ML-AI Assignment

Dongsheng Yang (D2)
Cognitive Informatics Lab
July 20, 2023

Part 1

Data analysis

a. How many samples are there?

Type	Train	Test
Good	250	180
Not-good	50	
Total	300	180

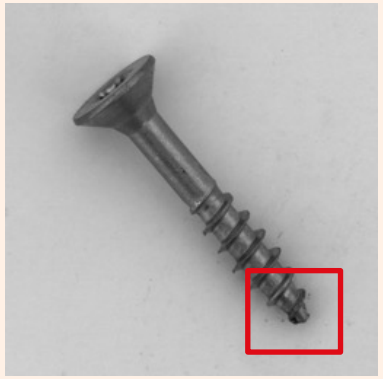
b. Are the samples evenly distributed?

- No, Good samples are 5 times of Anomaly samples.
- It's an imbalanced dataset.

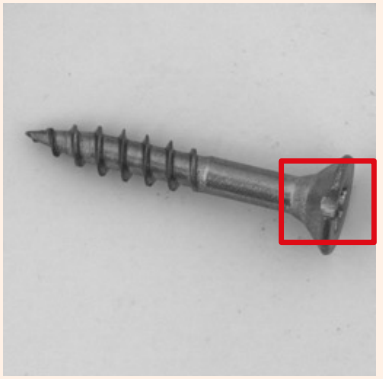
Type	Train	Test
Good	250	180
Not-good	50	
Total	300	180

c. What kind of anomalies are there?

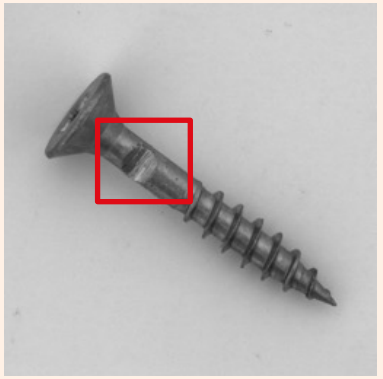
Manipulated_front001.png



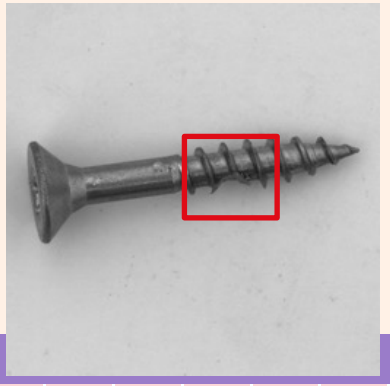
scratch_head001.png

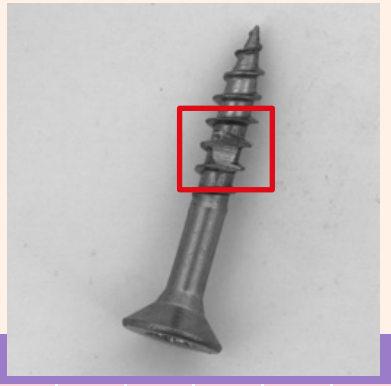


scratch_neck001.png



thread_side001.png





thread_top001.png

d. Is the data sufficient to train a model?

The dataset is a quite small dataset. So, the model would be overfitting easily. Therefore, I used data augmentation for the raw data.

like RandomHorizontalFlip, RandomVerticalFlip, GaussianBlur, RandomRotation, etc.

e. What kind of problems can we expect?

- Training with a small number of dataset will easily cause overfitting, which means the model will have very high performance in training set but low performance in test set.

- Training with imbalanced dataset will cause the model classification ability different for two classes. Thus, accuracy should not be the only measurement for evaluating the model.

f. What can we do to improve the detection of anomalies in this data?

There're multiple ways to improve the detection of anomalies.

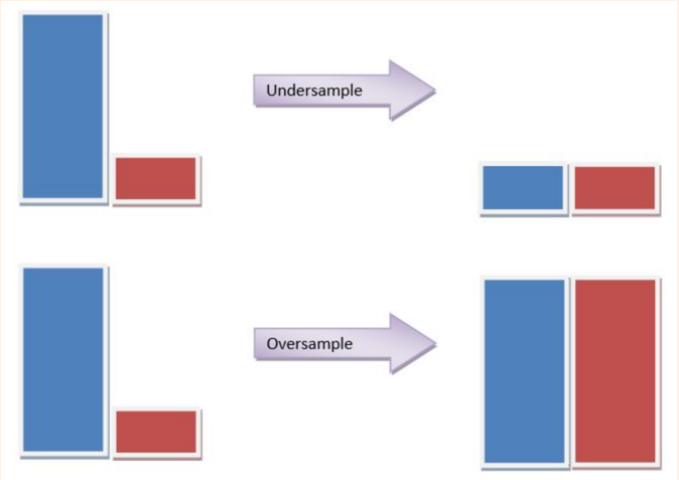
1. Choose Proper Evaluation Metric

Besides accuracy, the other metrics such as **precision** is the measure of how accurate the classifier's prediction of a specific class and **recall** is the measure of the classifier's ability to identify a class. Thus, for an imbalanced class dataset, **F1 score** is a more appropriate metric. It is the harmonic mean of precision and recall and the expression is as below

$$F_1 = 2 * \frac{percision\ * recall}{percision + recall}$$

2. Resampling

This method is to upsample or downsample the majority or minority of the class.



f. What can we do to improve the detection of anomalies in this data?

3. customize loss function

We can also assign weights to each of the classes in loss function. Here's my sample Python code for CrossEntropyLoss in PyTorch.

$$l_n = -w_{y_n} \log \frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})} \cdot 1\{y_n \neq \text{ignore\_index}\}$$

Equation for weighted CrossEntropyLoss (from PyTorch documentation)

```
# ------------------
'''
# use different weighted cross entropy loss while not upsampling the data
import sklearn.utils.class_weight as class_weight
import numpy as np
imbalanced_labels =len([i[1] for i in full_dataset.imgs])
class_weights=class_weight.compute_class_weight('balanced', classes=np.unique
(imbalanced_labels), y=np.array(imbalanced_labels))
class_weights=torch.tensor(class_weights,dtype=torch.float)
criterion = nn.CrossEntropyLoss(class_weights)
'''
# ------------------
```

g. Optional: Anything else that you think is important (graphs, visualizations, measurements...)

Confusion matrix is also a good tool for validation. We will get both True predictions for each classes and ERROR predictions (the false positive and true negative) intuitively.

		Predicted condition	
	Total 8 + 4 = 12	**Cancer** 7	**Non-cancer** 5
Actual condition	**Cancer** 8	6	2
	Non-cancer 4	1	3

Part 2

Coding And Model Building

Model selection

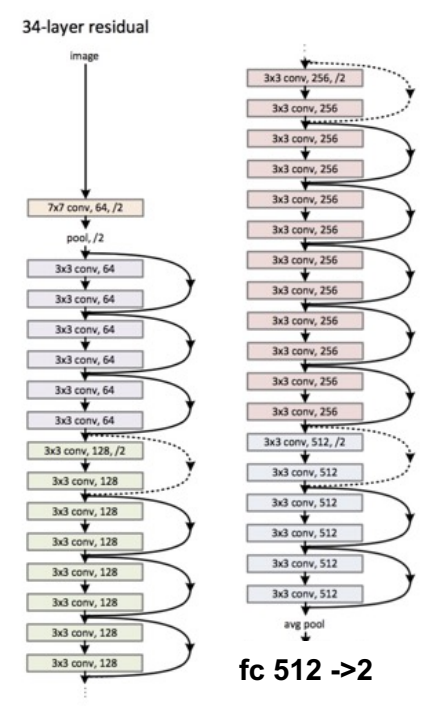
This anomaly detection problem could be regarded as a binary classification problem.

ResNet is noted for excellent generalization performance with fewer error rates on recognition tasks and is therefore a useful tool for our task.

I choose ResNet-34 as the pre-train model, and finetune from ImageNet.

Pre-train weights: **ResNet34_Weights.IMAGENET1K_V1**
Info from pytorch:

acc@1 (on ImageNet-1K): 73.314
acc@5 (on ImageNet-1K): 91.42



Data processing

- Training set has 250 good and 50 not-good figures in total. I split it into 90% for training set and 10% for valid set. (I also tried with a 0.2 ratio for splitting the valid set, but the training result got worse.)

- I also labelled and made my test set using the data in the test folder, which contained 55 good and 55 not-good images.

To deal with the imbalanced data problem, I used weighted sampler in my Dataloader.

```
# get weights for training dataset
weights = make_weights_for_balanced_classes(train_dataset.data, len(train_dataset.classes))

# sampler
samples_weight = torch.from_numpy(np.array(weights))
samples_weigth = samples_weight.double()
sampler = torch.utils.data.sampler.WeightedRandomSampler(samples_weight, len
(samples_weight), replacement=True)

# training loader using sampler, while using sampler, shuffle should be False
train_loader = DataLoader(
    train_dataset, batch_size = batch_size, sampler = sampler, num_workers = 4, shuffle =
    False
)
```

Hyperparameters and Loss func

Hyperparameters

- Learning rate: 1e-4
- Batch size: 64
- Epochs: 60
- weight_decay=1e-3
- optimizer: AdamW

Loss function

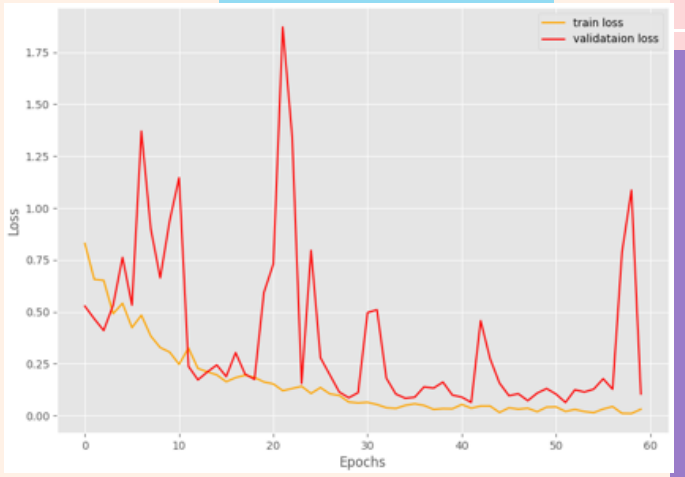
nn.CrossEntropyLoss()

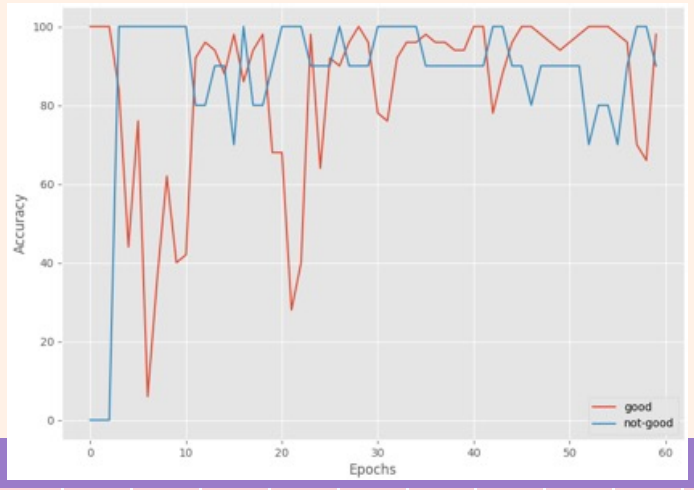
$$l_n = -w_{y_n} \log \frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})} \cdot 1\{y_n \neq \text{ignore\_index}\}$$

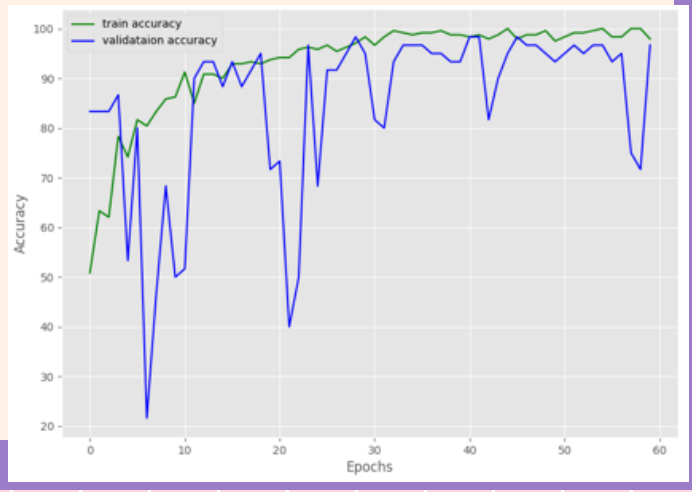
Evaluation - Training

Right top - loss curve (train and valid set)
Right bottom - accuracy curve (train and valid set)
Left bottom - class-accuracy curve (valid set)

Reason: valid set is too small. (good/not-good : 50/10)







Evaluation - Inference

Valid Set (sample: 50/10)

Accuracy: 95.0000, (57/60)

Precision: 0.8182

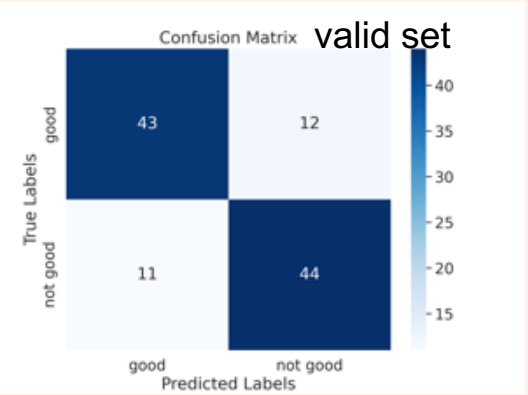
F1 Score: 0.8571

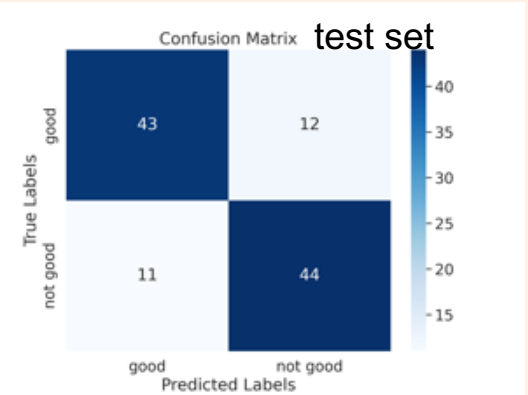
Test Set (sample: 55/55)

Accuracy: 79.0910, (87/110)

Precision: 0.7857

F1 Score: 0.7928





$$precision = \frac{tp}{fp + tp}$$

$$recall = \frac{tp}{tp + fn}$$

where fp - false positive, tp - true positive, fn - false nagetive

$$F_1 = 2 * \frac{percision * recall}{percision + recall}$$

Existing Problems

Reason for this results:

1. Training data is not big enough to learn this task.
2. Training data is imbalanced, which makes it hard to learn the class with small amount of data.

Existing problem:

1. Valid set has limited data, which is not a good criteria for judging which checkpoint has higher possibility to perform well on test set.
2. I could have tried to modify not only the fully connected layer but also the hidden layer of ResNet-34. Also, I could have tried ResNet-50 or other models (like GAN, diffusion model)

Thanks for Listening!