HW1 report

b05902052 劉家維

模型描述

- RNN
  - 備註：以下模型若沒有特別注明，RNN model 指的都是 4 層雙向 LSTM 的版本。
  - 輸入參數維度：(800, 108)
    - 方便起見，先將所有序列 repeat 直到長度為 800，不足部份以末尾的 silence 的參數補齊。
    - 108 維是 fbank 的 69 維 + mfcc 的 39 維。
  - 模型層
    - 雙向的 LSTM 層，輸出維度 128 維，權重以常態分佈初始化
    - Dropout 層，比例=0.4
    - 雙向的 LSTM 層，輸出維度 128 維，權重以常態分佈初始化
    - Dropout 層，比例=0.2
    - 雙向的 LSTM 層，輸出維度 128 維，權重以常態分佈初始化
    - Dropout 層，比例=0.2
    - 雙向的 LSTM 層，輸出維度 128 維，權重以常態分佈初始化
    - Dropout 層，比例=0.2
    - 全連接 (Dense) 層，輸出維度 64 維，activation 函數=relu
    - Dropout 層，比例=0.2
    - 全連接 (Dense) 層，輸出維度 39 維，activation 函數=softmax
    - 損失函數：交叉熵損失函數，優化器=adam
- CNN + RNN
  - 輸入參數維度：(800, 108)
    - 同 RNN
  - 模型層
    - 一維卷積層，卷積核 64 個，時域窗長度=11，權重以 He 常態分佈初始化
    - Dropout 層，比例=0.2
    - 一維卷積層，卷積核 32 個，時域窗長度=11，權重以 He 常態分佈初始化
    - 雙向的 LSTM 層，輸出維度 128 維，權重以常態分佈初始化
    - Dropout 層，比例=0.4
    - 雙向的 LSTM 層，輸出維度 128 維，權重以常態分佈初始化
    - Dropout 層，比例=0.2
    - 雙向的 LSTM 層，輸出維度 128 維，權重以常態分佈初始化
    - Dropout 層，比例=0.2
    - 雙向的 LSTM 層，輸出維度 128 維，權重以常態分佈初始化
    - Dropout 層，比例=0.2
    - 全連接 (Dense) 層，輸出維度 64 維，activation 函數=relu
    - Dropout 層，比例=0.2

- 全連接 (Dense) 層，輸出維度 39 維，activation 函數=softmax
- 損失函數：交叉熵損失函數，優化器=adam

如何提升性能

- 描述模型與技巧
  - 在 trimming 的時候，一次考慮一個寬度為 5 的時域窗，若該時域窗中有 3 個以上的 frame 被預測為同個 phone 才輸出該 phone，最後再將連續的 phone 刪除。
  - 在得出最佳模型時，使用 3 個不同的模型（分別為 RNN, CNN+RNN, CNN+RNN+CNN），取這 3 個模型中每個 phone 被預測為不同分類的機率的乘積來做判斷。
- 使用的原因
  - 一開始在預測的時候，因為模型預測的準確率不佳，因此使用這方法過濾一些預測錯誤的雜訊。隨著模型的準確率愈來愈高，此方法依舊有效，只是時域窗的寬度和門檻也要隨之調整，讓過濾效果不要太強。

（寬度, 門檻）\ 準確率	60%	85%	95%
(2, 2) (naive)	-	-	**7.33734**
(5, 3)	**21.99759**	**9.60240**	**7.18072**
(7, 3)	**22.31084**	**9.67710**	**7.28433**
(7, 4)	**15.60000**	**11.06265**	**8.71566**

  - 這是參考 Ensemble 的作法。因為使用這方法已經是 deadline 當天，所以沒有真的去做 Ensemble (Boosting / Bagging)，而是直接拿 3 個不同模型的預測機率共同預測結果，但是也有小幅提升。

Model	RNN	CNN + RNN	CNN+RNN+CNN	合併前兩者	合併三者
Edit Distance	**7.90602**	**8.82409**	**10.63132**	**7.26746**	**7.18072**

實驗結果與設定

- 比較並分析 RNN 和 CNN 的結果
  - 以下模型皆跑到 acc 收斂，最多約 150 epochs

Model	RNN	CNN + RNN
Edit Distance	**7.90602**	**8.82409**

  - CNN + RNN 的組合沒有比想像中的好，甚至比純 RNN 差，感覺是因為 CNN 與 RNN 相比更容易過擬和的緣故，但還需要更進一步研究。
  - 感覺在 CNN 中池化層算是蠻重要的一環。只是因為這是一對一的標籤，池化會造成幀數縮減。我也有試過每隔 4 個 frame 抽取 1 個 frame 出來的方式讓有池化層版本的 CNN 訓練，但是效果不佳（Edit Distance : **13.92048**）。

        - 詳細作法是將 800 個 frame 分成 4 組： [::4], [1::4], [2::4], [3::4] ，這樣每個序列的長度都是 200 個 frame。然後再將這四組分別放進有持化層版本的 CNN+RNN 訓練 4 個模型，最後用這 4 個模型共同預測。
    - 比較可惜的一點是我沒有實作出二維卷積。因為同一個 frame 中相鄰的 feature 其實是相鄰的頻段，所以一個合理的作法是拿 fbank 和 mfcc 分別用二維卷積訓練出兩個模型，最後再合併。
    - 其他發現
        - RNN 在跑時與 CNN+RNN 相比慢了許多。CNN 在訓練時，顯卡使用率幾乎都在 80% 以上，每個 epoch 也只要約兩分鐘。RNN 在訓練時，顯卡使用率都在 20% ~ 50% 左右，每個 epoch 約需四分半。而且我的 CNN+RNN 是在 RNN 層之前多疊兩層 CNN，差距大到有點奇怪。不知是否是 tensorflow 較慢的關係，因為之前有聽說過 CNTK 在 RNN 方面遠比 tensorflow 快，既然 keras 都有支援，以後會再實驗看看。
- 比較並分析與其他模型的結果
    - 以下模型皆跑到 acc 收斂，最多約 150 epochs

Model	純 RNN 2 層	LSTM 2 層	雙向 LSTM 2 層	雙向 LSTM 4 層
Edit Distance	**22.31084**	**15.94216**	**10.78072**	**7.90602**

    - LSTM 相對於最基本的 RNN 來說差距甚大，因為 LSTM 可以記憶較長期的資訊。
    - 雙向的 LSTM 又比單向 LSTM 還要好很多，因為講話的聲音會同時被前後要發出的聲音所影響，不是只與前面的聲音有關。
    - 層數也是深度學習中影響準確率顯著的因素，但是訓練時間會被拉得很長。
    - 關於有沒有將參數 repeat 這件事情，原本以為擅自 repeat 會造成樣本偏差，但是就訓練結果來看沒有太大影響。