, if a regression model has a large weight on the major factors, we would be able to expect positive outcomes for prediction tests.

## 6 Conclusion

In this paper, we compared three types of NNs for build time estimation: ANN<sup>r</sup>, ANN<sup>v</sup>, and CNN. To investigate the effectiveness of input geometries on estimation results, we distinguished four design treatments based on scale and rotation. For feature data, representative geometric features and voxel data were considered. Moreover, slicing applications were adopted to obtain build time for label data. In the computational experiments, we trained the NN-based regression models and tested the accuracy of prediction. In most cases, CNN-based estimation showed better performance than the other ANN models. Moreover, we revealed that the CNN-based estimation is sensitive to the size of

**Figure 15:** Confusion matrix (PrusaSlicer and UltimakerCura combined) for percent errors (%) when predicting build time from all design treatments: (a) ANN<sup>r</sup>; (b) ANN<sup>v</sup>; and (c) CNN.

3D models. This means that 3D models should be large enough to increase the performance of the CNN model.

Our build time estimation methods have a limitation that it is challenging to be generalized. If we want to estimate build time for another type of AM process with different process parameters, our NN models should be trained from the beginning. Our NN models for build time estimation are not appropriate in a situation where AM process types and process parameters change frequently. However, our methods are more appropriate in another situation that build time estimation is repeatedly, rapidly or roughly needed with little change in process parameters. For instance, our estimation methods can be applied to a 3D printing farm that usually runs dozens or hundreds of the same 3D printers with similar process parameters. As another example, build time in the early design stage is roughly estimated since specific process parameters are not available. In such case, the default process parameters that are widely used for build time estimation are often used.

For future work, this paper leaves further topics spanning three research directions. First, instead of simulation time based on slicing applications, actual build time can be used for label data. However, this could be costly and time consuming to obtain a large enough number of data for the training and testing of NNs. Second, except for an ME process, other AM processes (e.g. vat polymerization and powder bed fusion) could be considered. Third, the proposed build time estimation could also be adopted for other time-critical tasks such as part maintenance (Kim et al., 2019a, b) or 3D printing farm (Skawiński & Siemiński, 2017). Lastly, the effect of process parameters (e.g. layer thickness) on the performance of NNs could be investigated. In this study, we used consistent values for key process parameters such as layer thickness, although two different virtual 3D printers were considered. However, by altering the parameter values, the effect of process parameters could be analysed.

## Acknowledgments

This work was supported by the GRRC program of Gyeonggi province in South Korea (GRRC KGU 2020-B01: Research on Intelligent Industrial Data Analytics), and by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT; No. 2021R1G1A1012481).

Commercial equipment and materials might be identified to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the U.S. National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

We thank Allison Barnard Feeney and Vijay Srinivasan at NIST, Hyunseop Park at Pohang University of Science and Technology, and Dongmin Shin at Hanyang University for their valuable comments and input.

## Conflict of interest statement

None declared.

## REFERENCES

- Aggarwal, C. C. (2018). *Neural networks and deep learning: A textbook*. (1st ed.). Springer.
- Alexander, P., Allen, S., & Dutta, D. (1998). Part orientation and build cost determination in layered manufacturing. *Computer-Aided Design*, 30(5), 343–356.
- Anitescu, C., Atroshchenko, E., Alajlan, N., & Rabczuk, T. (2019). Artificial neural network methods for the solution of second order boundary value problems. *Computers, Materials and Continua*, 59(1), 345–359.
- Baumann, F. W. (2018). Thirty Thousand 3D Models from Thingiverse. *Data*, 3(1), 5.
- Caggiano, A., Zhang, J., Alfieri, V., Caiazzo, F., Gao, R., & Teti, R. (2019). Machine learning-based image processing for on-line defect recognition in additive manufacturing. *CIRP Annals*, 68(1), 451–454.
- Ceruti, A., Marzocca, P., Liverani, A., & Bil, C. (2019). Maintenance in aeronautics in an industry 4.0 context: The role of augmented reality and additive manufacturing. *Journal of Computational Design and Engineering*, 6(4), 516–526.
- Chen, C. C., & Sullivan, P. A. (1996). Predicting total build-time and the resultant cure depth of the 3D stereolithography process. *Rapid Prototyping Journal*, 2(4), 27–40.
- Choi, S. H., & Samavedam, S. (2002). Modelling and optimisation of rapid prototyping. *Computers in Industry*, 47(1), 39–53.
- Di Angelo, L., & Di Stefano, P. (2011). A neural network-based build time estimator for layer manufactured objects. *The International Journal of Advanced Manufacturing Technology*, 57(1–4), 215–224.
- Eranpurwala, A., Ghiasian, S. E., & Lewis, K. (2020). Predicting build orientation of additively manufactured parts with mechanical machining features using deep learning. In *Proceedings of the ASME 2020 IDETC/CIE Conference*.
- Francis, J., & Bian, L. (2019). Deep learning for distortion prediction in laser-based additive manufacturing using big data. *Manufacturing Letters*, 20, 10–14.
- Giannatsis, J., Dedoussis, V., & Laios, L. (2001). A study of the build-time estimation problem for stereolithography systems. *Robotics and Computer-Integrated Manufacturing*, 17(4), 295–304.
- Gogate, A., & Pande, S. (2008). Intelligent layout planning for rapid prototyping. *International Journal of Production Research*, 46(20), 5607–5631.
- Guo, H., Zhuang, X., & Rabczuk, T. (2019). A deep collocation method for the bending analysis of Kirchhoff plate. *Computers, Materials and Continua*, 59(2), 433–456.
- Imani, F., Chen, R., Diewald, E., Reutzel, E., & Yang, H. (2019). Deep learning of variant geometry in layerwise imaging profiles for additive manufacturing quality control. *Journal of Manufacturing Science and Engineering*, 141(11), 111001.
- Kadir, A. Z. A., Yusof, Y., & Wahab, M. S. (2020). Additive manufacturing cost estimation models—a classification review. *The International Journal of Advanced Manufacturing Technology*, 107(9), 4033–4053.
- 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(5–8), 2555–2569.
- Khorram Niaki, M., & Nonino, F. (2017). Additive manufacturing management: a review and future research agenda. *International Journal of Production Research*, 55(5), 1419–1439.
- Kim, H., Cha, M., Kim, B. C., Kim, T., & Mun, D. (2019a). Part library-based information retrieval and inspection framework to support part maintenance using 3D printing technology. *Rapid Prototyping Journal*, 25(3), 630–644.
- Kim, H., Cha, M., Kim, B. C., Lee, I., & Mun, D. (2019b). Maintenance Framework for Repairing Partially Damaged Parts Using 3D Printing. *International Journal of Precision Engineering and Manufacturing*, 20(8), 1451–1464.

- Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*, San Diego.
- Kwon, O., Kim, H. G., Kim, W., Kim, G.-H., & Kim, K. (2020). A convolutional neural network for prediction of laser power using melt-pool images in laser powder bed fusion. *IEEE Access*, 8, 23255–23263.
- Medina-Sanchez, G., Dorado-Vicente, R., Torres-Jiménez, E., & López-García, R. (2019). Build time estimation for fused filament fabrication via average printing speed. *Materials*, 12(23), 3982.
- Montgomery, D. C., & Runger, G. C. (2013). *Applied statistics and probability for engineers*. (6th ed.). Wiley.
- Munguía, J., Ciurana, J., & Riba, C. (2009). Neural-network-based model for build-time estimation in selective laser sintering. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 223(8), 995–1003.
- Oh, Y. (2019). *Assembly design and production planning towards additive manufacturing-based mass customization*, Ph.D. thesis, State University of New York at Buffalo.
- Oh, Y., Ko, H., Sprock, T., Bernstein, W. Z., & Kwon, S. (2021). Part decomposition and evaluation based on standard design guidelines for additive manufacturability and assemblability. *Additive Manufacturing*, 37, 101702.
- Ruffo, M., Tuck, C., & Hague, R. (2006). Empirical laser sintering time estimator for duraform PA. *International Journal of Production Research*, 44(23), 5131–5146.
- Samaniego, E., Anitescu, C., Goswami, S., Nguyen-Thanh, V. M., Guo, H., Hamdia, K., Zhuang, X., & Rabczuk, T. (2020). An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Computer Methods in Applied Mechanics and Engineering*, 362, 112790.
- Scime, L., & Beuth, J. (2018). A multi-scale convolutional neural network for autonomous anomaly detection and classification in a laser powder bed fusion additive manufacturing process. *Additive Manufacturing*, 24, 273–286.
- Seo, G., Ahsan, M. R., Lee, Y., Shin, J.-H., Park, H., & Kim, D. B. (2021). A functional modeling approach for quality assurance in metal additive manufacturing. *Rapid Prototyping Journal*, 27(2), 288–303.
- Skawiński, P., & Siemiński, P. (2017). The 3D Printer Farm – function and technology requirements and didactic use. *Mechanik*, 90(8–9), 796–800.
- Spruegel, T. C., Bickel, S., Schleich, B., & Wartzack, S. (2021). Approach and application to transfer heterogeneous simulation data from finite element analysis to neural networks. *Journal of Computational Design and Engineering*, 8(1), 298–315.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 1929–1958.
- Stolt, R., & Elgh, F. (2020). Introducing design for selective laser melting in aerospace industry. *Journal of Computational Design and Engineering*, 7(4), 489–497.
- Valueva, M., Nagornov, N., Lyakhov, P., Valuev, G., & Chervyakov, N. (2020). Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*, 177, 232–243.
- Vania, M., Mureja, D., & Lee, D. (2019). Automatic spine segmentation from CT images using Convolutional Neural Network via redundant generation of class labels. *Journal of Computational Design and Engineering*, 6(2), 224–232.
- Wang, Y., Huang, J., Wang, Y., Feng, S., Peng, T., Yang, H., & Zou, J. (2020). A CNN-Based Adaptive Surface Monitoring System for Fused Deposition Modeling. *IEEE/ASME Transactions on Mechatronics*, 25(5), 2287–2296.
- 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(11), 111701.
- Worrall, D. E., Garbin, S. J., Turmukhambetov, D., & Brostow, G. J. (2017). Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5028–5037).
- Zhang, Y., & Bernard, A. (2013). Generic build time estimation model for parts produced by SLS. In *High Value Manufacturing: Advanced Research in Virtual and Rapid Prototyping. Proceedings of the 6th International Conference on Advanced Research in Virtual and Rapid Prototyping* (pp. 43–48).
- Zhang, Y., Bernard, A., Valenzuela, J. M., & Karunakaran, K. (2015). Fast adaptive modeling method for build time estimation in additive manufacturing. *CIRP Journal of Manufacturing Science and Technology*, 10, 49–60.
- Zhang, Z., Jaiswal, P., & Rai, R. (2018). FeatureNet: Machining feature recognition based on 3D convolution neural network. *Computer-Aided Design*, 101, 12–22.
- Zhang, B., Jaiswal, P., Rai, R., Guerrier, P., & Baggs, G. (2019a). Convolutional neural network-based inspection of metal additive manufacturing parts. *Rapid Prototyping Journal*, 25(3), 530–540.
- Zhang, B., Liu, S., & Shin, Y. C. (2019b). In-process monitoring of porosity during laser additive manufacturing process. *Additive Manufacturing*, 28, 497–505.
- Zhang, Y., & Moon, S. K. (2021). Data-driven design strategy in fused filament fabrication: status and opportunities. *Journal of Computational Design and Engineering*, 8(2), 489–509.