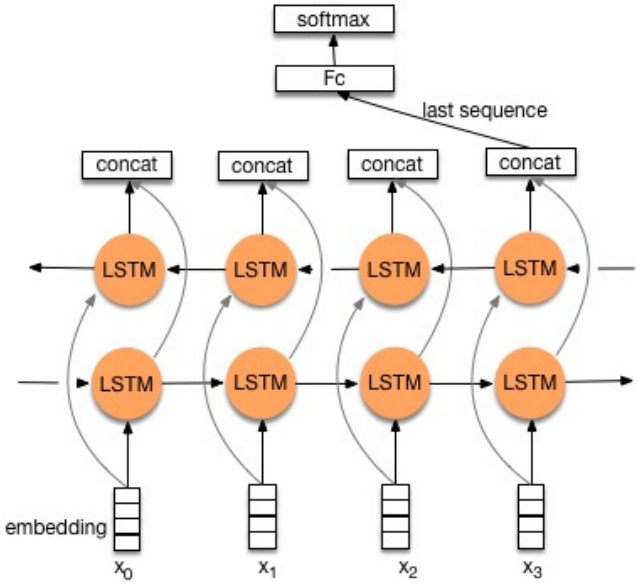
ADLxMLDS HW1 Report

R06922022 資工所碩一 曹燁文

（一）模型描述

1. RNN (使用 mfcc feature)



credit: http://doc.paddlepaddle.org/release_doc/0.9.0/doc/_images/bi_lstm1.jpg

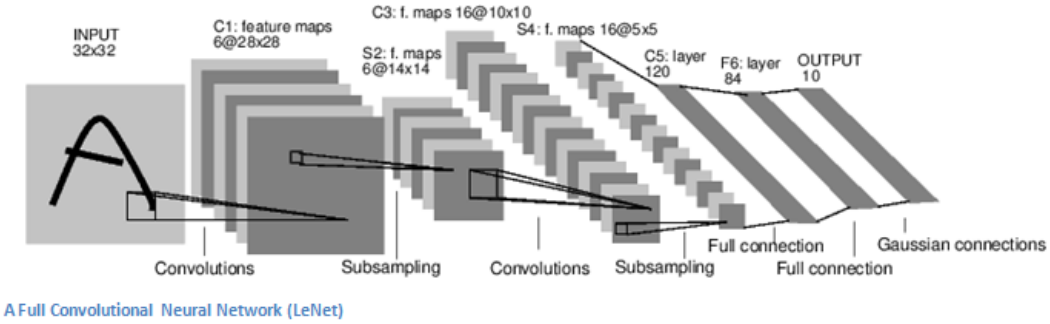
使用 Tensorflow 的雙向 LSTM，一層 hidden layer，大小 512。模型大致如上圖(沒有用到 embedding)，由於使用 Tensorflow 的 dynamic_rnn，訓練時可以每個 batch 輸入不同長度的句子，於是我將一個句子當成一個 batch 輸入，之後透過兩個相反方向的 LSTM 的訓練，因為將 output 接在一起，輸出到 FC (Dense) 之前維度變成 1024，再透過 FC 轉成 48 維進入 softmax。

產生答案之後，使用我稱為"Trimming"的技巧，將雜訊簡單過濾掉，可以讓 performance 大幅的提高。具體描述如下：由於音訊是連續的，透過觀察答案可以發現每組 phone 是連續出現 3 次以上，於是我透過判斷每個 frame 前後的關係，例如 AABAA，就改成 AAAAA，等判斷，將其結果平滑化，以降低 edit distance 的錯誤率。

2. CNN+LSTM (使用 mfcc feature)

使用 Keras 的 CNN 和 LSTM。首先預處理時將每個 frame 的前後 5 個併起來，成為 11*39 的圖片，且我將句子切開，變成一句 70 個 frame，每句之間有 10 個 overlap，於是 input 變成(batch, 70, 11, 39, 1)。CNN 使用兩層，分別用 32 和 64 當 filters 數量，kernel size 都是 3x3，且都有接 2x2 的 Maxpooling，過 flatten 之後放進 4 層 hidden layer size 256 的雙向 LSTM 訓

練，如之前的模型將兩個輸出接在一起再進入下一層。最後一樣透過 Dense layer 轉成 48 維進入 softmax。下圖為一個 CNN 的範例。



credit: https://adeshpande3.github.io/assets/LeNet.png

產生答案之後使用了我稱作”Threshold”的技巧(感謝 r06922062 張心愉同學提供)，即輸出答案的時候，若是該 frame softmax 出來的機率未達到設定的 threshold，就不輸出該 frame，實驗證明在 validation 上有使用這個技巧十分有效，且使用”Threshold”之後再搭配”Trimming”結果更好。

3. Best Model (使用 mfcc+fbank 共 108 維 feature)

由於 CNN+LSTM 的訓練速度很慢，也很吃記憶體，於是決定使用單純的 LSTM，並且加上其他技巧優化。一開始先把 fbank 和 mfcc 合併，使用雙向 LSTM 共 6 層，每層的 hidden layer size 是 400，並且在第 3 和第 6 層加入 dropout，雙向的兩個結果不像之前用 concat 模式，改成 sum 模式，可以不用讓中間的 feature 數量變成兩倍。輸出結果亦使用”Trimming”和”Threshold”，這使我第一次在 leaderboard 上面突破 7。

之後參考了 Hinton 等人的” Distilling the Knowledge in a Neural Network” 這篇 paper，其中提到最重要的概念是

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

即在一般的 softmax 式子中加入”溫度 T”，一般的 softmax T=1，若是提高 T 可以讓輸出的結果更”soft”，例如原本的輸出結果是[0.1, 0.8, 0.1]，提高 T 可以變成像是[0.25, 0.5, 0.25]，讓各個 label 的機率更加接近一些。作者認為這樣提高熵的方式可以留住更多資訊。

與作者概念不同，不是利用 soft target 和 hard target 再訓練一個有相同 performance 的小 model，我是利用 soft 的概念，將原先 train 好的 model 的 softmax 加入 soft 的概念，提高溫度，對於每個 frame 都產生 soft target，並且利用這些 target 當作 input 再訓練另一個 5 層 size 400 的 LSTM，且這次訓

練也利用 soft 的概念，一開始將 T 設定為 5，每個 epoch 逐漸降低 T，我稱為”Cooling Down (CD)”，根據網友及論文敘述，提高 T 在訓練過程可以達到 regularization 的效果，甚至跟 dropout 有差不多的表現。最終這種方法讓我的 performance 更加進步。

（二）實驗結果與設定

Model	**LSTM Hidden**	**mode**	**lr (1e-4)**	**epoch**	**Threshold**	**Valid**	**Kaggle (Public)**
LSTM	250x1	X	3	14	0	14.9811	15.4294
BiLSTM	512x1	concat	10	25	0	13.4649	14.8362
CNN+BiLSTM	512x2	concat	10	5	0	10.6378	11.7232
CNN+BiLSTM	256x4	concat	8	12	0	9.5027	10.3842
CNN+BiLSTM	256x4	concat	8	12	0.6	8.2135	8.8757
BiLSTM	200x7	concat	8	34	0	8.1459	9.2090
BiLSTM (fbank)	256x6	sum	7	35	0.8	9.58	8.3559
BiLSTM (fbank+mfcc)	350x6	ave	5	18	0.65	7.0833	8.3616
BiLSTM (fbank+mfcc)	350x6	sum	5	28	0.7	7.4	8.0791
BiLSTM (fbank+mfcc)	350x6	sum	5	34	0.8	7.05	8.3107
BiLSTM (fbank+mfcc)	400x6	sum	4	39	0.7	6.86	7.7288
BiLSTM_Soft	400x3	sum	4	8	0.85	6.76	7.7119
BiLSTM_Soft	400x5	sum	4	8	0.95	6.84	**7.5706**
BiLSTM_Soft Window(見下段)	400x5	sum	4	8	0.95	6.81	7.7401

使用 CNN 的 model 比起單純使用 LSTM 收斂速度快很多，大約十個 epoch 就有不錯的表現。Soft 的 model 因為算是 finetune，所以一開始幾個 epoch 就有良好的表現。此外實驗時間不夠，無法充分對於許多方法做比較，例如 feature 使用 fbank、mfcc 或是兩者結合何種最好、CNN 應該要建多深才有比較好的效果、雙向 LSTM 要用 concat 還是 sum 或是其他 mode 最好等比較。

（三）其他未成功或未實驗的想法

1. Window

我嘗試了一種我稱作”Window”的後處理方法(感謝 b03902034 劉浩然同學提供)。如同”Trimming”希望能使結果更加平滑，維持 frame 會重複出現一段的特性，”Window”法首先統計每段 frame 在 training data 裡的平均長度，然後將結果從頭開始，每個 frame 都展開上述統計的長度，判斷若是該 frame 不是這個 window 裡出現最多的 frame，就不要輸出這個 frame。此種方法結合”Trimming”之後並沒有變好，後來我想因為每段 frame 雖然作統計取平均長度，但在句子的不同部分，frame 段的長度相差很大，所以 window 可能會淘汰不該淘汰的 frame，尤其是當 model 對於每個 frame 的預測強度不夠時，會讓預測結果更糟糕。

2. padding

之前有思考過前面提到的切句子的方法的利弊，好處在可以縮短 timestep，雖然讓資料量增加，但是可以透過放入更多 batch 達到平行化，效率大幅增加，然而因為切句子時沒有考慮斷點的性質，雖然有 overlap 但還是可能會影響學習的結果，例如模型可能無法學會開頭或是結尾的特性，或是某些 frame 段與另外一些 frame 段的關係就學不出來。原本我有打算嘗試使用 sil 當作 padding 將每句補到長度 800，但是使用的機器有記憶體限制加上是稍舊的 GPU，訓練速度太慢所以最後放棄此種方法。

3. seq2seq

無論是”Trimming”或是”Window”都是 rule-based 的後處理方法，所以我想過訓練一個 sequence to sequence 的模型，輸入原先模型在刪掉重複的 frame 之前的輸出，學習如何產出最終的 sequence。然而不曉得是 model 是只有單層 256 的 LSTM 或是本身就不適合這樣使用，最終結果很差，edit distance 大約在 20 上下，與理想中的個位數有很大的差距，也由於時間不足無法做更深入的研究，例如加深模型的規模，或是使用更方便的套件跑更先進的模型。

4. Resnet

在 2015 年提出、圖像辨識領域中得過多項競賽冠軍的模型 Residual Network(簡稱 Resnet)也適合用在這次的作業裡，但由於記憶體限制，圖片不能合併太多，否則記憶體會裝不下，也因為 Resnet 很深的特性，訓練速度非常慢，沒有足夠多的時間做實驗。根據 http://cs231n.stanford.edu/reports/2017/pdfs/804.pdf 這篇 report，當中 Resnet 的結果與 4 層 1024 的 LSTM 結果相當。我想若能結合 Resnet 和 LSTM 一起訓練，定能達到更佳的結果。