

## DSP 2021 fall Project

姓名: 吳添毅

學號: R10944011

### Part 1

1. I used:

**PyTorch** / Lightning

CPU / **CUDA**

Colab / **PC** / Other: \_\_\_\_\_

2. Kaggle result:

Accuracy	0.99666
----------	---------

3. Preprocessing details. (15pts)

1. 針對 training data 的 2 個 channels 做 Normalization，然後 transform 到 testing data 上。

2. 先做 FFT(Fast Fourier Transform) 後再 Normalization 然後 transform 到 testing data 上。

發現其實效果差不多，用相同架構的 model 搭配不同前處理的 Accuracy 效果很接近。

Preprocessing Architecture	Normalization	FFT & Normalization
3 convolution + ReLU + Dropout(p=0.2)	0.99666	0.99
4 convolution + ReLU	0.99666	0.99333
Similar UNet Arch + ReLU + Dropout(p=0.2)	0.96333	0.96333

4. Method, training details (model arch., why did you select this arch?), experimental setting (hyper-parameters) (30pts):

原本架構為 4 layers of 1d convolution 搭配 ReLU，最後加上 fully connected layer 分成 3 類。→ 0.99666

添加 Dropout(p=0.5) 在每層 ReLU 後面 → 0.98

添加 1d batch normalization 在每層 ReLU 前面 → 0.37

刪減最後一層 1 d convolution 並添加 Dropout(p=0.5) → 0.99666

也有使用類似 Unet 的想法，從一開始的 neurons 為  $20 \rightarrow 40 \rightarrow 80 \rightarrow 40 \rightarrow 20$ 。  
→ 0.96333

最佳架構(總和 accuracy 和 model 大小)為 3 layers of 1d convolution 搭配 ReLU +  
Dropout(p=0.2) → 0.99666

Hyperparameters:

batch\_size = 48

lr = 3e-3

epochs = 30

loss function: Cross Entropy

optimizer: Adam

架構圖:

Layer (type)	Output Shape	Param #
Conv1d-1	[-1, 20, 2284]	540
ReLU-2	[-1, 20, 2284]	0
Dropout-3	[-1, 20, 2284]	0
Conv1d-4	[-1, 40, 325]	8,840
ReLU-5	[-1, 40, 325]	0
Dropout-6	[-1, 40, 325]	0
Conv1d-7	[-1, 20, 64]	7,220
ReLU-8	[-1, 20, 64]	0
Dropout-9	[-1, 20, 64]	0
Linear-10	[-1, 3]	3,843

Total params: 20,443

Trainable params: 20,443

Non-trainable params: 0

Input size (MB): 0.12

Forward/backward pass size (MB): 1.37

Params size (MB): 0.08

Estimated Total Size (MB): 1.57

##### 5. What have you learned (Interesting Findings and Special Techniques)? (30pts)

原本提供的架構 accuracy 就已經很高了，所以我有試著將 convolution 的層數減少和添加 dropout，發現 dropout 掉一些可以避免 overfitting 的問題並順利提升 accuracy。

我加了 Batch Normalization 看 training 的 accuracy 可以達到跟 dropout 一樣的水準，但表現卻非常差，所以我猜可能是 overfitting 太嚴重。  
感覺就只能一直測試不同參數和架構，向 dropout 有測試過 0.8, 0.5, 0.3, 0.2，發現 0.2 好像是最好的，但因為有上傳次數的限制，也不敢隨隨便便亂測試。  
但我發現我原本實作的 model 有 accuracy 為 1，但後來卻弄不出一樣結果 QQQ，我猜可能是因為我用 notebook 的關係導致梯度沒被清掉才會 accuracy 為 1 吧(?)，我後來把 model 的 weight 拆開印出來發現我的每層 layer 都一樣但確實作不出來。

## Part 2

1. Explain your method and result in detail. (7pts)

Accuracy	0.74666
----------	---------

2. Preprocessing details.

與 task1 一樣有做一般的 Normalization 或是先 FFT 再 Normalization。  
但做起來看起來差不多。

3. Method, training details (model arch., why did you select this arch?), experimental setting (hyper-parameters)

我一開始用 Linear Regression 然後輸出 3 類的 log probability 總和，設一個 threshold，低於某一值就當作 anomaly。→ 0.23000

後來想說如果我基於 LR model 的 threshold(低於當作 anomaly)，高於 threshold 則輸入 task1 的 model。→ 0.51000

第一步用 oneclassSVM 和 IsolationForest 來分出 anomaly，之後再將正常訊號丟入 task1 的 model。→ 0.22666

使用 autoencoder 的想法，設定 decode 的訊號的 Error 當作 threshold(低於當作 anomaly)，高於 threshold 則輸入 task1 的 model。→ 0.74666

Hyperparameters:

batch\_size = 128

lr = 3e-3

epochs = 30

loss function: Mean Square Error

optimizer: Adam

AutoEncoder 架構圖:

Layer (type)	Output Shape	Param #
Linear-1	[-1, 2, 8000]	128,008,000
Tanh-2	[-1, 2, 8000]	0
Linear-3	[-1, 2, 4000]	32,004,000
Tanh-4	[-1, 2, 4000]	0
Linear-5	[-1, 2, 2000]	8,002,000
Linear-6	[-1, 2, 4000]	8,004,000
Tanh-7	[-1, 2, 4000]	0
Linear-8	[-1, 2, 8000]	32,008,000
Tanh-9	[-1, 2, 8000]	0
Linear-10	[-1, 2, 16000]	128,016,000
Sigmoid-11	[-1, 2, 16000]	0
Total params: 336,042,000		
Trainable params: 336,042,000		
Non-trainable params: 0		
Input size (MB): 0.12		
Forward/backward pass size (MB): 1.25		
Params size (MB): 1281.90		
Estimated Total Size (MB): 1283.27		

4. What have you learned (Interesting Findings and Special Techniques)?

網路上搜尋 one class classification, anomaly detection 大多都是 oneclassSVM 和 isolationForest 這 2 個方法，但我自己測試發現它其實會把很多正常的訊號也判為異常，用起來成果也不怎麼好，所以就朝著 Neural Network 前進。

原本想要嘗試 PCA 的，但礙於時間問題所以就沒嘗試， autoencoder 我也是第一次寫，所以也只是參考 pytorch 官網的基本教學去做修改，我發現我的 neurons 數目會大大的影響我的 accuracy，但時間不夠多所以也只有測試幾種排列組合。