3D Print Defect Classifier

INFO6147 - Deep Learning with Pytorch

Leonardo Arteaga

Project Report

Fanshawe College
April 4, 2024

1 Introduction

In the field of 3D printing, identifying defects during the printing process is a critical challenge that directly impacts the quality and reliability of the printed objects. These defects can range from minor surface irregularities to significant structural failures, affecting both the aesthetic and functional aspects of the printed items.

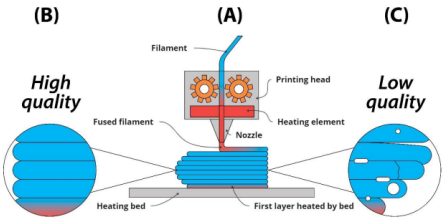

Figure 1: Scheme of Fused Filament Fabrication [1]

Addressing this issue, the focus of the project is to leverage the capabilities of deep learning to develop a classifier capable of distinguishing between defective and non-defective prints. This model aims to identify potential defects during the printing process. The significance of this work is merely academic and to explore deep learning techniques within the framework of fuse filament fabrication.

1.1 Objectives

This project aims to hit the following goals:

- Develop a classification model employing PyTorch, aimed at categorizing images of in-process 3D prints as either 'good' or 'defective'.
- Optimize model performance through hyperparameter optimization, utilizing the grid search technique for systematic exploration of parameters.
- Conduct a preliminary comparison of the model's effectiveness with two different base models: EfficientNet [2] and ResNet [3], to observe their respective impacts on model performance.

2 Methodology

2.1 Data Collection & Preparation

The foundation of the study was based on a dataset of '34 classes of 3d printing errors in 34 different shapes' which comprised images labeled according to the presence of printing errors. [4]

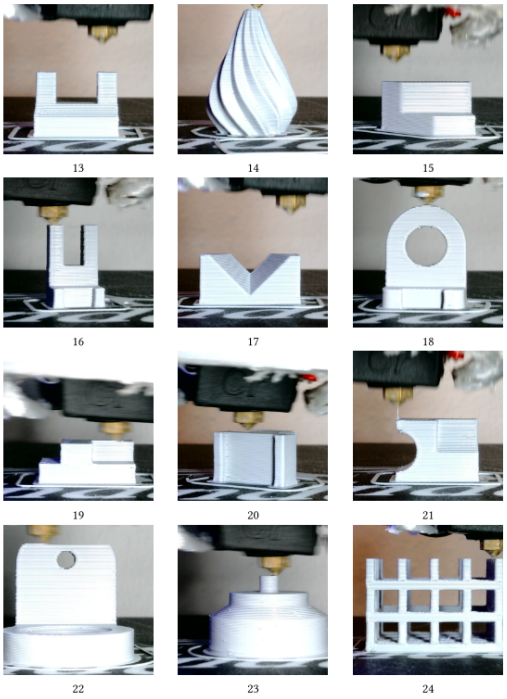

Figure 2: Dataset training images [4]

	Class	Samples
0	Good	5069
1	Under-extrusion	2962
2	Stringing	2798
4	Spaghetti	134

Table 1: Dataset original classes

As of the original state of the dataset, it is not possible to meet the task requirements. It was required to do some transformations on the defective classes *(Under-extrusion, Stringing, and Spaghetti)*, they were merged into a single Defective class, resulting in the following configuration:

	Class	Samples
0	Good	5069
1	Defective	5894

Table 2: Dataset revised classes

Since the models were pre-trained on ImageNet1K, it was necessary to apply certain transformations to the dataset before training. The most notable transformations include resizing and normalization of the images.

2.2 Model Selection

As an exploration and comparison effort two prominent convolutional neural network architectures,

EfficientNet-b0 and ResNet18, were chosen as base models for this task. The motivation for this election was to compare the performance of both models in the same task.

The final architecture used to leverage the task is quite simple, a backbone based on the previously mentioned models pre-trained on ImageNet1K, with the classifier layer overridden for binary classification. The model allows freezing all layers to fine-tune only the classifier layer or allow gradient calculation to all layers via a flag.

2.3 Training Process

For training, a grid search for hyperparameter tuning was done. Focusing on optimizing epochs, learning rate, weight decay, freezing hidden layers, and even backbone architecture to explore its impact on the model's accuracy *(See Table 4)*.

Considering EfficientNets performance [2] it is quite evident that EfficienNet-b0 as a backbone would outperform ResNet-18, the reason to avoid an extensive grid search with ResNet-18.

For statistics, training and validation loss were tracked throughout all the epochs. Of the 7 models trained, the best model trained in terms of performance metrics was model 5.

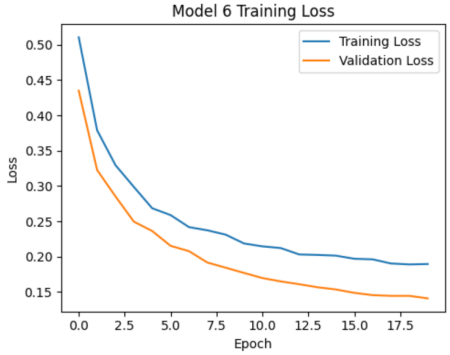

Figure 3: Training Loss, Model 6

3 Results and Discussion

Overall, all the models trained performed well on the testing set, some of them getting excellent results.

3.1 Model Evaluation

Confusion matrix is the main artifact used to evaluate the model's performance. Particularly, F1-score and accuracy were the main metrics used to assess the performance.

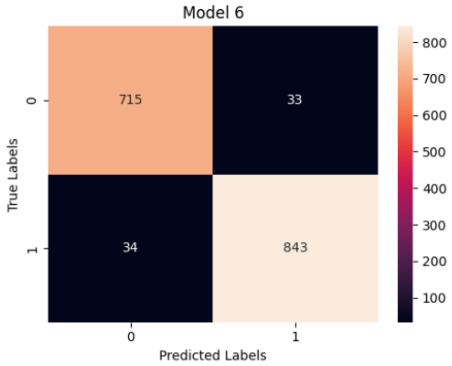

Figure 4: Confusion Matrix, Model 6

Model	Accuracy	F1 Score
1	0.8886	0.8947
5	0.9637	0.9661

Table 3: Performance Metrics (Best/Worst)

3.2 Challenges and Solutions

During the development of the model, finding an optimal threshold was a thing to consider to improve the model's performance. Initially, the *Threshold* was set arbitrarily to 0.7, which led to an accuracy of around $\tilde{0}.87$.

To find an optimal value for the *Threshold* ROC curve was used to find a *Threshold* value that maximizes f1-score.The plot and some testing runs showed that values around (0.50-0.58) are within the optimal range.

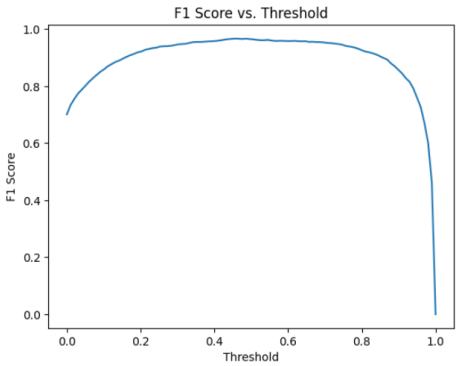

Figure 5: ROC Curve F1-Score vs Threshold

References

[1] Kirill S. Erokhin, Sergei A. Naumov, and Valentine P. Ananikov. Defects in 3d printing and strategies to enhance quality of fff additive manufacturing. a review, 2023.

[2] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2019.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2016.

[4] NILSHAGENBEYER. 3d printing errors, 4 classes of 3d printing error in 34 different shapes, 2024.

[5] Zhiyuan Fang, Jianfeng Wang, Lijuan Wang, Lei Zhang, Yezhou Yang, and Zicheng Liu. Seed: Self-supervised distillation for visual representation, 01 2021.

Appendices

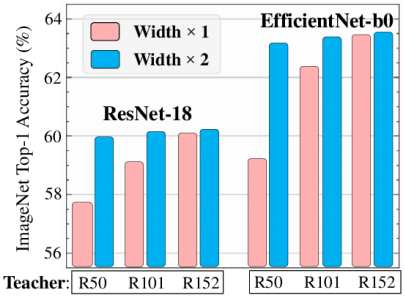

Figure 6: ResNet-18 vs EfficientNet-b0 performance comparison [5]

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
EfficientNet-B0	**77.1%**	**93.3%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**79.1%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**80.1%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.6%**	**95.7%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.9%**	**96.4%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.6%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.8%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.3%**	**97.0%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

We omit ensemble and multi-crop models (Hu et al., 2018), or models pretrained on 3.5B Instagram images (Mahajan et al., 2018).

Figure 7: EfficientNet performance report on ImageNet [2]

Table 4: Grid Search Parameters

	Epochs	LR	Weight Decay	Freeze Layers	Backbone	Accuracy
0	5	0.001	0	True	EfficientNet-b0	0.9132
1	5	0.001	0	True	ResNet-18	0.8886
2	5	0.001	0.001	False	EfficientNet-b0	0.9237
3	10	0.001	0	True	EfficientNet-b0	0.9434
4	10	0.001	0.001	False	EfficientNet-b0	0.9458
5	20	0.001	0	True	EfficientNet-b0	0.9637
6	20	0.001	0.001	False	EfficientNet-b0	0.9588