Image-based failure detection for material extrusion process using a convolutional neural network

Hyungjung Kim¹ · Hyunsu Lee² · Ji-Soo Kim² · Sung-Hoon Ahn^{2,3}

Received: 24 June 2020 / Accepted: 29 September 2020 / Published online: 9 October 2020
© The Author(s) 2020

Abstract

The material extrusion (ME) process is one of the most widely used 3D printing processes, especially considering its use of inexpensive materials. However, the error known as the “spaghetti-shape error,” related to filament tangling, is a common problem associated with the ME process. Once occurring, this issue, which consumes both time and materials, requires a restart of the entire process. In order to prevent this, the user must constantly monitor the process. In this research, a failure detection method which uses a webcam and deep learning is developed for the ME process. The webcam captures images and then analyzes them by machine learning based on a convolutional neural network (CNN), showing outstanding performance in both image classification and the recognition of objects. Sample images were trained based on a modified Visual Geometry Group Network (VGGNet) model and the trained model was evaluated, resulting in 97% accuracy. The pre-trained model was tested on a 3D printer monitoring system for its ability to recognize the “spaghetti-shape-error” and was able to detect 96% of abnormal deposition processes. The proposed method can analyze the ME process in real time and informs the user or halts the process when abnormal printing is detected.

Keywords 3D printing process · Material extrusion · Failure detection · Process monitoring · Deep learning · Convolutional neural network

1 Introduction

1.1 Global trend about 3D printer

Currently, considering the “Fourth Industrial Revolution,” 3D printing, or additive manufacturing, is ready to emerge from its niche status and become a viable alternative to conventional manufacturing processes in an increasing number of applications. In fact, it is now an enabling technology in smart factories and in cloud manufacturing [1, 2]. The advantages of 3D printing over other conventional manufacturing

technologies are leading to significant changes in product development processes. This approach uses direct digital manufacturing processes that directly transform 3D data into actual parts without requiring tools or molds [3]. Additionally, the layer manufacturing principle can also produce functionally integrated parts in a single production step, reducing the need for assembly activities [4]. This technology can transform manufacturing companies by, for example, reducing the time required for product development, allowing changes of product manufacturing strategies, and enabling the customization of products [5].

There are several different processes developed for 3D printers, such as material extrusion (ME), vat photopolymerization (VP), and power bed fusion (PBF), among others, each with its own unique set of competencies and limitations [6]. Among these printing processes, the ME process is one of the most commonly used 3D printing processes for the fabrication of pure plastic parts at a low cost and with minimal material usage and ease of material changes. Moreover, this ME process is widely used with both low-cost desktop 3D printers and high-end industrial 3D printers [7–9]. In the ME process, a part is produced by extruding a molten material which forms layers as the material hardens. As shown in Fig. 1, in 2018, more than \$9 billion of global value related to

Hyungjung Kim and Hyunsu Lee share equally first authorship

✉ Sung-Hoon Ahn
ahnsh@snu.ac.kr

¹ Sales Division, Doosan Robotics Inc., Suwon 16648, Republic of Korea

² Department of Mechanical Engineering, Seoul National University, Seoul 08826, Republic of Korea

³ Institute of Advanced Machines and Design, Seoul National University, Seoul 08826, Republic of Korea

3D printers was generated due to the simplicity and affordability of this process [2, 10]. Also, in 2018, ME process accounted for 68% of the total 3D printer market.

Although ME process is now a mature production process, there is a certain level of failure related to low-skilled users or errors in the ME process printers, which can increase the use of resources such as time, energy, and materials [11, 12]. For instance, a failure rate of 20% can lead to longer printing times [13]. This can raise the overall cost of the final part. Specifically, the “spaghetti-shape-error,” mentioned in the Abstract, related to filament tangling, requires a restart of the entire build process. However, in the absence of real-time process monitoring, quality control in the ME process is mostly limited to offline techniques, leading to high scrap rates during production [14]. Also, to boost the digital transformation in the factory floor, especially for small- and medium-sized enterprises (SMEs), an appropriate monitoring solution could be an affordable way in the current market challenges. Therefore, there are strong needs to develop failure detection techniques for the ME process. To enhance the quality and usability of the process and to reduce the energy, time, and material losses, the goal of this work is to develop a failure detection method for the spaghetti-shape-error in the ME process using a machine learning without a significant change of the ME process and the need for expert knowledge to integrate additional expensive sensors on the ME process printers.

1.2 Fault detection in the ME process

Given that many users of commercial ME process printers are not skilled with the 3D printing process, actual material waste levels could be greater than those under ideal operating conditions without human or printer errors [15]. During the ME process, failures can occur for various reasons, such as

incorrect part orientation of the model build up, missing material flows, and detachment of the printed layers, among others [8]. For example, a strong relationship between the temperature and errors in geometry has been found in ME process printers; if the ME process printer does not include a hot chamber or a heated plate, deformation of large parts can occur due to unbalanced material cooling and subsequent shrinking [16]. However, most commercial ME process printers do not have a functionality to detect printing failures due to a lack of feedback control and monitoring tasks. Hence, it is not easy to detect errors in the geometry of the part during the ME process, users do not have references with which to evaluate whether a printed part is correct or not, and there is no information about the correct shape of the component during the layer-by-layer manufacturing process. In some cases, detecting material flow problems and optimizing process parameters can improve the quality of a fabricated part [8, 14], but all failures that can occur during the printing process ultimately cannot be prevented. Thus, to increase the utilization of ME process printers, users must monitor their printers at the same location constantly during the printing process. In this manual and traditional approach, however, watching the printing status requires much labor and time [17]. Therefore, a real-time monitoring system must be considered to reduce failed printing jobs and to reduce the unproductive time required for manual monitoring of the printing process.

Recently, to avoid failures such as the spaghetti-shape-error, users and manufacturers have installed video cameras or webcams in or in front of the printers to facilitate remote supervision [18, 19]. However, with these supervising methods, most video data watched remotely can only be interpreted by human users. In other words, the monitoring task is not reduced but only transferred from a location close to the printer to one further away. Other monitoring

Fig. 1 Growth of global additive manufacturing process (modified from [2])

approaches, including the use of laser scanning sensors [10], current sensors [20], and thermography [21], have also been investigated to detect failures. Although these condition-monitoring methods can identify conditions superior to the sensing ability of the users and printers, they require expensive sensors, the expertise of sensing knowledge, and increased integration complexity with, for instance, controls and wiring.

Recent advances in computer vision make possible various technologies, including automatic inspections, the event detection, and the reconstruction of objects or environments. In the ME process, few studies have looked into the potential use of image processing to detect failures. Vision methods can classify error cases and detect missing material flows and detachments using a blob detection approach [18]. Machines detect defects through an assessment of the printing progress and a comparison of the actual progress with the expected progress using a multi-camera system [22]. These studies demonstrated the feasibility of pattern recognition for failure detection during the ME process, but they require additional steps such as setting up rules for reasoning, comparing geometric images of the in-process and final parts, and manipulating part geometries for comparisons with the control data of the printing process. In addition, among these approaches, pattern recognition for detecting errors such as the spaghetti-shape-error has yet to be studied.

2 Image-based failure detection by machine vision for the ME process

Quality control is an essential element in a modern manufacturing system. Defect detection is required to reduce manufacturing costs and improve product quality levels during the manufacturing process. Defect detection is used to ensure product quality by detecting defects using inspection methods such as manual measurements and visual analysis [22]. Recently, the development of computer vision technology and the lack of labor have led to the introduction of image-based detection.

It is common manually to configure many of the functions that can be used to classify individual pixels to establish a detection model. The image recognition performance is improved through feature points extracted through pixel calculations based on features computed in local adjacent areas around the initial pixel. However, in order to engineer and interpret these features, a significant level of human expertise and/or actual subjects must be used to establish the imaging capabilities of the target defect [23].

In this section, existing vision algorithms are discussed, and it will be shown that it is difficult to detect defects in the stacking process with existing algorithms. The conventional computer vision approach recognizes an object by extracting

the feature points of the target object and comparing the values and positions of the feature points. Scale invariant feature transform (SIFT) method selects feature points that are easily identifiable and extracts feature vectors for local patches around the feature points. A SIFT vector is a 128-dimensional vector that divides feature points into 4×4 blocks, obtains a histogram of the gradient direction, and determines the sizes of the pixels in each block, and then connects them in a line [24]. Histogram of oriented gradients (HOG) is a vector obtained by dividing the cell into a certain size, obtaining a histogram for the pixel direction in which the gradient magnitude is greater than or equal to each cell, and then connecting these histogram columns in a row. In other words, HOG can be viewed as a histogram template in the tilt direction. HOG can be seen as a method used between template matching and histogram matching. It maintains information in units of blocks, but it is robust to local changes given its use of a histogram inside each block [25]. A Haar-like feature point is essentially a feature element that uses region and brightness differences of an image, with various types of elementary features that combine the feature elements of the object of various sizes and at various positions. A feature is then extracted [26]. The method known as Ferns is similar to SIFT in that it initially extracts feature points from movies or images and computes them for local patches around them. The method then selects two random points within the patch and uses the feature for which the difference in brightness between the two pixels is positive/negative. Compared with the Harr-like feature point, if the Harr-like feature point uses the brightness difference in the area unit, Ferns uses the difference in brightness in the pixel unit and uses only the sign, not the value [27]. Speed-up robust feature (SURF) proceeds with feature point extraction, principal direction determination, and descriptor generation similarly to the SIFT algorithm. SURF integrates images to speed up processing compared with SIFT. However, the performance of SURF does not match that of SIFT. Finally, oriented FAST and rotated binary robust independent elementary feature (BRIEF) (ORB) algorithm are in fact two algorithms. One is feature from accelerated segment test (FAST) feature point detector and the other is BRIEF descriptor. FAST is an algorithm that finds feature points in images in real time. Unlike SIFT, which has several features in one feature point, it has only one feature in one feature point. The SIFT algorithm is slow because it has a high-dimensional vector of 128 dimensions. Therefore, an alternative to this is to binarize the descriptor, and this algorithm is the BRIEF descriptor. ORB was developed to combine these two algorithms, i.e., the FAST feature point detector and the BRIEF descriptor [28].

As shown in Fig. 2, feature points are created around the spaghetti shape. However, it is difficult to match a spaghetti-shape error of the same shape because the size and direction of the feature points of the figure above and the following figure

Fig. 2 Image feature point extraction. **a** Origin images. **b** Scale invariant feature transform (SIFT). **c** Speeded-up robust features (SURF). **d** Oriented FAST and rotated BRIEF (ORB)

differ. It means all spaghetti-shaped errors cannot be identified as vectors in the same direction. Therefore, for the spaghetti-shape error features targeted in this paper, typical image feature extraction methods cannot provide a model for distinguishing between normal deposition forms and spaghetti-shaped errors due to the atypicality of the target geometry.

3 Method: Image-based failure detection by CNN for the ME process

In recent years, convolutional neural network (CNN) has led to tremendous improvements in image processing applications. Image recognition and classification are now possible for neural network learning without the need for the calculations used in traditional image recognition and classification algorithms.

With the development of composite multiplying neural networks, there is an example in the manufacturing field that extracts defects using a CNN in the form of an image of a fabrication process on a laser powder bed. There are several studies that use sound data with a CNN to measure defects in gears to determine if they are defective. As another example, vibration data from bearings is used with a CNN [29–31]. CNNs have been applied to various manufacturing fields in addition to image processing.

As mentioned above, conventional image processing extracts feature points using SIFT, HOG, Harr-like feature points, and Ferns and classifies and recognizes images using SVM, a classifier. Recently, with the development of deep

learning algorithms, images are often recognized and classified using a composite product neural network. Such a network automatically extracts the feature points of the convolutional product filter through training. This section briefly describes the structural features and roles of the CNN and describes the CNN structure used in this study.

In this study, the ME process was monitored with the suggested CNN-based failure detection method. A detailed conceptual presentation of the method is shown in Fig. 3, and the image dataset come from [32]. This method based on a CNN model is trained with acquired images and the trained model detects the stacking process with a webcam to determine if the process is feasible and if it does not fail due to a spaghetti-shape error.

3.1 Input layer

All CNNs operate in an input layer an input volume of size width $\times$ height $\times$ depth [33]. The input layer of the Visual Geometry Group Network (VGGNet) CNN was originally designed to operate on color images from the ImageNet dataset and is of size 224 pixels $\times$ 224 pixels $\times$ 3 pixels, where the depth spans the three color channels (red, green, and blue) [33, 34]. When applying transfer learning to a pre-trained CNN, the CNN architecture, including the size of the input layer, must remain unchanged. This implementation is performed by putting a fixed input value of depth 3, and is mainly seen in supervised learning. Its size was chosen based on the authors' experience with the 128 pixels $\times$ 128 pixels patch. In particular, since only the input image is used to recognize the

Fig. 3 Concept diagram of CNN-based failure detection method

spaghetti error, the image size was determined experimentally by finding the depth and hyperparameters of the CNN.

3.2 Hidden layers

Once the data are stored in the input layer, mathematical operations are applied to the data in a sequence of “hidden layers,” so named because of the operations learned by the CNN during training. The VGGNet (or VGG-19) CNN has a total depth of 19 layers, 19 of which are considered hidden for the purposes of this subsection. As shown in Fig. 4, the

data stored in the input layer are first operated on by a convolution (Conv) layer. The convolution operations extract features using filters via the summation of the element-wise multiplication of two matrices, as discussed in [31, 33]. Critically, these filters’ parameters are not chosen by a human, rather they are learned by the CNN during training. For this reason, one may consider the Conv layer to be an optimized filter bank. Then, the filters learned by CNNs for the first Conv layer are typically highly similar regardless of the specific classification application. The filters used in the first Conv layers are of size 3 pixels $\times$ 3 pixels $\times$ 64 pixels. The size of

Fig. 4 Result of the image through the convolutional layer and pooling layer and structure of CNN

the filter specifies the area of the input data over which the convolution is performed, while the “stride” of the filter specifies the spatial distance between the centers of the convolutions. In other words, for a stride of one, the convolution area moves one pixel in a given direction between operations. In the first Conv layer of the model, the stride is 1 [33]; i.e., the convolution area moves four pixels in a given direction between operations. A larger stride reduces the dimensionality of the Conv layer, but reduces the spatial resolution at which features are extracted. The convolution operations result in a data volume with a depth equal to the number of filters and a width (W) and height given by Eq. (1). The volume of the first Conv layer in our model is 128 pixels $\times$ 128 pixels $\times$ 64 pixels. Because these filters operate through the depth of the input data volume, they are often referred to as kernels.

$$W_{i+1} = (W_i - F + 2P) / S + 1 \quad (1)$$

where W_{i+1} is the size of the output layer width (or height), W_i is the size of the input layer width (or height), F is the spatial width (or height) of the kernel, S is the stride of the kernel in the width (or height) direction, and P is the number of padding pixels explicitly used during the convolutions of the input data. Note that, in the first Conv layer of the model, P is set to zero which is zero padding. Also note that the output size of a pooling operation can also be computed using this equation, while hyperbolic tangent functions are often applied to the kernel outputs and others have determined that far superior training speeds can be obtained through the use of rectified linear units (ReLU) which are defined in (2). Note that the ReLU layer does not alter the size of the data volume; i.e., the output of the first ReLU layer in the model is of size 128 pixels $\times$ 128 pixels $\times$ 64 pixels.

$$\text{ReLU}(x) = \max(0, x) \quad (2)$$

where ReLU is the output of the ReLU operation and is the output of the kernel, i.e., the response of the convolution. The dimensionality of a CNN would increase unsustainably through the depth of the CNN without down-sampling (pooling) the responses from the lower layers. There are several methods by which down-sampling may be achieved, but all of them operate spatially; i.e., dimensionality is reduced along the width and height of the data volume without affecting the depth of the volume. In the presented model, down-sampling is accomplished via a max pooling layer [33]. Max pooling operates by only passing the maximum response within a given window on to the next layer. For example, the window size of the first max pooling layer of our model is 2 pixels $\times$ 2 pixels; therefore, only the maximum of the responses within a window is passed on to the next layer. Interestingly, while pooling windows are traditionally non-

overlapping, all of the max pooling layers in VGGNet utilize windows of size 2 pixels $\times$ 2 pixels and a stride of two and therefore operate on overlapping regions. In addition to reducing the dimensionality of the CNN, pooling operations have also been shown to mitigate overfitting [33]. Following the input layer, Conv layer, ReLU layer, and max pooling layer, the data volume is once again convolved with a set of kernels and the responses are stored in a second Conv layer. Notably, while the first Conv layer extracts low-level features such as blobs, edges, and lines, the second Conv layer extracts higher level features. For example, the second Conv layer’s analysis of the data volume may allow for the detection of intersections of vertical and horizontal lines, e.g., corners. This process is repeated through the depth of the model for a total of five Conv layers with each Conv layer extracting higher and higher level features. After the final Conv layer and associated ReLU layer, a fully connected (FC) layer is constructed of size 1 pixel $\times$ 1 pixel $\times$ 2048 pixels. A FC layer is equivalent to a Conv layer in which each kernel has a spatial size equal to that of the input data volume. Therefore, each convolution operation produces a single response. Finally, softmax is used as the classifier of the output layer.

3.3 Training

The previous three subsections describe the architecture of the model and the operations performed on the input data during classification. This subsection is intended to provide a brief overview of the training process for the original VGGNet CNN as well as the application of transfer learning used to convert it to model capable of classifying spaghetti-shape errors. Only the training parameters used by the authors for transfer learning are provided below; refer to [33] for a more complete discussion regarding the training of the VGGNet CNN. CNN training operates using a process known as backpropagation [35]. Initially, all of the weights of all of the kernels throughout the depth of a CNN are randomized. While not previously discussed explicitly, weights are simply the element-wise values composing a filter or kernel. During the “forward pass” stage of backpropagation, the training data are passed through the depth of the CNN; because the kernel weights are initially randomized, the classification performance will initially be extremely poor. Since the training data are labeled by the human with ground-truth classifications, the performance of the untrained CNN can be quantified. The first 1 of the vector is the number of pixels, and the 1×2 vector is the size of the vector [0,1]. The value of the set can be seen as a kind of label for classifying as 0 (failure) or 1 (success) when a vector of size $1 \times 1 \times 2$ passes the softmax activation function. The result value of softmax activation function is a decimal value between 0 and 1, which is determined as 1 when 0.5 or more, and 0 when 0.5 or less, which is often used for classification. Therefore, the softmax output is nominal 0 or 1,

Table 1 Specification of hardware

Specification	Value
Operating system	Microsoft® Windows® 10
System RAM	Samsung® 8 GB (DDR4)
Processor type (CPU)	Intel® core i5-7500
Processor speed (CPU)	3.4 GHz
Graphics card (GPU)	NVIDIA® GeForce GTX 1050
On-board RAM(GPU)	2 GB

but actually it means a probability value. The error between this softmax output value and the desired output can be defined by various energy functions. As the goal is to reduce the classification error, it is desirable to adjust the weights in the direction opposite to the gradient of the loss function. The calculation of the gradient is considered the “backward pass” stage of the backpropagation process. Both VGGNet utilize a method known as stochastic gradient descent (SGD) to calculate the weight adjustment. In traditional GD, the loss function is defined for the entirety of the training dataset. While this approach can produce high classification accuracies, it is too computationally expensive to be used for backpropagation through the depth of a CNN. For this reason, CNN utilizes SGD which defines the loss function only over a subset of the training dataset [36]. However, in this experiment, the loss function is obtained using Adam [37]. Adam is an algorithm that combines the existing root mean square propagation (RMSProp) and momentum methods. Similar to the momentum method, this method stores the exponential mean of the slopes calculated up to a certain point, and similarly to RMSProp, it stores the exponential mean of the squares of the slopes [37, 38].

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) \nabla_{\theta} J(\theta) \quad (3)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) (\nabla_{\theta} J(\theta))^2 \quad (4)$$

However, in Adam, m and v are initially initialized to 0, so at the beginning of the train, m_t , v_t is determined to be biased close to 0 and unbiased. By unfolding the expressions of m_t and v_t in the form of $\sum$ and putting expectation on both sides, we can obtain the unbiased expectation through the following correction. With these calibrated expectations, we compute $\hat{v}_t$ where $\hat{m}_t$ and G_t are placed in the gradient.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}, \quad \theta = \theta - \frac{\varepsilon}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t \quad (5)$$

Each subset of the training dataset is known as a “mini-batch” and is randomly (hence the “stochastic” nomenclature) delineated at runtime. Each time convergence is achieved for the set of mini-batches covering the entire dataset, the entire backpropagation process is repeated, and each repetition is referred to as an “epoch.” During the training, whole weights are initialized randomly and backpropagation is applied through the depth of the CNN. For training of the final layer, an unscheduled learning rate of 0.001 was used and a total of 50 epochs were executed. Finally, it should be noted that, during the described training process, only the kernel weights are learned. In other words, the architecture of the CNN remains static and is not automatically optimized. During the CNN design process, a human programmer manually modifies the CNN architecture (hyperparameters) in order to achieve improved validation performance. In the next session, we show how to extract the spaghetti-shape error of a laminated 3D printer through experiments, with optimization of the CNN structure as well.

The global feature point of this image is extracted by iteratively conducting multiplication and pooling layers. Figure 4 shows the result of a spaghetti-shape error image passing through the composite product layer. First, we found the contours and extracted meaningful feature points based on them.

Fig. 5 Number of images at the each data set

Fig. 6 Examples of data augmentation

Fig. 7 Example images used at success detection test

Case (a): completed process and Success #1

Case (b): completed process and Success #2

Case (c): completed process and Success #3

Fig. 8 Example images used at failure detection test

Finally, we confirmed that the most meaningful values were shown numerically.

Thus far, we have investigated the basic structure used in a CNN and the functions used. This study is based on the basic structure of VGGNet among various CNN models. Therefore, following the boxes at the bottom of Fig. 4, Conv1_1, Conv1_2, Pooling1, Conv2_1, Conv2_2, Pooling2, Conv3_1, Conv4_1, Conv4_2, Pooling4, Conv5_2, Pooling5, Conv6_2, and 2 dense layers were used in structure of the CNN model; we used a total of 20 layers, including twelve convolutional layers, six pooling layers, and two dense layers, as noted above, where depth means the number of filters, and the figure in parenthesis next to the filter is the size of the filter. Finally, classification was conducted by using the

softmax function in the output layer. Then, this model is optimized by Adam.

4 Experiment setup: Data acquisition and data augmentation

First, the experimental environment of this research is as Table 1. From an economic point of view, the configuration of hardware and software was selected as specifications of general desktop computer instead of high-performance workstation.

All learning and data processing processes were performed with python language. Then, Keras library was used for

Fig. 9 Result of each failure detection test executed at example images

Predicted failure	47 True Negatives	1 False Positives
Predicted success	3 False Positives	49 True Positives
	Actual failure	Actual success

Fig. 10 Result of failure test of process images

learning and TensorFlow was used as backend. TensorFlow is a library for deep learning provided by Google. The dataset of the experiments in this paper was compiled while processing images and was collected in the form of images. Some datasets were actually obtained, but some were obtained by searching. However, there was a limit when attempting to collect the datasets because the results obtained by users of the ME process were not recorded as data depending on the state. With learning only from the collected data, 156 learning data instances were divided into 39 validation instances. For classifications using the CNN models, the data given for learning the CNN models are distinguished by training sets and validation sets. In learning, training sets divide learning for model learning to ensure that the correctness and overfitting of the learned model and the loss of the model converge well. Validation sets have fewer data than training sets for the

purpose of evaluating the models they learn. In this study, as shown in Fig. 5, the total data was divided by about 4:1, and success and failure data were divided according to this number and then used as learning data. The accuracy of the learning was high, but overfitting occurred in the test evaluation, and it was not known whether the model results were properly validated. There was a limit because there were too few data instances to run the CNN with the collected dataset. Therefore, in this experiment, we were forced to implement data augmentation on the image set. Several methods that can be used to expand image data have been presented in the literature. Cropping, shift, flip, brightness, saturation, channel shift, etc. were used as a method of data augmentation, and each change was randomly numbered to increase dataset by 100 times. No extended image lost the feature points of the original image. The following is an example of an image with data augmentation, specifically in Fig. 6. When data augmentation is performed, numerous shift and rotation methods are used. As the image does not move out of the frame according to the movement, the number of movements is minimized as much as possible. A total of 15,600 images were learned by data augmentation, with each image learned 100 times, and 3900 images were used as the validation data. The total number of images used in each dataset is shown in Fig. 5.

5 Result and discussion

In this experiment, the most basic structure of a CNN is composed of a convolutional layer and a pooling layer, as well as a fully connected layer, such as VGGNet, using the ReLU activation function. Through these structures, the most suitable size and the most suitable composite product structure of the dataset images are obtained through experiments. Although this represents the simplest structure, the classification shows

Fig. 11 Fabrication process time order of success/failure probability graph

good accuracy in two classes; the success or failure of which is distinguished as in this experiment. Also, the model structure of this study consumes about 5 min or less to train the model.

Confusion matrices are a metric commonly used to evaluate deep learning algorithms. Fundamentally, a confusion matrix compares a deep learning algorithm's classifications to the ground-truth classifications. In all implementations, the data used to generate a confusion matrix must be separated from the data used to train the deep learning algorithm. Traditionally, the entire available dataset is divided into three subsets known as training, validation, and testing datasets. During the training process, the deep learning model is fit to the training data set. The performance of the model can then be evaluated using the validation dataset and the human programmer may decide to modify the design of the model based on these results. Once the design of the deep learning algorithm and any accompanying methodology is complete, the true performance can be estimated using the testing dataset which also serves as a final check that the model has not been over-fitted to the training data.

Through our structure, we tested whether we can extract an image with an actual spaghetti-shape error. We set up a special test image set for this purpose. As shown in Figs. 7, 8, and 9, respectively, the test images were checked to determine if a model with obvious spawn or complete spawning errors would be extracted and the test images came from [39–43]. We tested 16 untrained objects which are eight failure cases and eight success cases. One hundred of the different images obtained by capturing images of the process of eight success and failure objects respectively were selected. One hundred untrained images were used for the test, and fifty of them are failures and fifty are successes.

The graph in Fig. 9 is a probability graph that determines success or failure for images (a) and (d). And, Figs. 7 and 8 show images of success and failure cases for the same object. When the spaghetti-shape error occurs for the same object or for the first object, it is detected. In this case, 100 images captured from actual ME processes were fitted as shown in the confusion matrix in Fig. 10. Overall, the failed image prediction rate was 94%, and the successful image prediction rate was 98%. Through this result, the accuracy of the recognition rate of success and failure targets was averaged to develop a model with a total accuracy of 96%. Because the failed images used as the subjects in this experiment trained the exact spaghetti-shape error and a twisted image, even if the model is successful, the accuracy predicted by the success is significantly lower than when the image is twisted.

Figure 11 shows a graph of the real-time detection outcome of the ME process. The process above fabricated a statue on horseback. The process lasted a total of 26 min, and it was found that the process failed to detect the fabrication process in 22 min.

6 Conclusions

In this study, failures known as “spaghetti-shape errors” which occurred during the ME process were detected by a CNN-based failure detection method based on captured image data from a webcam. The CNN algorithm demonstrated approximately 96% accuracy when used to classify images. This has the effect of catching qualitative spaghetti-shape errors rather than catching quantitative spaghetti-shape errors. The CNN model-based failure detection method used in this study was used to detect errors in real time using a webcam. The results of this study will allow users to detect when a spaghetti-shape error arises, to catch failures, and they will make the total process time shorter. Moreover, this method can inform the user through the image when such a failure occurs during the ME process. With this proposed failure detection method, the ME process can be evaluated and analyzed numerically in real time. These evaluation and analysis techniques are expected to be used as basic research materials for an IoT-based smart factory, especially for SMEs.

Funding This research was supported by the National Research Foundation of Korea (NRF) grants funded by the MSIT (2018R1A4A1059976) and Korea Institute for Advancement of Technology (KIAT) grants funded by the Korea Government(MOTIE) (P0008691, HRD Program for Industrial Innovation).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Cohen D, Sargeant M, Somers K (2014) 3-D printing takes shape. *McKinsey Quarterly* 1:1–6
2. Wohlers Associates Inc., Wohlers Report 2017 (2017)
3. Kim D, Kim T, Wang X, Kim M, Quan Y, Oh J, Yang I (2018) Smart machining process using machine learning: a review and perspective on machining industry. *Int J Precis Eng Manuf-Green Technol* 5(4):555–568
4. Piller FT, Weller C, Kleer R (2015) Business models with additive manufacturing—opportunities and challenges from the perspective of economics and management. *Lecture Notes in Production Engineering*:39–48
5. Weller C, Kleer R, Piller FT (2015) Economic implications of 3D printing: market structure models in light of additive manufacturing revisited. *Int J Prod Econ* 164:43–56

6. D'Aveni R (2018) The pan-industrial revolution: how new manufacturing titans will transform the world, Houghton Mifflin Harcourt
7. Lee CS, Kim SG, Kim HJ, Ahn SH (2007) Measurement of anisotropic compressive strength of rapid prototyping parts. *J Mater Process Technol* 187-188:627–630
8. Anitha R, Arunachalam S, Radhakrishnan P (2001) Critical parameters influencing the quality of prototypes in fused deposition modelling. *J Mater Process Technol* 118(1–3):385–388
9. <https://www.3dhubs.com/knowledge-base/industrial-fdm-vs-desktop-fdm> (accessed 2019.12.12)
10. Faes M, Ferraris E, Moens D (2016) Influence of inter-layer cooling time on the quasi-static properties of ABS components produced via fused deposition modelling. *Procedia CIRP* 42:748–753
11. Wohlers Associates Inc., Wohlers Report 2016 (2016)
12. Song R, Telenko C (2017) Material and energy loss due to human and machine error in commercial FDM printers. *J Clean Prod* 148: 895–904
13. Wittbrodt BT, Glover A, Laureto J, Anzalone G, Oppliger D, Irwin J, Pearce JM (2013) Life-cycle economic analysis of distributed manufacturing with open-source 3-D printers. *Mechatronics* 23(6):713–726
14. Rao PK, Liu JP, Roberson D, Kong ZJ, Williams C (2015) Online real-time quality monitoring in additive manufacturing processes using heterogeneous sensors. *J Manuf Sci Eng* 137(6):061007
15. Song R, Telenko C (2016) Material waste of commercial FDM printers under realistic conditions. Paper presented at the Solid Freeform Fabrication 2016: Proceedings of the 26th Annual International Solid Freeform Fabrication 2016: Proceedings of the 27th Annual International Solid Freeform Fabrication Symposium—An Additive Manufacturing Conference
16. Baumann F, Schön M, Eichhoff J, Roller D (2016) Concept development of a sensor array for 3D printer. *Procedia CIRP* 51:24–31
17. Kim HJ, Jung WK, Choi IG, Ahn SH (2019) A low-cost vision-based monitoring of computer numerical control (CNC) machine tools for small and medium-sized enterprises (SMEs). *Sensors* 19(20):4506
18. Baumann F, Roller D (2016) Vision based error detection for 3D printing processes. *MATEC Web of Conferences* 59:06003
19. Ceruti A, Liverani A, Bombardi T (2017) Augmented vision and interactive monitoring in 3D printing process. *Int J Interact Des Manuf* 11:385–395
20. Tlegenov Y, Lu W, Hong G (2019) A dynamic model for current-based nozzle condition monitoring in fused deposition modelling. *Progress in Addit Manuf* 4:211–223
21. Krauss H, Zeugner T, Zaeh M (2014) Layerwise monitoring of the selective laser melting process by thermography. *Phys Procedia* 56: 64–71
22. Straub J (2015) Initial work on the characterization of additive manufacturing (3D printing) using software image analysis. *Machines* 3:55–71
23. Zhao D, Li S (2005) A 3D image processing method for manufacturing process automation. *Comput Ind* 56(8–9):975–985
24. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. *Int J Comput Vis* 60:91–110
25. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 1, 886–893
26. Lienhart R, Maydt J (2002) An extended set of Haar-like features for rapid object detection. In: *Proceedings. International Conference on Image Processing*, 1, pp 900–903
27. Ozuyisal M, Calonder M, Lepetit V, Fua P (2009) Fast keypoint recognition using random ferns. *IEEE Trans Pattern Anal Mach Intell* 32(3):448–461
28. Rublee, E., Rabaud, V., Konolige, K., Bradski, G. (2011). ORB: an efficient alternative to SIFT or SURF. 2011 International Conference on Computer Vision, 2564–2571
29. Ferguson MK, Ronay A, Lee Y-TT, Law KH (2018) Detection and segmentation of manufacturing defects with convolutional neural networks and transfer learning. *Smart and Sustainable Manufacturing Systems* 2(1):137–164
30. Wang T, Chen Y, Qiao M, Snoussi H (2018) A fast and robust convolutional neural network-based defect detection model in product quality control. *Int J Adv Manuf Technol* 94:3465–3471
31. 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. *Addit Manuf* 24:273–286
32. https://www.google.com/search?tbm=isch&sxsrf=ACYBGNT3WUY0F0L63gjyBVunfg4uBLcmJw%3A1578027226099&source=hp&biw=1357&bih=800&ei=2sgOXpbEA9KmmAWb44KgBQ&q=3d+printer+failure&oq=3d+&gs_l=img.3.0.35i39l2j0l8.142.521..1249...1.0..0.106.309.1j2.....0.....1..gws-wiz-img.6SDd6DEpkQA (accessed 2019.10.21)
33. Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Proceeding advances in neural information processing systems* 25 (NIPS 2012)
34. Zhang X, Zou J, He K, Sun J (2015) Accelerating very deep convolutional networks for classification and detection. *IEEE Trans Pattern Anal Mach Intell* 38(10):1943–1955
35. Zhang Z (2016) Derivation of backpropagation in convolutional neural network (CNN). University of Tennessee, Knoxville, TN
36. Bottou L (2010) Large-scale machine learning with stochastic gradient descent. *Proceedings of COMPSTAT 2010*:177–186
37. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. *arXiv*:1412.6980
38. Dauphin, Y., Vries, H. d., Bengio, Y. (2015). Equilibrated adaptive learning rates for non-convex optimization. *arXiv*:1502.04390
39. <https://www.youtube.com/watch?v=TCNZGXDDfUc> (accessed 2019.10.21)
40. <https://www.youtube.com/watch?v=FqQAjkZObEY> (accessed 2019.10.21)
41. https://www.youtube.com/watch?v=uzKMV_O42SY (accessed 2019.10.21)
42. <https://www.youtube.com/watch?v=hTzFrAPbOes> (accessed 2019.10.21)
43. <https://www.youtube.com/watch?v=cgAhF1FuUJA> (accessed 2019.10.21)

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.