

1. Model Description

Train dataset 總共提供了 3696 筆 sentence 資料

在此我假設每一句 sentence 最長的 phone length 是 800

並且採用 MFCC + FBANK (39 + 69 = 108 dim) 作為 phone features

而因為要輸出的 framewise prediction 是 based on 39 phones，所以我的 label 也直接轉換並設成 39 dim

因此我的訓練特徵 X.shape 為 (3696, 800, 108)

訓練標籤 y.shape 為 (3696, 800, 39)

另外，每個 sentence 的原始 phone length 長度不一，因此如果超過 max_sent_length 800 則必須要砍掉多出的 phone sequence，而若低於 max_sent_length 800 則是必須要做 padding，而我的 padding method 是做 phone sequence repeat 直到 max_sent_length。

Phone sequence repeat 示意：

(假設 max_sent_len 為 8，但該 sample 長度只有 3)

[1,2,3] -> repeat padding -> [1,2,3,1,2,3,1,2]

接著我切 10% 資料作為 validation，剩下 90 % 作為 training，並餵進 RNN 和 RNN + CNN Model (以 Keras 實作)

A. RNN

Layer (type)	Output Shape	Param #
simple_rnn_1 (SimpleRNN)	(None, 800, 1024)	1160192
batch_normalization_1 (Batch Normalization)	(None, 800, 1024)	4096
dropout_1 (Dropout)	(None, 800, 1024)	0
time_distributed_1 (TimeDistributed)	(None, 800, 39)	39975
Total params: 1,204,263		
Trainable params: 1,202,215		
Non-trainable params: 2,048		
Train on 3326 samples, validate on 370 samples		

Layer1: SimpleRNN # 資料屬於連續時間序列適合使用 RNN 架構

Layer2: BatchNormalization # 將 layer1 的輸出做標準化

Layer3: Dropout # 用於避免 overfitting

Layer4: TimeDistributed (Dense) # 最後的輸出層

使用 TimeDistributed 將 Dense 包裝為對時間序列輸入處理的層

最後的 activation function 為 softmax

Loss Function 採用 categorical_entropy，Optimizer 採用 adam

B. RNN + CNN

Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 800, 512)	332288
batch_normalization_1 (Batch Normalization)	(None, 800, 512)	2048
dropout_1 (Dropout)	(None, 800, 512)	0
simple_rnn_1 (SimpleRNN)	(None, 800, 512)	524800
batch_normalization_2 (Batch Normalization)	(None, 800, 512)	2048
dropout_2 (Dropout)	(None, 800, 512)	0
time_distributed_1 (TimeDistributed Dense)	(None, 800, 39)	20007
Total params: 881,191		
Trainable params: 879,143		
Non-trainable params: 2,048		
Train on 3326 samples, validate on 370 samples		

Layer1: Conv1d # 使用 CNN 可以學到 phone neighbor 之間的訊息

Layer2: BatchNormalization # 將 layer1 的輸出做標準化

Layer3: Dropout # 用於避免 overfitting

Layer4: SimpleRNN # 資料屬於連續時間序列適合使用 RNN 架構

Layer5: BatchNormalization # 將 layer4 的輸出做標準化

Layer6: Dropout # 用於避免 overfitting

Layer7: TimeDistributed (Dense) # 最後的輸出層

使用 TimeDistributed 將 Dense 包裝為對時間序列輸入處理的層

最後的 activation function 為 softmax

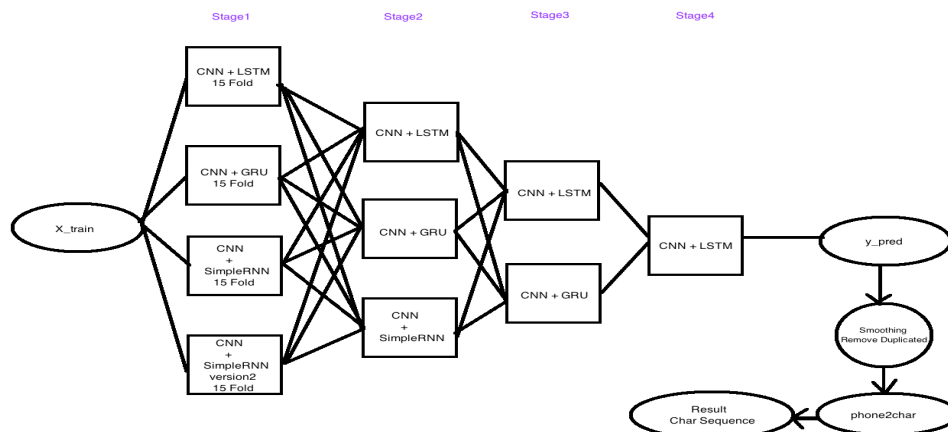
Loss Function 採用 categorical_entropy，Optimizer 採用 adam

2. How to improve your performance

A. Describe the model and technique

以下為我 Best Model 的架構，主要是使用了 Ensemble Learning

(每個方框內受限於版面，只大致描述該 model 架構)



B. Why do you use it

a. Stacking

將多個 model predict 的結果 concatenate 起來餵入下一個 Stage，以 Stage1-2 為例，因為 Stage1 的代表 model 有四個，輸出都是 39 dim，因此每個 Stage2 model 的 input shape[2] 會是 $39 \times 4 = 156$ 。此概念類似於老師於上課中提過的 word2vec，後面的 model 在 train 時可以反饋訊息回到前面 Stage 的 model，因此是個不錯的 model 合併的方式。

b. K-Fold

K-Fold 意指將資料切成 K 份，每次選擇其中一份作為 validation set，其他作為 training set，因此會 train K 個 model。由於 Stage1 是比較重要的部分，他的影響會一直延續到後面 Stage 的 model，因此我將 Stage1 中的四個 model 都做了 15-Fold，並將 15 個 model predict 的機率結果相加作為該 model 的輸出代表。做 K-Fold 的優點是整合多個 model 的同時也解決了因為要做 train-test-split 而使得可能部分的資料被拿去當 validation 而沒有被 train 到的問題。

c. Smoothing

最後的結果可能會有如右邊的例子：aaaa**b**aaaaa**cc**aaaa

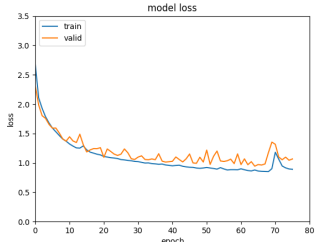
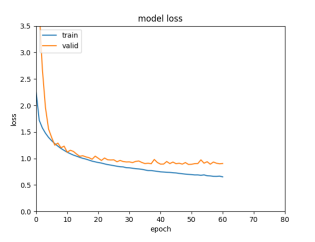
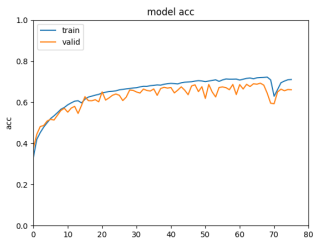
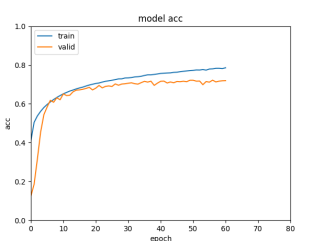
而經過觀察 training 的 label 我們得知通常 phones 是具有連續性的，因此我們可以很大膽地推測上述 sequence 中 b 與 cc 很可能是 model 的誤判，而可以矯正回 a 和 aa。實作上我設了一個 smooth range (3~5)，在該 range 內如果發現有相同的 phones 存在，就將兩個 phones 之間的其他 phones 同化。

3. Experimental results and setting

[以下使用相同的 Training 和 Validation 且 batch_size = 128 , patience = 10]

A. Compare and analyze the results between RNN and CNN

我以我第 1 點的兩個 model 來做比較

	RNN	CNN + RNN
參數量	1204263	881191
Epoch Time (GPU)	29s	22s
Best Record	# 65 epoch	# 50 epoch
Loss		
Train/Valid Loss	0.8796/0.9442	0.7003/0.8870
Acc		
Train/Valid Acc	0.7142/0.6899	0.7714/0.7216

RNN + CNN 在參數量較少的情況下，在各數據表現仍都優於 RNN

B. Compare and analyze the results with other models

	CNN + SimpleRNN	CNN + LSTM	CNN + Bidirectional LSTM
參數量	881191	918183	883111
Epoch Time (GPU)	22s	43s	68s
Best Record	# 50 epoch	# 27 epoch	# 24 epoch
Train/Valid Loss	0.7003/0.8870	0.6545/0.8323	0.5160/0.6920
Train/Valid Acc	0.7714/0.7216	0.7844/0.7321	0.8296/0.7801

在參數量差不多的情況下，整體而言，我們可看出 LSTM 和 Bidirectional 都有助於 model 的 performance