AI-Cup report

b07902042 葉璟諄 b07902052 鄭達詠

Goal

因為看到一篇與主題相關的論文，我們決定用「**Convolutional Neural Network** (**CNN**)」來進行training，希望利用CNN最大程度的學習 feature 與 groud truth 之間的關聯性，並在我們不清楚feature重要性排序的情況下利用 Neural Network 自行分配權重以及Convolutional Layer對輸入進行局部觀察的特性來達到篩選feature的目的。

具體作法是產生x_train(歌的feature)和y_train(這個frame為boundary的機率)，boundary是指onset或offset，把x_train，y_train的資料傳給CNN，建立出可以求出「**此frame是否為boundary機率**」的model。再對任兩個boundary之間的pitch求中位數，算出一個音的onset、offset跟pitch。

Dataset

因為「**boundary**」和「**非boundary**」的frame數量太過懸殊，若將所有資料直接餵給CNN，CNN會傾向將所有frame判斷為非boundary，判斷boundary的準確度高但預測是否有音的表現差，所以必須調整比例。用groundtruth產生y_train時，若這筆資料不是boundary，就用$\mathbf{P_{in}}$的機率去決定這個frame是否要加入。實測後發現$\mathbf{P_{in}}$=**1/12**的時候，兩者的比例大概是1:1，應該是比較理想的，至於x_train最初就只是把23個features加進去，用除以該feature最大值的方法來normalize，然後處理成CNN可接收的格式，還沒經過篩選。之後留下與已選的y_train相對應的frame作為最終的data set。

CNN model

CNN的inputlayer為一個23*5的矩陣，對應到目標frame及其前後兩個frame的23種feature。CNN中間總共有**兩層2D convolutional layer**，第一層的kernel長寬為5*3，第二層的則為3*3，node的數量皆為64。convolutional layer的輸出經過relu function整流後進入下一層，由於input長寬大小有限，我們省略了一般CNN常見的pool layer。之後進入**大小為64的dense layer**，一樣以relu整流後傳入最後的output layer，並以sigmoid function輸出 [0, 1] 之間的數值，代表目標frame是boundary的預測機率。Loss function採用的是常見於二分法的Binary Cross Entropy，優化則採用Adam Optimizer。我們採用tensorflow.keras實作。

```
model = Sequential()
model.add(Conv2D(64, (5, 3),  input_shape = (23, 5, 1)))
model.add(Activation('relu'))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))

model.add(Flatten())

model.add(Dense(64))
model.add(Activation('relu'))

model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

(圖一) CNN model實作

Post-processing

CNN model 訓練完成後，對每首歌進行預測，會得到得到長度為frame個數的陣列，代表此frame為boundary的機率，此時要設定一個門檻機率P_b，決定是否要把此frame視為boundary，實測後發現**P_b = 0.8**時表現最佳(如表一)，得到boundary list後用中位數求兩boundary之間frame的pitch，當作這個音的pitch，就會有 [**onset offset pitch**] 三個屬性。此時考慮到一個問題是，如果在計算中位數時，兩個boundary之間的frame數是偶數時，會把中間兩數平均，可能會將差異極大的兩值平均到，因此在資料中加入一個極大的值，使個數變為奇數，避免此狀況發生。

P_b	accuracy
0.1	0.198852
0.2	0.250936
0.3	0.300143
0.4	0.343917
0.5	0.390434
0.6	0.434538
0.7	0.472959
0.8	0.493032
0.9	0.450276

(表一) P_b對500首歌的accuracy

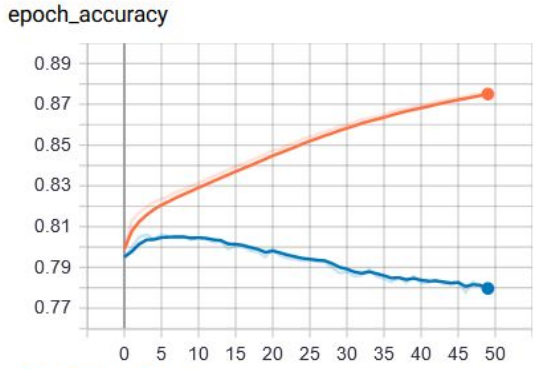
Feature selection

直接把23個features都餵進去，表現不太好，原本認為CNN會自己對feature的重要性進行評估，但把一些不重要的拿掉會更準。其中有12個features是chroma　vector，1個是chroma_std，主要的功能是**捕捉音樂的和聲和旋律特徵，以及音色樂器的變化**，對於人聲轉譜的來說相對不重要(只有vocal)，因此拿掉這13個features，表現會更好。

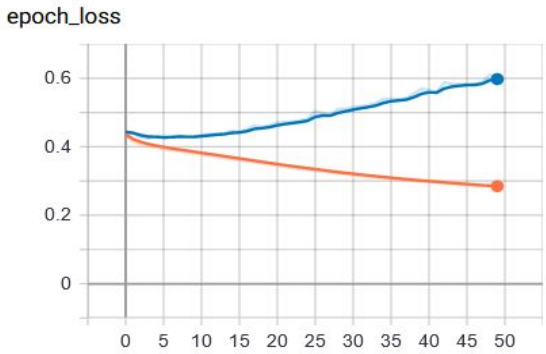
Improvement over the baseline method

1. CNN層數：多層的CNN複雜度較高，可能可以增加對資料的理解。但過高的層數也要避免overfitting的情形，尤其各種歌曲不能保證曲風相似。
2. Window大小：目前採用的方法是觀察目標frame及其前後兩段，共5個frame。在CNN增加複雜度的狀況下可以考慮增加window的大小，以不模糊目標frame為前提。
3. 改從原始音檔讀入，繪製波形：這是可能大幅增加準確度的改良方法。將input改為frame內的波型，output依舊為是boundary的可能性。CNN的專長為圖像處理與語音波形辨識，換成真正圖像式的input有助於CNN學習。原始檔案的讀取、波形分析、圖片轉換相對困難且仰賴大量空間與時間，依我們現有的計算能力難以達成。
4. Loss Function、Optimizer：使用Binary Cross Entropy作為loss function是適合二分法的計算方式，卻未必能最大化F-measure的評分數值。我們曾嘗試寫入類似F-measure的loss function，但相對應的optimizer卻相當難設計，直接採用一般的optimizer無法與loss值呼應，幾乎無學習效果。
5. Batch Size、Epoch、Dense layer等等CNN相關參數：如教授上課所說，neural network的學習方法目前依舊未知，各種參數對學習效果都可能影響甚大。這種狀況下要改善學習表現相當不穩定，只能改變個別參數觀察。

Result



(表二) Epoch Accuracy



(表三) Epoch Loss

表一表二分別為accuracy與loss隨epoch的變化圖，橫軸為epoch數，縱軸分別為accuracy跟loss。其中橘色折線代表training set，藍色折線則代表validation set，兩者比例為7:3。可以看到橘色的accuracy隨epoch上升，loss隨epoch下降，model對於training set仍有學習空間。藍色的accuracy在10 epoch後有明顯下降的趨勢，loss也開始增加，代表可能發生over fitting。但根據我們實測結果發現，10 epoch之後的model仍然在Public set的表現有所進步。

以上述方法在Public Set得到的最佳的accuracy為0.217。

Insight and conclusions

實驗結果發現performance沒有顯著上升，甚至是下降，推測是CNN並沒有那麼適合人聲轉譜，比較適合做圖片處理。改採用DP或是條件判斷較容易理解原理並優化模型。若要使用neural network且不改變輸入輸出的型態的話，此題目應該使用RNN類型的模型，比較好針對歌的feature和資料性質去做預測，但礙於時間問題並沒有再嘗試。

Division of labor

葉璟諄：協助產生dataset與建立model，並實測結果。

鄭達詠：設計實作方法與model優化。

Reference

[1] Tim O'Brien. MUSICAL STRUCTURE SEGMENTATION WITH CONVOLUTIONAL NEURAL NETWORKS. http://cs231n.stanford.edu/reports/2016/pdfs/220_Report.pdf

[2] Feature Extraction https://github.com/tyiannak/pyAudioAnalysis/wiki/3.-Feature-Extraction?fbclid=IwAR1XRCcCUQGU1CVATBnf7N1s5NWgCXXZg9SJ7AQteg5YyglT4slfkh6L7mI

[3] Deep learning with python,tensorflow,and keras tutorial https://www.youtube.com/watch?v=wQ8BIBpya2k&list=PLQVvvaa0QuDfhTox0AjmQ6tvTgMBZBEXN