A Technical Study for Defect Detection in Additive Manufacturing (3D Printing) using a Convolutional Neural Network

Noah Hamilton
University of South Florida, hamiltonn@usf.edu

***Abstract* – Additive Manufacturing is a booming industry and is being used to create complex models and parts. They have 3D printers and other machines like CNC which are designed to create usable objects by fusing together plastic with heat. Additive Manufacturing takes a lot of time to learn and perfect. It is very costly to continuously test through trial and error. Previous research has investigated three parts of the process for optimization including initial Geometric Design, Process Parameter Configuration, and Anomaly Detection. This paper initially focused on the process parameter configuration but shifted to defect detection as the research progressed. This is usually a very time-consuming and expensive process, and my goal is to reduce error detection time significantly. I created my own dataset, tabulated into a database, and then created a neural network to classify defects in a 3D Printed model into three categories: stringing, clean, and failed prints. Finally, I compared the neural network model against test data that I separated with a train-test split of the data set I created. The goal being to optimize defect detection using an automated model.**

Key Terms – 3D Printing, Machine Learning, Additive Manufacturing, Neural Network

Introduction and Motivation

How can Machine Learning be used in Additive Manufacturing to reduce waste and automate the process parameter configuration process? This is one of the most time-consuming steps in the manufacturing process. Additive Manufacturing relies on a geometric computer-generated design, which is then sliced into code with parameters that a printer then executes. AI in manufacturing was one of the top technology trends for the IEEE Computer Society in 2021 with a goal in increasing reliability. This technical study on a small-scale 3D printer could then be applied to larger scale operations such as construction, automotive, and aerospace industries. This is a prime example of a Cyber-Physical system as the computer generates a model that is then realized with an additive manufacturing method such as a 3D printer.

Initially, this project was designed to optimize printing process parameter, but after some trouble identifying minute differences between settings, I noticed a more pressing problem that was the identification of a bad print. This led to the shift of focus to a defect detection and classification problem. The new objective was to create a Convolutional Neural Network (CNN) to classify images of prints into three categories: Clean/Processed, Stringing, and Failed Prints. Each of these categories can be seen in Figure 1 below. As a written description, the criteria used for identifying the category is as follows: stringing is identified as filament between towers, failed is identified as not completing both towers, and clean is both towers completed without filament between.

Figure 1: Examples of prints by category (stringing – left, failed – middle, clean – right) Photo Credit: Noah Hamilton

The new objective has the goal of creating the best classifier with highest accuracy. Creating defect detection models is a goal of many manufacturing companies and can help in many ways to save money and resources.

Background of Current Work

In the paper, “Machine Learning for Advanced Additive Manufacturing” scientists and engineers explored the many ways that typical algorithms and Machine Learning models could be applied to additive manufacturing processes. They expanded on all three phases from initial design, parameter configuration, and lastly anomaly detection after completion. The article nicely compares generic methods to machine learning models emphasizing the benefits of using a more cost-efficient method for training and testing in manufacturing. This technical study focuses on the process parameter configurations as this is the most wasteful and time-consuming aspect. The previous research compared a generic algorithm with a small range of input values and combinations to produce and optimal print. However, this takes a wide range of print failures to identify the correct solution. With a larger scale, ML can be used to identify interdependencies between process parameters to reduce cost greatly. The research also references the use of a CNN to analyze print speed, extrusion multiplier, and fan speed based on images captured of the prints. I used a similar

technique with different parameters choices. To add on to this current work, a convolutional neural network was applied to a set of personally created 3D models to classify them based on features after the fact. In the next section, the technical details of the approach, techniques used, and results are explained.

TECHNICAL APPROACH AND RESULTS

To begin the technical approach section, it is important to know what a convolutional neural network is and how they work.

A Convolutional Neural network is a typical neural network that contains an input layer, hidden layer, and output layer of neurons connected to each other with the addition of a special convolutional layer. This convolutional layer uses filters that extract features from images while maintaining positional data inside of a picture. These networks are very good at picking up patterns including lines, gradients, circles, or even faces. A CNN mimics the behavior of the human eye's ability to identify patterns within images. These neural networks are composed of layers that are connected to each other. Many different types of layers exist, but some of the more common layers are Convolution, Pooling, Dropout, ReLU Activation, and SoftMax. Figure 2 below shows an example of how these layers can be connected to create a CNN.

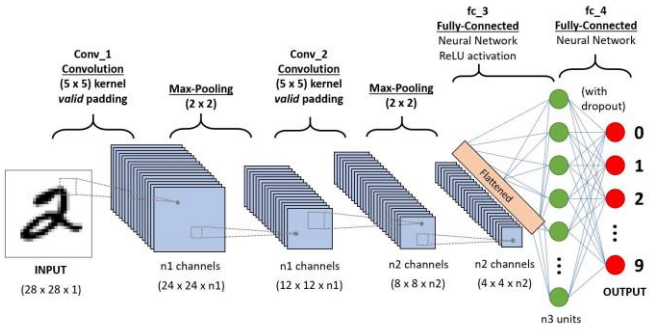

Figure 2: Example of CNN using an input image from the mnist dataset. Credit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

The network requires images or matrices to work with to analyze. For an image, it is typical to assign a weight based on the RGB or grayscale value to each pixel and use those values to compute weights between layers in the network. Each of the layers mentioned previously has a different purpose in manipulating the matrix to take an input image and assign a class to it. A convolution layer multiplies each number by each other number and adds them together to get a summarized version of a larger image. Next, a pooling layer takes the highest value from each region to make a new matrix with only those values. The pooling layer helps to reduce the spatial dimension of the model and increases performance as neural networks are computationally heavy. A dropout layer is then applied to force the neural network to disable some of the networks during the learning phase to increase accuracy and decrease overfitting. The last two layers (ReLU and Softmax) are used to normalize the values obtained and then turn the real values for weights into probabilities for classifications. It is important to note that the convolution and pooling layers are often used more than once to identify and target multiple features within a single input image.

After explaining the basis of CNNs and layers, the implementation for defect detection can be seen clearly. The first step in creating a model for classification is acquiring data. Existing datasets did not have the characteristics for this test especially in the 3D printing arena or additive manufacturing. For this project, an open-source model was used from Thingiverse, prepared using the Cura Slicer Software and printed on a Crealty Ender3 3D printer using CCTREE 1.75mm PLA filament. Each print took between 30-45 minutes. Python3 using TensorFlow with Keras was used to create the Convolutional Neural Network. After initially planning on tuning printing parameters, a small table was created with 8 separate models and 5 of each model was printed. Table 1 shows the settings changed between prints and Figure 3 shows all the models together.

Table 1: Distribution of eight models with printing parameters specified.

Model Number	Max Resolution (mm)	Layer height (mm)	Print Speed (mm/s)	Retraction (mm at mm/s)
1	0.5	0.12	50	5 at 50
2	0.5	0.12	50	None
3	0.5	0.12	30	5 at 50
4	0.5	0.12	30	None
5	0.5	0.2	50	5 at 50
6	0.5	0.2	50	None
7	0.5	0.2	30	5 at 50
8	0.5	0.2	30	None

Figure 3: All the 3D printed models sorted by class from left to right (clean (before post-processing), stringing, and failed prints. *Note: There were 12 clean, 14 stringing, and 14 failed prints.*

After creating all 40 models, each 3D model was photographed against a carpet background and sorted into three folders based on my classification. To be small enough for the model to process, the image resolution must be reduced from a High-Definition Photograph. 50x50 pixels was chosen initially as recommended online. Figure 4 shows an example of what the image looks like to the CNN.

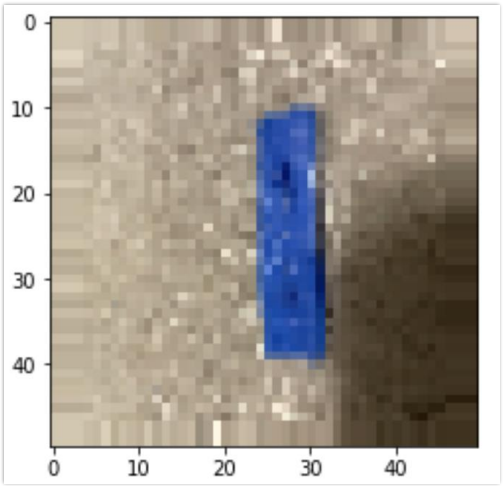

Figure 4: Failed print compressed to 50x50 pixels for CNN.

One of the initial concerns with this project was the small data set available. A technique known as Data Augmentation was used that creates more images for training from the initial 40 models. This process is available through the Keras library and rescales, zooms, flips, and shifts the images of the model to create more data for training. This data augmentation also sets aside a specified amount of the images for validation, specifically 20% validation split was used.

After creation of the dataset, it is time to create the model itself. In this implementation, three convolutional layers each paired with a MaxPooling Layer were used to identify features and reduce the spatial dimension of the images. Then a Flatten and a Dense Layer were added as hidden layers to associate the features with the inputs and outputs. Lastly, the model is compressed into 3 neurons and a Softmax function is used to transform the output into a probability distribution. Figure 5 shows the implementation of this in the Python Code.

Next step in creating a Convolutional Neural Network is the training phase. The purpose of training is to assign weights to each of the branches connecting the neurons within the layers and adjust the weight based on the accuracy of each classification of the output. The model attempts to assign weights each epoch and then edits those weights based on the accuracy and loss and then evaluates of the validation set. This is the most time-intensive step in creating a CNN. It took roughly 15 minutes each time just to run the training code.

```
# Building the model
model = Sequential()
# 3 convolutional layers
model.add(Conv2D(32, (3, 3), input_shape=(50, 50, 3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# 2 hidden layers
model.add(Flatten())
model.add(Dense(128))
model.add(Activation("relu"))

model.add(Dense(128))
model.add(Activation("relu"))

# The output layer with 3 neurons, for 3 classes
model.add(Dense(3))
model.add(Activation("softmax"))
```

Figure 5: Python Code for the Convolutional Neural Network Layers.

To evaluate the model, two parameters are commonly used called Accuracy and Loss. Accuracy is defined as the number of errors the model made on the data or the percent of data classified correctly. Loss is known as the average distance between the true value of the problem and the values predicted by the model. Loss is often subjective and depends on the problem. For example, in this problem, the model classifies images as numbers between 0,1,2, so loss should be very low, but some projects classify based on 0-1000 and in that case higher values of loss can be acceptable. In interpreting accuracy and loss, the ideal case is High Accuracy and loss, this means that there are small errors on few pieces of data. However, this was not the case in this project, we had High Accuracy and High loss as seen in Figure 6, which means large errors on pieces of data. This is most likely due to a relatively small dataset.

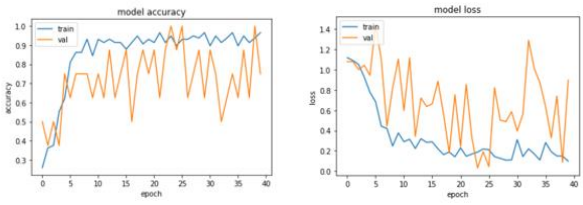

Figure 6: Accuracy and Loss plots for initial model created. There was relatively high accuracy and low loss for training data; however the validation parameters did not have enough data for the model to consistently classify images.

From this initial model, an attempted correction was made by increasing all of the compression photos to 250 by 250 pixels to provide greater resolution. The number epochs or

training cycles was also increase from 40 to 100. However, as shown in Figure 7 below, this led to even greater loss and less accuracy due to overfitting.

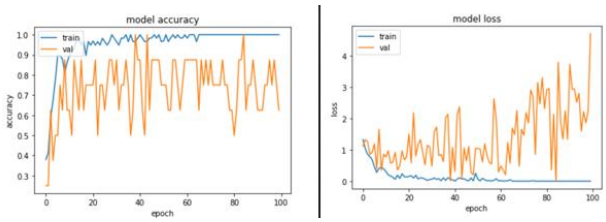

Figure 7: Loss and Accuracy plots for attempted correction.

The most useful part of a Convoultional Neural Network that classifies images is the ability to predict models and images that are not contained within the training data. Three more models were made and a prediction algorithm was run using the initial model as it had a higher accuracy. The model correctly identified failed prints with a 93% confidence, but incorrectly idetified models in the stringing and clean categories. Possible causes of this are the small dataset and the low resolution of the image and the details are too fine for the model to interpret.

From current literature, some ways in the future to improve the model are using more data augmentaion, more aggressive dropout, using more convolutional blocks, and corss validation.

Overall, a convolutional neural network was created to analyze defects within 3D printed models that was approximately 70% despite a small dataset. This implementation would be a great tool to implement into a live machine that could provide feedback constantly and potentially stop a failing print or adjust settings to stop strining early. This would drastically help improve efficiency within an additive manufacturing process and reduce cost. Furthermore, models similar to this can be used for facial recognition, optical character resolution, and IOT device networks. Despite a low accuracy and incorrect predictions, the process and proof of concept for this convolutional neural network can set a precedent for future applications in the Cyber Physical Systems for Defect Detections models.

REFERENCES/RESOURCES

[1] Jin, Z., Zhang, Z., Demir, K. and Gu, G., 2020. Machine Learning for Advanced Additive Manufacturing. Matter, 3(5), pp.1541-1556.

https://towardsdatascience.com/all-the-steps-to-build-your-first-image-classifier-with-code-cf244b015799

https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

https://colab.research.google.com/notebooks/io.ipynb#scrollTo=p2E4EKhCWEC5

https://courses.analyticsvidhya.com/courses/convolutional-neural-networks-cnn-from-scratch?utm_source=blog&utm_medium=build_image_classification_model_using_python_and_keras

https://datascience.stackexchange.com/questions/42599/what-is-the-relationship-between-the-accuracy-and-the-loss-in-deep-learning#:~:text=Accuracy%20can%20be%20seen%20as,on%20a%20lot%20of%20data

Retraction Test Design: https://www.thingiverse.com/thing:2563909

LINKS TO CODE AND DATASET

https://colab.research.google.com/drive/15lnxYv9uKvHlH52uKPbkO8n2y4V6rD20?usp=sharing

https://drive.google.com/drive/folders/1BAE5OIu7aLTQ2nxsrvR7k7NerTCNiYBh?usp=sharin