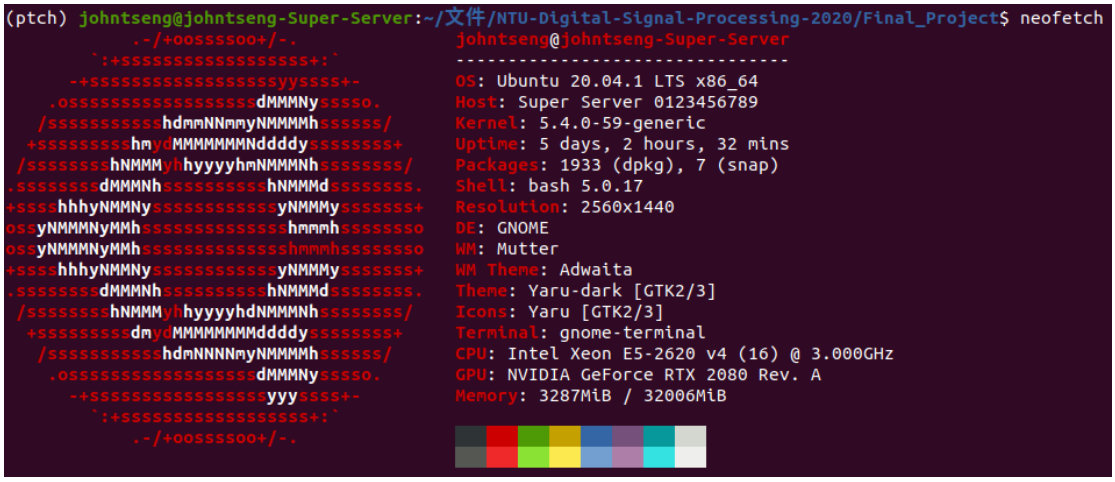
數位訊號處理導論 Final Project 資工碩二 曾品翔 R08922126

開發環境:



Model & Hyperparameter Setting:

Model:

Layer1: conv1d(kernel size: 7, stride: 2, padding:3, group: 2) + sigmoid + batchnorm

Layer2: conv1d(kernel size: 7, stride: 2, padding:3, group: 4) + tanh + batchnorm

Layer3: conv1d(kernel size: 7, stride: 2, padding:3, group: 8) + relu + batchnorm

Layer4: conv1d(kernel size: 3, stride: 2, padding:1, group: 8) + relu + batchnorm

Layer5: conv1d(kernel size: 3, stride: 2, padding:1, group: 8) + relu + batchnorm

Layer6: flatten + fully connected (64*625 連接到 4)

Hyperparameter:

Training batch size: 128

Number of epoch: 100

Optimizer: Adam, weight decay: 0.001

Loss criterion: CrossEntropyLoss

實驗過程:

Input format and preprocessing detail:

這次作業的輸入格式是一筆資料中有兩個一維的訊號要做處理。我一開始就先把資料丟到 CNN model 中，直接使用 conv1d 來做 CNN，發現效果並不好。然後把資料全部減掉平均，然後再除以最大值，也就是把資料的範圍限縮到 0~1 之間，發現這樣效果也沒有提升。仔細看了一下資料後發現，因為兩個資料其實滿不一樣的。

Channel 1: Vibration data

Channel 2: Rotational speed data

兩個資料當然不能直接用 conv1d 做運算，因為 conv 就是一個 filter 會和所有的 channel 做運算完之後相加得到一個新的 feature map，那兩個 channel 所代表的意涵大相逕庭，所以不能直接用 conv1d 也就滿合理的。

Method:

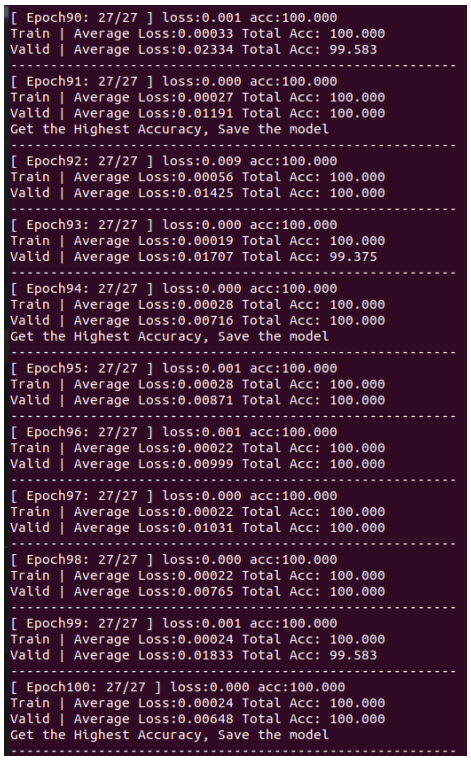
基本上方法我是直接選擇了 CNN 來做這次的作業，從前面 data 的樣貌來看，不能夠直接無腦的用 conv1d 來做完作業，兩個不同的資料必須要分開處理。好險 conv 的運算中有提供 group 這項參數來供大家調整 model，這個 group 的作用是把 input channel 分群計算，因為剛開始的 number of input channel 是 2，group 設成 2 就代表把兩個 channel 分開來做 conv1d，那就很棒的解決了上面發現的問題，在 model 中，我將 group number 逐層放大單純只是覺得可以分更多群而已，發現這樣好像有收斂得快一點。

下一步就是 model 堆積木的過程了，因為 dataset 本身並不大，所以我的 hidden layer 只有 5 層的 CNN 加上一層 fully connected，然後 Hyperparameter 的選擇，因為剛開始 input channel 是 2，後面就逐層放大 2 倍。另外有考量到訊號資料，如果 kernel size 設定太小，感覺看到的視野太過侷限，所以我的 kernel size 就設為 7，並且連續疊三層，每一層的 conv1d 都有把 stride 設定為 2，是為了使每次的 feature 能夠縮減下來。

最後其他 activation 的選擇，放了 sigmoid 和 tanh 只是單純為了有不太一樣的選擇，效果其實並沒有差到很多。經過前面的 conv1d 做完，就算 channel1 和 channel2 兩個的意涵不相同，分開來做 conv1d 計算之後就簡單的把所有的 feature 做 flatten 並且接上 fully connect 到 output，就完成了我的 model。

Model selection:

上面的做法都只有使用的了 training data，那麼 validation data 就是要拿來做 model selection 使用的。作法就是每次使用 training data train 完一個 epoch 之後，就會使用 validation data 來做 validation。因為資料都有給 label，所以我就是按照誰的 accuracy 比較大，還有 loss 也要比較低，就選擇那個 model，要同時滿足兩個條件我才會儲存 model，因為 train 到最後其實可以發現 validation accuracy 可以到 100%，所以不得已才要把 loss 放進來作為考量。



可以看到最後幾個 epoch 的訓練狀況，真的就是 train & validation 的 accuracy 都是 100，但是每個 epoch 完都還是有所變動。

作業心得:

第一個心得就是因為我這個作業是公布完 final project 的隔天我就已經快要弄得差不多了，那時候丟到 kaggle 上面拿到了 99.583 的正確率，那時候還是第一名，我就心滿意足的先不做了。後來過好一陣子之後，打開來發現一堆人都拿了 100%正確率，我就開始嘗試使用不同的 model 來做，結果我的 accuracy 就再也沒有突破過 99.583 了，之後就是比較低的數據。

另外我在做訓練的過程中，其實有發現可能是資料集太小吧，所以丟進深度學習裡面，常常會有 model 不太穩定的情況，常常訓練到一半可能 loss 就大爆炸，所以 batch norm 是一定要加的，weight decay 也是因為發現 model 太不穩定，所以才放進來的。其實這代表的我的 model 可能參數還是太多，導致 overfitting 的情況發生，但是看了一看 model 的參數我已經是盡量能少就少了，太少又更學不起來。然後很奇怪的現象是，我有沒有去對 input data 做正規化，結果好像都差異不大，所以後來做通通沒做了。另外關於 conv1d 的參數調整，其實我調了非常多種的組合，最後發現效果就卡在一個極限上不去，那可能就是 conv1d 的極限就在這裡了。

有時間的話我其實滿想嘗試看看使用 spectrogram 下去搭配 conv2d 去做實驗的，不確定效果好不好，然後再使用 random forest，這樣其實就有三個不同

的方法來做同一筆資料，就可以做 ensemble，也就是使用投票機制來做決定，只是期末有點忙就沒有特別去做嘗試。

關於 model 不穩定這點，還有很多心得可以補充，就算是用同一個 model 來做訓練，也使用同樣的設定，但是我丟到 kaggle 得到的結果居然也可以不一樣，而且差距還滿大的，可能從 92%~99%的正確率都有，讓我覺得靠運氣的成分非常大。所以關於 reproduce 的地方，其實還真的是滿困難的，畢竟每次 model 訓練完都不太一樣。

使用 loss 去選 model，這點其實爭議也不小，因為不是 validation loss 越小，就代表得到的 accuracy 越高，我也曾經把 validation loss 降到 0.001 以下，但是得到的正確率卻不高，所以這代表我的方法還是有先天的缺點在吧，可能 input data 不能使用 raw data，還是要轉換成 spectrogram 再去試試看才有機會達到其他同學的 100%。

關於這個 final project，整體來說是滿有趣的，雖然我以前就聽過 kaggle 也註冊過 kaggle，但是從來沒有打比賽過，是個滿新鮮的資料科學競賽的體驗，從分析資料看資料到最後看 kaggle 排名，都是滿有趣的一段過程。