DSP 2021 fall
Project

姓名:劉旭庭
學號:r10922129

Part 1

1. I used:
   Python
   PyTorch Lightning
   CUDA
   Workstation(Linux)

2. Kaggle result:

Accuracy	0.99666

3. Preprocessing details. (15pts)
   1. 將兩個 channel 作 StandardScaler normalization
   2. 將兩個 channel 作 MinMaxScaler normalization

   最後發現方法一效果較好，可能因為 training data 中有一些 outliers，因此在做 MinMaxScaler 時的 scale 會太極端，可能可以改善的方法是在此之前先把 outliers 找出來(不過會因此花較多時間處理)。

   相比之下，StandardScaler 在做 normalization 時會用到 standard deviation，比較不容易受到 outliers 影響，且這個 normalization 的 model 是以高斯分佈為基礎。

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

這次作業先簡單跑了助教在課堂的範例架構(簡稱 basic)：

```
nn.Conv1d(2,20,kernel_size=13,stride=7),
nn.ReLU(),
nn.Conv1d(20,40,kernel_size=11,stride=7),
nn.ReLU(),
nn.Conv1d(40,80,kernel_size=9,stride=5),
nn.ReLU(),
nn.Conv1d(80,160,kernel_size=7,stride=5),
nn.Linear(1920,3)
```

batchsize=48
optimizer=adem(3e-3)
epoch=30
效果還不錯(acc:97%)

接著有嘗試把 data 轉為 2d 的 spectrum(spicy.fft)，利用 2d convolutional network 來訓練(大致架構如上，kernel size 由大到小為 5,5,3,3，stride:3,3,2,2)，但結果不理想(acc:90%)，可能因為 2d convolutional network 較複雜，在這個簡單的 task 上面資料不多的情況下很容易 overfitting。

於是回到原本的 1d convolutional network，並且把模型改為

```
nn.Conv1d(2,32,kernel_size=13,stride=7),
nn.ReLU(),
nn.Conv1d(32,64,kernel_size=11,stride=7),
nn.ReLU(),
nn.Conv1d(64,128,kernel_size=9,stride=5),
nn.ReLU(),
nn.Conv1d(128,256,kernel_size=7,stride=5),
nn.Linear(3072,3)
```

batchsize=48
optimizer=adem(3e-3)
epoch=30
發現結果比 basic model 好，於是基於新的 1d convolutional network 來調整參數，讓 epoch 從 30 變成 50，performance 有些微進步，後來把 batchsize 改為 24，結果也有提升
batchsize=24
optimizer=adem(3e-3)
epoch=50

5. What have you learned (Interesting Findings and Special Techniques)? (30pts)
   助教講解作業的時候有提到 pytorch lightening，之前只有聽朋友講過沒實際用過，之後有自己去官網看 pytorch lightening document 還有講解的影片，有許多方便的功能(不用自己寫 training loop 還有 validation loop，可以利用 callback 的功能去儲存最佳的 checkpoint 或是 earlystopping)，寫完這次 project 之後我會把 pytorch lightening 的 document 看熟，應該還有很多方便好用的功能以及寫法可以發掘，是我這次寫 project 的大收穫。
   另外這次 project 也讓我明白理論與實作的差異性，以理論來說這次 project 很簡單，model 照理來說一定可以學好，但實作上 network 的設計還有超參的調整牽一髮動全身，有時候微調一些東西 performance 就直接爆炸，況且一天只能在 kaggle 測 5 次，要如何在短時間內找到最適合且有效率的架構設計我覺得十分需要實作經驗，並非看課程影片或是講義就能一步設計到位。

Part 2

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

Accuracy	0.51333

方法是利用 train 好的 classifier(part1)，把 class 0,1,2 分別丟進去取最後一層 layer 的值，利用這些值來 train sklearn 的 EllipticEnvelope，類似 oneclass classifier，分別對 class 0,1,2 各訓練一個 EllipticEnvelope classifier，但 EllipticEnvelope 比較不會 overfitting。
在 predict 時會先把資料丟近 classifier(part1)，得到最後一層 layer 的值，然後再丟近三個 EllipticEnvelope classifier 做比較，如果三個 classifier 都 output -1(data 不屬於 class 0,1,2)就 predict 3，否則就按照原本的 classifier(part1)來分類。

訓練 EllipticEnvelope classifier 時的參數為(contamination=0.2)，訓練資料比例是 1200:150:150，其中 1200 是自己的類別。

(結果發現 class 2 的 EllipticEnvelope 很容易把 class 3 分到 class 2，短時間內還想不到如何改進)