# Report

b03705027 鄭從德

# Model description

## 1) RNN

我的 RNN model 最終的架構如下：

```
274  opt = RMSprop(lr=0.0004, decay=1e-6, clipvalue=0.5)
275  model = Sequential()
276  model.add(Bidirectional(GRU(512, recurrent_dropout = 0.35, dropout=0.4,
277  model.add(Bidirectional(GRU(256, recurrent_dropout = 0.5, dropout=0.5,
278  model.add(Bidirectional(GRU(128, recurrent_dropout = 0.5, dropout=0.5,
279  model.add(GRU(256, recurrent_dropout = 0.5, dropout=0.5, return_sequenc
280  model.add(TimeDistributed(Dense(NUM_CLASS, activation='softmax')))
281  model.compile(loss='categorical_crossentropy',optimizer=opt)
```

一共用了 4 層的 rnn，最後一層的 RNN 沒有使用 Bidirectional 是因為 training 時間會過久。

這個 model 是我最好的 model，訓練到第 350 epoch 左右會達到 validation loss 最低點，val_loss 約 0.198，Levenshtein Distance 約 6.95。

## 2) CNN+RNN

我的 CNN+RNN 最終 model 如下：

```
269  opt = RMSprop(lr=0.001, decay=1e-6, clipvalue=0.5)
270  model = Sequential()
271  model.add(TimeDistributed(Conv2D(32, (3,3), activation='relu',padding='
272  model.add(TimeDistributed(MaxPooling2D(pool_size=(3,2))))
273  model.add(Dropout(0.4))
274  model.add(TimeDistributed(Flatten()))
275  model.add(Bidirectional(GRU(512, recurrent_dropout = 0.4, dropout=0.4,
276  model.add(Bidirectional(GRU(128, recurrent_dropout = 0.45, dropout=0.45
277  model.add(Bidirectional(GRU(64, recurrent_dropout = 0.43, dropout=0.43,
278  model.add(TimeDistributed(Dense(NUM_CLASS, activation='softmax')))
279  model.compile(loss='categorical_crossentropy',optimizer=opt)
```

資料前處理的部份，我將每個 frame reshape 成(3,39)，加入前後兩個 frame 的 data，變成一張類似圖片的 input。我一共做了一層的 Convolution 2D，經過 max pooling 之後在通過三層 RNN 。在經過約 65 個 epoch 之後會達到 val_loss 的最低點約為 0.220，分數為 8.2。

# How to improve your performance (1%)

1. **Normalization**：我對於每一筆 sample 的 39 個(MFCC 表現比較好) feature 各進行一次 normalization，這樣照理說可以去掉**不同的人**在不同 feature 值特別高或特別低的差異，

儘留下一句話不同音節的差異。還有，Normalization 應該在 padding 之前做，才能避免叫短的句子內容被稀釋。加上 normalization 後，在 LD 分數上可以進步約 0.4 分。(8.9->8.5)

2. **選擇 Bidirectional：**選擇 Bidirectional 對我的 model 幫助很大，在使用之後有顯著的進步（LD 好幾分）。選擇他的原因是因為以前在 text 的處理上就有顯著幫助，而且這次的語音處理也應該是類似的 sequence 處理。加入 Bidirectional 之後，每一層 hidden layer 的 output 數量變兩倍，相當於 kernel size 開兩倍的複雜度，但是我透過 Bidirectional 與同複雜度的 RNN 的比較 ：**Bidirectional(GRU(256)) vs. GRU(512))**，結果是 Bidirectional 的效果好很多。

**3. Ensemble：**我所作的 ensemble 是透過兩個分數接近，胖瘦不同的 RNN model 進行。就將兩個不同 model 的 output 機率全部相加，在選擇最高的 output 當作最終答案。我拿了兩個 val_loss 約為 0.22, 0.23, 分數為 8 分左右 的 model 進行 ensemble 之後，分數可以進步到 7.5。

**4. Output 篩選：**我在每一個 frame 預測出結果之後，輸出 csv 之前，加了一個簡單的篩選條件：只選擇連續輸出超過一次的 label。這麼做是為了要避免突然有一些 frame 預測出來的結果很奇怪，如果也把他視為 output label 的話我的 LD 分數一加一減一次就會跟正確答案差 2。至於如何選擇這個次數的 threshold，是透過實際測驗發現一次是最好的。我一個普通 8.7 分的 model，透過 threshold= 1(連續超過 1 次就選)的篩選之後可以進步到 7.2，但是如果 threshold 設定為 2，分數就變成 8.5，threshold=3 以上就會 error 比原本還要大。

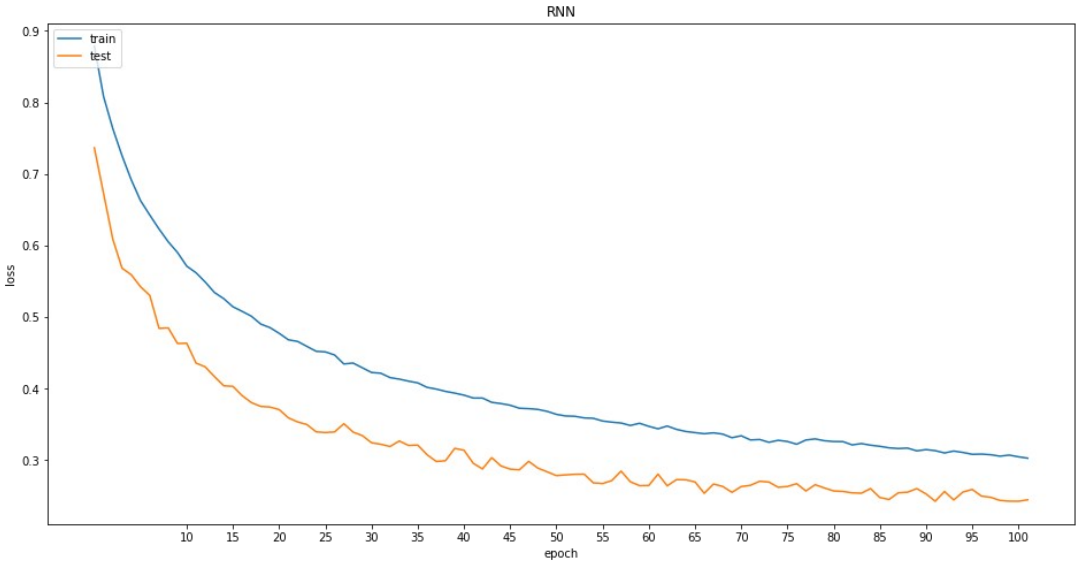
## Experimental results and settings (1%)

我利用一個比較沒那麼複雜的三層 RNN model（不然要 train 太久）來進行實驗，嘗試直接比較有無 CNN convolution 對於 Learning Curve 的影響 。

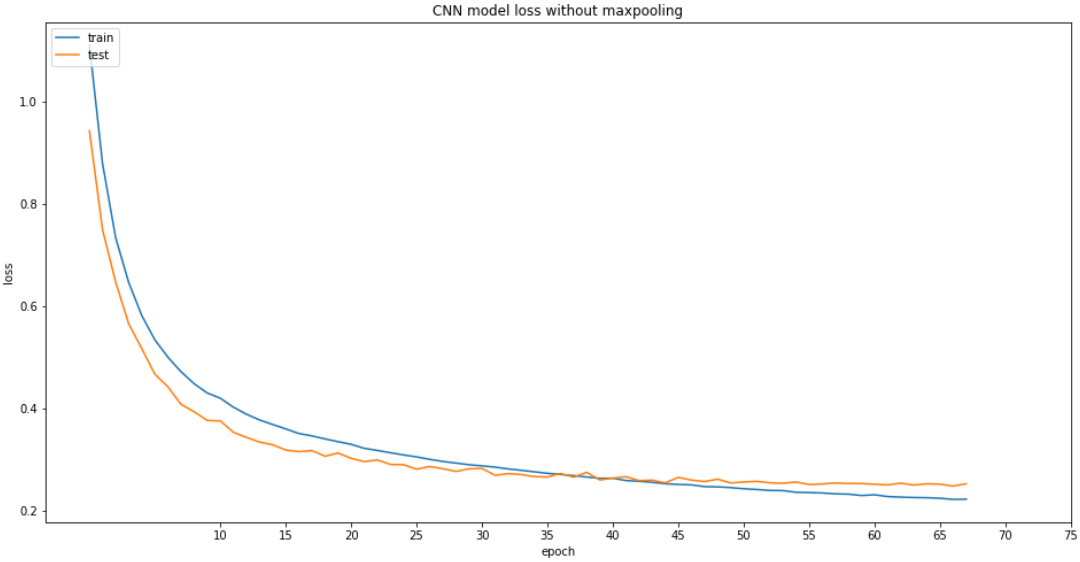
原始 RNN model:

```
model.add(Bidirectional(GRU(512, recurrent_dropout = 0.4, dropout=0.4,
model.add(Bidirectional(GRU(128, recurrent_dropout = 0.45, dropout=0.45
model.add(Bidirectional(GRU(64, recurrent_dropout = 0.43, dropout=0.43,
model.add(TimeDistributed(Dense(NUM_CLASS, activation='softmax')))
```

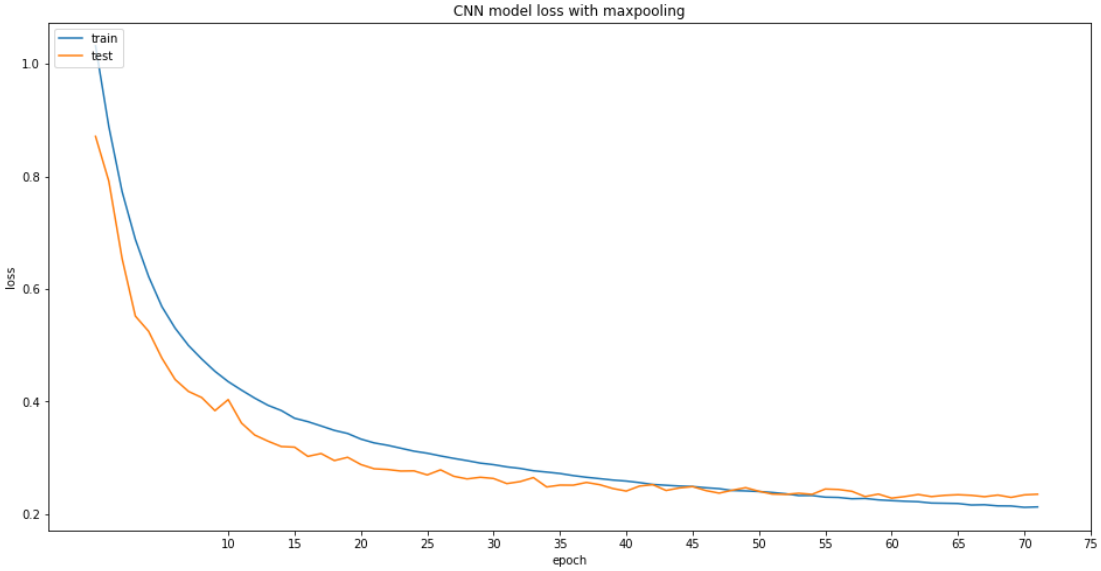
原始 RNN training curve（GRU）: 前 100 個 epoch




前面加上 CNN 2D，without max pooling:




加上 CNN 2D，with Max Pooling:



我們可以從上這幾張圖中得出一些結論：

### 1) Max Pooling vs No Max Pooling

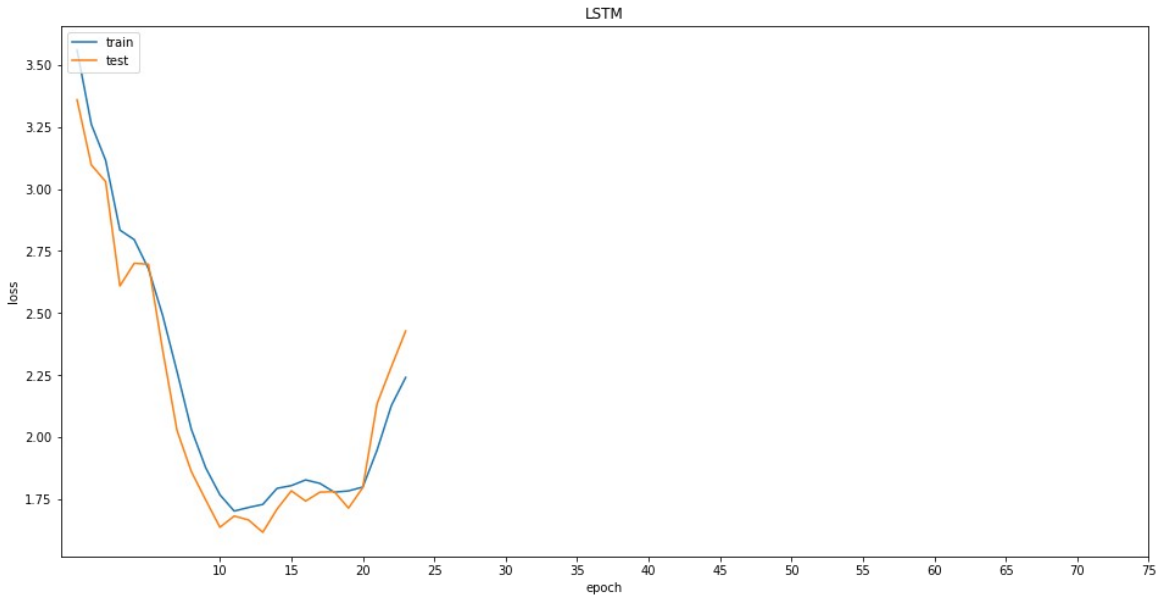
從後兩張有沒有 max pooling 的兩張圖我們可以看出，有 max pooling 的 model 訓練時間也較短，表現也較佳，可能是因為沒有經過 maxpooling 的 model 最於後面的 RNN 而言可以過度複雜。最終的 val_loss，有 maxpooling 的可以降到 0.220，而沒有的則為 0.24，所以之後我的 CNN 的 model 都會加上 maxpooling。

### 2) RNN vs CNN

可以看出第一張的 RNN model 明顯的要比後面的 CNN 健康很多，訓練到第 80 個 epoch 還完全沒有 overfit 的問題(使用同樣的 lr 參數)，相比同樣的 val_loss 下的 training loss 也可以發現 RNN 高上許多，有更多進步空間。所以我後來 model 都選擇 RNN 因為比較好 train，表現較好，也不會太複雜。我另外實做過 Con1D, 還有多幾層的 Conv2D，但是效果都沒有 RNN 好。所以最終仍然選擇增強 RNN 為主的 model。

### 3) GRU vs LSTM

補上一張我的 model 換成 LSTM 的悲劇照：




比起 GRU 難 train 很多，model 較複雜，訓練時間較長(約 10%)，而且容易噴 nan，所以我都選擇 GRU。