Kaggle Report: Early detection of 3D printing issues

Yuanbin Cheng

April 20, 2023

How to run my code

Set up

```
pip install -r requirements.txt
```

Training

```
python src/train.py
```

Testing

```
python src/test.py
```

The result will show in the dir: datasets/submission1.csv

My final method introduction

Since this is a problem to solve the overfitting problem, we can not use the print_id during the test. Therefore, I cannot treat every frame in the video as a whole entity and the model can only treat every image as the input and give independent results for the image only.

In the end, I do not modify the baseline model, the major thing I perform is that I ramped up the data augmentation during the training. During the training, for each epoch and each image, I will randomly corp part of the image from the scale in (0.5, 1.0), and randomly change the brightness, contrast, hue, and saturation. Each value I set up is a pretty high value.

During the training, since the training and test datset, we only have one hundred videos, although each video contains thousands of images. The images in each video are very similar to each other. Therefore, the training does not need a very high number of the epoch, which will cause very severe overfitting on the video in the training dataset. In my experiment, I only trained the model for 11 epochs.

The other thing I perform is the voting mechanism during the testing. For each image in the testing dataset, I did the same data augmentation (randomly corp, ColerJitter, etc.). I did the augmentation multiple times, each time the data augmentation will take a different part of the image and turned the color of the image randomly. The model will give a label to each augmentation. In my case, I set up the total augmentation number as 20 (based on the computation recourse I have). Then the final result for each test image is given by the voting from this mechanism. For instance, during the testing, I take 20 patches from an image, and my model will give a label to each patch, if there are more than 10 times, the model gives me the True label, and I will set up the final prediction result as True, otherwise, False.

What I tried to do

Label every image in the same video same (do not allow by the role)

In the beginning, I noticed that there are only 44 videos in the test dataset. And every image in each video is given the same label. So I just train the model, and predict the images in the same video. If in the video (around 2000 images), there are more than 10% or more than 5% of the samples are predicted as True by the model, I will label the whole video True, otherwise, False.

The above method can easily have a result of more than 80%. Based on my experiments. But since the discussion board says that the consist of the label is a BUG not FEATURE, I throw away this try.

Refine the dataset

This method is based on the observation that only part of the images in the training dataset contains the "under extrusion" phenomenon. So I did refine the dataset so that makes only the image does have the "under extrusion" phenomenon given the True label.

However, I give up this method, because not only "training", but also "testing", there are tons of images that are labeled True but do not contains the "under extrusion" phenomenon. The refinement of the training dataset will destroy the consistency between the training and testing.

Modify the model structure

I tried to increase/decrease the layer and increase/decrease the complexity of the baseline model. But the result does not change a lot. I believe that the baseline model itself has enough representative ability for the dataset complexity. Therefore, I decided to remain the complexity of the baseline model and concentrate on other directions.

What in my mind but no time to do

Use some pre-train model as the fixed feature extraction

Use the pre-trained image model as the feature for the classifier. For instance, download some pre-trained models on IMAGE-NET from HuggingFace, then extract the representation layers of the pre-trained networks and use that as the fixed image feature extraction.

Mixup

The idea is from the following paper:

mixup: Beyond Empirical Risk Minimization

https://arxiv.org/abs/1710.09412

Mixed up the samples, which can smooth the decision boundary of the neural network. I believe this idea can increase the robustness ad fix the overfitting problem kind of.