Topic : Sound recognition　Team : NTU_b05902043_不要鯨魚

1.Introduction and Motivation

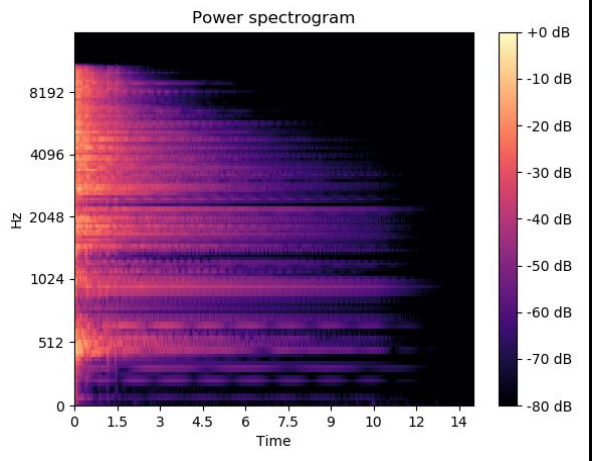
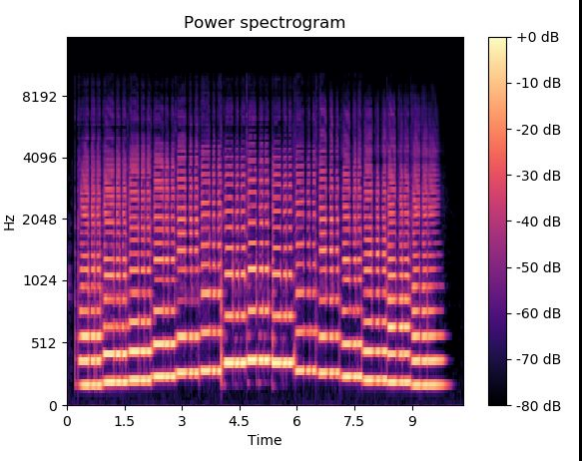
這次的報告是實做一個41種類的聲音分辨器，因為音檔的大小很大，所以不能直接用讀進來的np array來訓練模型，一定要經過化簡的步驟才行。這次我們選擇了不同的兩種作法來實現這件事。化簡完後在接上NN, CNN model 來進行預測。

會做這個題目主要是因為之前上別門課的時候有人做了一個聲音辨識的題目，當時就驚為天人，想說一定要做一個類似的東西。這次剛好有這個機會，就選擇聲音分析了。

2.Method one

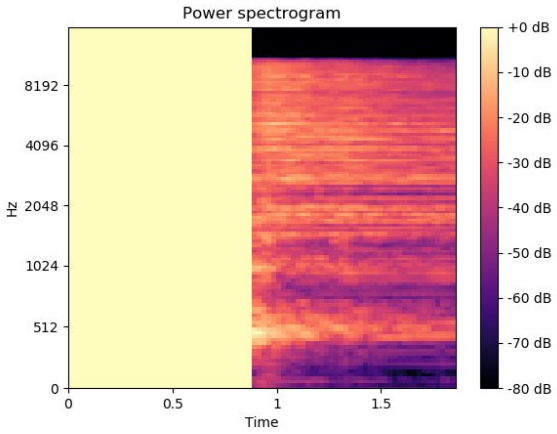
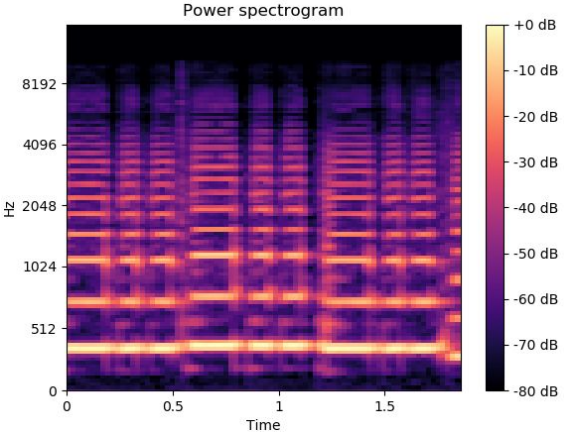
這個化簡的作法是將聲音的檔案讀進程式中後將其轉成spectrogram[1]，然後在利用RNN network來尋找不同種類的聲音的features。這個想法其實挺簡單的，不過實作的過程中還是遇到了一些問題。

在產生spectrogram的部份，我們使用了一個名叫“librosa”的軟件，這個軟件的功能挺強大的，只要呼叫一個函數就能產生出spectrogram了。音檔轉的圖檔大致上就像下面所示:

Hi-hat	Saxophone
	

就如同上圖所示，我們在實際操作上忽略了高於8704的頻率，因為在轉檔的過程中我們發現通常更高頻率通常都沒有什麼東西。除此之外，我們也將振幅的顯示方式換成dB，否則差距太大，只會有幾個亮點而已。

而問題就出在於圖像的長度上了，可以明顯看出圖像的長度跟音檔的長度是一樣的，所以每個圖像都不一樣大。但是CNN model的輸入是要一樣大小的。為了解決這個問題，我們決定選擇整張圖中最亮的那個點(能量最高的地方)，擷取那裡的前後40個數據點(大約2秒)，而如果擷取的區段超出了原來音頻的長度，就全部填零。擷取完的圖像大致長的像:

Hi-hat	Saxophone
	

這樣轉換完的資料就夠小了，每個音頻都轉換成只有128x80個數字。因此，我們就開始訓練CNN model了。

CNN structure

Layer (type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 80, 128, 48)	480
conv2d_11 (Conv2D)	(None, 80, 128, 48)	20784
conv2d_12 (Conv2D)	(None, 80, 128, 48)	20784
max_pooling2d_4	(None, 40, 64, 48)	0
conv2d_13 (Conv2D)	(None, 40, 64, 96)	41568
conv2d_14 (Conv2D)	(None, 40, 64, 96)	83040
conv2d_15 (Conv2D)	(None, 40, 64, 96)	83040
max_pooling2d_5	(None, 20, 32, 96)	0
conv2d_16 (Conv2D)	(None, 20, 32, 192)	166080
conv2d_17 (Conv2D)	(None, 20, 32, 192)	331968
Max_pooling2d_6	(None, 10, 16, 192)	0
flatten_2 (Flatten)	(None, 30720)	0
dense_4 (Dense)	(None, 1024)	31458304
batch_normalization_3	(None, 1024)	4096
dense_5 (Dense)	(None, 512)	524800
batch_normalization_4	(None, 512)	2048
dense_6 (Dense)	(None, 41)	21033

最一開始的時候我們想要嘗試使用LSTM layer 作為我們 model 的主架構。因為感覺上聲音就跟時間蠻有關系的阿，像是倒著放影片我們就聽不懂了。然而，實際上LSTM的表現並不好，他最高也只做到0.58(accuracy for only one prediction)，而正

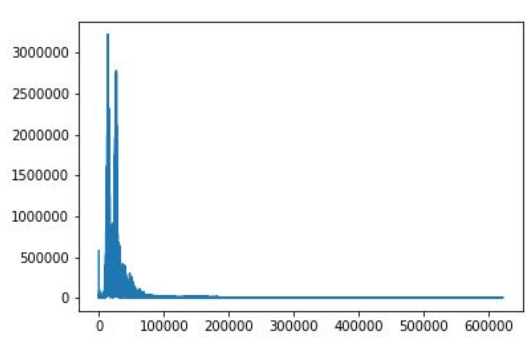
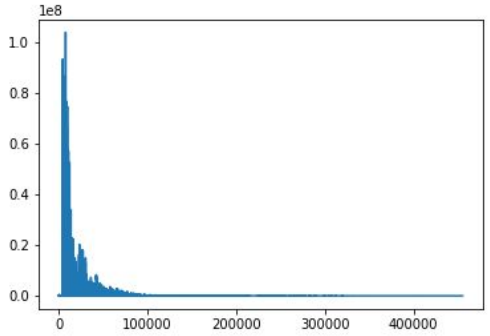
常的RNN能做到0.69。我們為了這個現象找了一些解釋的方式。首先，這是的分類的問題，根據我們個經驗，我們將狗叫聲/笑聲切成小段再隨機組裝回去，我們還是能辨識出聲音的種類的。其次，關於時間的資訊其實很大一部分都跟相位很有關系，而相位的資訊並沒有包含在spectrogram中，所以，使用LSTM其實只會複雜化模型增加訓練的困難度而已。

而進行過一段時間的測試後，我們發現上圖的結構是最好的。不過由於一些未知的原因，這種實做方式中的˙overfitting現象十分嚴重。在所有我們嘗試過得結構中，只要是variation acc超過0.60的，其訓練資料的準確度都會達到1。這兩個數值差大約0.4，顯示出有強烈的overfitting存在，但是沒有方式解決。

我們嘗試過在模型的 layer 1中添加Dropout，不過這樣做的結果要不是訓練資料的準確度也降到0.6左右要不就是完全訓練不起來。我們也嘗試過移出一些layer來降低複雜度，不過這樣的結果並沒有好轉，反而還稍稍下降了。

當我們做完spectrogram後，我們稍微嘗試了一下fft。主要就是直接將檔案丟去轉換，在將轉換完前1000個峰值的頻率與相位記錄下來作為訓練的資料丟入一個NN model 來進行預測。不過這並沒有成功。

我們分析了一下大概有下面幾種理由。首先，有一些聲音的fft結果很像，如下圖

Bark	Saxophone
	

不過這應該不是什麼大問題，因為像是振幅大小跟相位還是有差距的。最大的問題是在於樣本(labeled)的數量不足，有些種類的聲音的樣本數量真的很少，只有幾個。在這種情況下無法train起來感覺還蠻合理的。

3. Method Two: MFCC & 1d-conv

接著我們在 kaggle 上看到了一個 kernel 達到了 0.895 的分數[2]，因此我們試圖參考其架構並在最後 ensemble 起來。

他的架構是由兩個 model ensemble 起來的，分為 1d-convolution 及 MFCC 的部份。

首先是 1d-convolution，不像之前我們 preprocessing 先化為 spectrogram，在這個模型中，直接使用 raw data 做一維的 CNN。

Data preprocessing: resample 成 16000 後取出前兩秒。
Model : 4 層 CNN-layer 及 2 層 fully connected layer

Validation accuracy: 0.6047

第二種 model 是利用 MFCC (Mel-Frequency Cepstral Coefficients) 來做 data prerpocessing，可以篩出人耳較為敏感的頻率。

Data preprocessing: 取出前兩秒後以 librosa.feature.mfcc 進行 preprocessing
Model: 同樣為 4 層 CNN-layer 及 2 層 fully connected layer

Validation accuracy: 0.7150

實驗: 我們試了
(1) 加上 gaussian noise
(2) semi-supervised learning(將 test data 預測超過門檻值的加入 train data)
(3) 取出音量最大的二秒(因為我們觀察到大約有 2~3% 的音檔在開頭只有雜音)。

	validation accuracy for only one prediction
original	0.7150
Gaussian noise	0.7120
Semi-supervised	0.6940
Max volume	0.7040

可以看到幾乎都沒有什麼作用。尤其令人意外的是取出最大音量的部份。因為我們觀察到大約有 2~3 % 的數據在開頭是幾乎沒有聲音的，但取出最大音量的部份卻沒有變好，我們推測或許是因為其實有最大的音量不一定具有代表性。

最後我們將 dataset 隨機分成 10 等分並做 cross validation，最後再將 10 個 model 做 ensemble，在 kaggle 上的 public score 為 0.833 。

將 1d-conv 及 MFCC 做 ensemble，在 kaggle 上的成績為 0.883

4. Result

最後我們將三個方法做 ensemble 後，在 kaggle 上的分數為 0.914。而我們在最後的時候也嘗試看了一下model做出來信心不太高的音檔。結果發現其中有一些真的難以辨識，像是一個人打噴嚏跟笑，我們試聽的結果也不太能確定。

5. Conclusion

這次我們的實做方向偏向於頻譜分析，mfcc和spectrogram都是以頻率作為feature再來做分析。個人是認為這樣的成果還算是不錯，在單一預測的情況下準確率大概是0.8左右，考慮到有許多我們都難以辨識與太短的音檔，這樣的結果是能接受的。除此之外，我們一直沒有成功的實做出將phasor也納入考量的模型來，這或許是一個接下來能提高準確率的嘗試。

這次實做中ensemble其實也發揮了蠻大的作用，單一模型的上限就是0.83(kaggle)上下，而這些model合起來能達到0.91(kaggle)其實也挺厲害的。

6. Reference

[1]: https://librosa.github.io/librosa/generated/librosa.feature.melspectrogram.html
[2]: Beginner's Guide to Audio Data (https://www.kaggle.com/fizzbuzz/beginner-s-guide-to-audio-data)