ML2017FALL Final Project: Sound Classification

NTU_r06921058_95度的顯卡

R06921058 方浩宇

B04501127 凌于凱

D06921025 王瀚緯

R06942119 林宗憲

1. Introduction & Motivation

Data set總共有41種Label，主要都是非人聲的聲音檔，在train set中有分成3700個manual verified，以及5800個non verified，Test data 中有9400筆資料。這次題目的目標是要分類出不同的Label，每一筆Data可以Output出三種最高可能性的Label作為結果，如下圖。

fname	label
00063640.wav	Shatter Scissors Keys_jangling
0013a1db.wav	Flute Trumpet Violin_or_fiddle
002bb878.wav	Bass_drum Snare_drum Knock
002d392d.wav	Bass_drum Telephone Electric_piano

我們最後final project選擇這個題目的原因是因為，在hw3以及hw6已經有試過對於圖片以及語詞做學習，但是還沒有嘗試過對於聲音做處理，所以我們想透過這次final project學習到利用機器學習方式來處理聲音檔案的分類。

2. Data Preprocessing/Feature Engineering

I. Mel Spectrogram

這次資料集中的41種聲音檔案，並不是只有人類的聲音，參考過去的別人的資料，使用Mel Spectrogram而非MFCC的話可以在分辨非人類的聲音上有更好的表現，因此我們最後採用Mel Spectrogram來做為提取聲音特徵的方式。我們提取特徵的參數為

Band=64,Hop=160,fmin=20,fmax=16000,nfft=400,samplerate=44100

II. 統一長度(unify length)

這次資料集中的聲音長度都不一定相同，從最短幾秒到最長幾分鐘，若是要使用CNN進行訓練的話需要將長度統一才行。我們決定統一的長度方法是將全部

音檔轉換為Mel Spectrogram的形式之後統計它的長度，然後選擇最接近中位數的2的次方項(最後是使用512)

III. 切割/重複資料

我們統一長度的方法是將長的資料切成數段512長度的資料(EX：若有一個長度為1124的音訊，會切成三份，第一份為0~511，第二份為512~1023，第三份為612~1124)；將短的資料使用Repeat的方式padding到512的長度。

IV. Unsupervised Learning

在訓練出有一定準確率(我們是使用kaggle分數0.91)的model之後，使用這model predict Testdata，保留較有信心的資料(我們設定是預測中最高可能性和第二高可能性的機率相差3倍以上)，然後再將原本用來訓練的資料初沒有經過人工驗證的部分(品質較差的資料)去除，最後用原本資料中有驗證的部分和有信心的Testdata來進行訓練。

V. Image Generator

由於特徵處理的結果是一個2維的圖片，因此我們使用了Image Generator來進行Data Augmentation。我們使用了平移和垂直移動，標準化，Random Erasor(可以隨機消除一張圖片中的一部分)。

3. Model Description (At least two different models)

在這次期末專題中，我們嘗試了不少不同的NN，例如1DCNN、普通的簡易CNN、VGG、AlexNet、CRNN，最後留取了兩種訓練結果最好的Model：CRNN和VGG。兩種Model的架構和參數如下表。兩種model compile時都使用categorical_crossentropy作為Loss Function，Optimizer為Adam

CRNN		VGG	
Layer	Param	Layer	Param
Conv2D	64,kernel_size=(2,2)	Conv2D	64,kernel_size=(2,2)
BatchNormalization		BatchNormalization	
LeakyRelu	alpha=0.1	LeakyRelu	alpha=0.1
MaxPooling2D	poolsize=(3,2)	MaxPooling2D	poolsize=(2,2)
DropOut	0.1	DropOut	0.1
Conv2D	128,kernel_size=(2,2)	Conv2D	128,kernel_size=(2,2)
BatchNormalization		BatchNormalization	

LeakyRelu	alpha=0.1	Relu	
MaxPooling2D	poolsize=(3,2)	MaxPooling2D	poolsize=(1,2)
DropOut	0.1	DropOut	0.15
Conv2D	256,kernel_size=(2,2)	Conv2D	256,kernel_size=(2,2)
BatchNormalization		BatchNormalization	
LeakyRelu	alpha=0.1	Relu	
MaxPooling2D	poolsize=(2,2)	MaxPooling2D	poolsize=(2,2)
DropOut	0.1	DropOut	0.2
Conv2D	256,kernel_size=(2,2)	Conv2D	512,kernel_size=(2,2)
BatchNormalization		BatchNormalization	
LeakyRelu	alpha=0.1	Relu	
MaxPooling2D	poolsize=(2,2)	MaxPooling2D	poolsize=(2,2)
DropOut	0.1	DropOut	0.25
Conv2D	256,kernel_size=(2,2)	Conv2D	65124,kernel_size=(2,2)
BatchNormalization		BatchNormalization	
LeakyRelu	alpha=0.1	Relu	
MaxPooling2D	poolsize=(2,2)	DropOut	0.3
DropOut	0.1	Flatten	
Reshape	32,-1	Dense	512
BidirectionalLSTM	256	BatchNormalization	
BidirectionalLSTM	256	Relu	
LSTM	41,softmax	DropOut	0.3
		Dense	512
		BatchNormalization	
		Relu	

		DropOut	0.3
		Dense	41,softmax

I. VGG

一個非常知名的CNN架構，主要架構分為幾點：

A. 越長越好，原本用5,5的filter就改成用數個2,2的filter取代

B. 每兩層CNN就加倍filter數量

C. 每兩層CNN的部分就進行一次MaxPooling(2,2)

D. 逐漸增加的Dropout

E. 最後全連結層用數個較大的Dense連接

我們有稍微做一些簡單的調整，主要是讓架構稍微輕量化一點以減少訓練時間。

II. CRNN(Convolution Recurrent Neural Network)

上表左方為CRNN的架構，可以發現CRNN前半部的部分和VGG的架構非常相似，差別部分在於原本全連接層的部分改成使用RNN。

CRNN的重點就在於把原本CNN的全連階層使用RNN取代，主要是使用在聲音分析上面。由於聲音所轉換出來的頻譜圖同時具有圖片以及時間序列資料的特性，一段聲音中連續的幾個時間點之間的資料是有根據時間的關聯性的。因此在前面的CNN部分相當於取出一段轉換為頻譜圖的音訊中每一個時間點的Embedding Matrix，而我們也發現Maxpooling如果包括時間軸(不只使用(2,1)這類只對同一時間點的資料進行特徵放大，也有使用(2,2)的大小來對時間軸進行特徵的放大)會對結果有比較好的影響。再進行完CNN的部份時，會將原本band的維度從64維縮小成1維，Reshape之後使用LSTM進行訓練，在最後也和一般使用Dense作為輸出層的做法不同，改成使用LSTM作為輸出。

我們最後的結果是使用VGG的架構進行訓練並且Ensemble，主要原因是因為CRNN在訓練時會遇到Accuracy突然下降的問題，以及儲存參數之後載入繼續訓練會出錯，因此最後使用VGG的架構訓練。

4. Experiment and Discussion

I. Simple Baseline

一開始過simple baseline的model是將有人工確認label的聲音檔轉成MFCC（梅爾頻率倒譜係數），並將41個類別轉成one hot表達，然後餵進CNN去train，實驗結果MFCC的長度並非越長越好，可能是有些聲音檔案較短，padding後特徵會不見，另外CNN的model不用太複雜即可有還ok的結果，此時影響train的時間長短的因素是training data的量、MFCC長度、CNN模型的複雜度。

在以上過程發現audio_test有三個檔案無法播放（解決方式是忽略這三個檔案），還有使用未經人工確認的聲音檔來train結果比較不好，於是分兩階段做，第一階段先用有人工確認label的聲音檔train完後，拿那些未經人工確認的聲音檔來預測，如果預測出來的結果機率最高的符合它的label，則將它加入已人工確認的label，然後將這些training data再來重新train一遍，則可以得到比之前較好的結果。

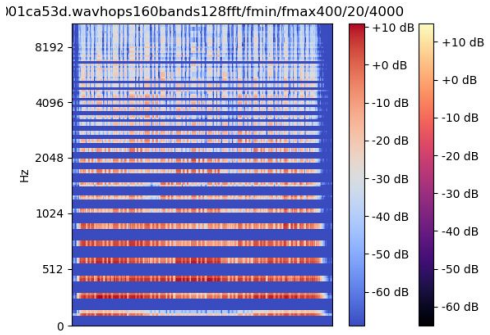
II. Strong Baseline

接下來的strong baseline也是利用將音訊檔轉換為MFCC，並且利用CNN的方式去train，原本是想用VGG的架構，但因為記憶體不足，所以就將模型架構做了一些修改，最後跑出來單model在kaggle的分數是0.795，比起之前的model有了一些進步；之後就以這個model當基礎，將原本的MFCC透過image generator去做標準化以及平移，透過bagging的方式，利用KFold產生10組training set，最後再把這10個model的output去做平均得到最後的答案，在kaggle上的分數為0.823。

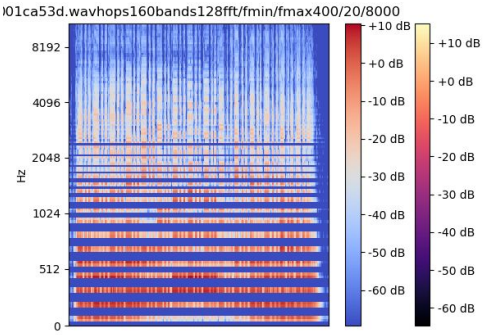
III. Final Ranking

最後是衝排名階段，我們參考網路上的資料，發現Mel Spectrogram的方式在這次的辨識中可能會有比較好的表現，因此我們改成將音訊檔轉換為Mel Spectrogram。接下來我們發現原本Unify Length的方式(保留中央長度等於目標長度的資料，刪除多餘的資料，若長度小於Unify Length則Pad Zero)會遺失掉太多可用的資訊，因此我們改成將資料切割成數段長度等同於Unify Length的資料，若資料長度小於Unify Length則用Repeat的方式Pad，而Unify Length則暫時設定為128。另外，在Test Data中也要做同樣的處理，而若是一個Data被分成數小段，則Predict時是分別把這幾個段落做Predict之後把結果相加。這時候Kaggle分數到達0.86左右。

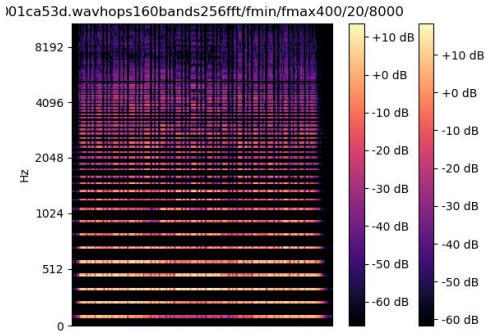
接下來我們重新嘗試了不同的轉換頻譜圖的參數並且將圖片儲存起來比較，如下圖。在經過幾次簡單的Model訓練之後，我們發現如圖四這種沒有黑條的比起其他種會更容易學習到東西。以及在比較Sample Rate時發現保留越多資訊也能夠幫助Model學習，因此我們Sample Rate從24000改成用44100來取樣。重新調整這些參數之後Kaggle分數到達0.89左右。



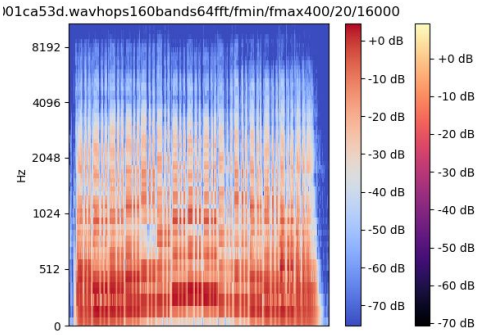
圖一



圖二



圖三



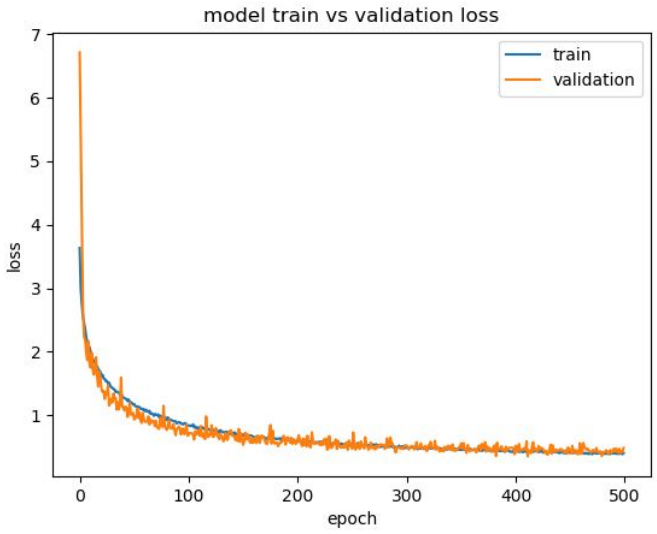
圖四

然後根據前段的結果，我們想說既然保留越多資訊能夠幫助模型訓練，那Unify Length如果設定長一點應該也能幫助訓練的結果，因此我們測試了256、512、1024等等參數。但是參數超過2048時即使Batch size只設定為8仍然會發生Memory Error，而1024訓練時間非常久(大約512的3~5倍)，因此我們最後採用512作為Unify Length的設定。改完之後單一Model的準確率到達0.906。

接著我們將訓練了2000個Epoch之後的Model進行訓練，並且每兩個Epoch儲存一次Model，用這種方法產生30個Model之後進行Ensemble(單純將全部結果相加並平均)。Ensemble之後Kaggle分數到達了0.915。

最後，我們使用前面Ensemble的Model所Predict出的Test Data中較有信心的部分(預測最高可能性與次高可能性的Label機率相差4倍以上)加入Training Data中訓練，並且移除掉Training Data中沒有經過人工驗證的資料，最後所得到的結果在經過Ensemble之後Kaggle分數0.928。

IV. 訓練過程



圖為VGG Model訓練前500Epoch時的Loss變化，從圖中可以發現經過最初收斂較快的50個Epoch之後，收斂速度變得相當慢，而且經過500個Epoch之後還沒有發生OverFitting的情況，我們後面將Model持續訓練，大約在超過1800個Epoch之後才會發生輕微的Overfitting的情況。

V. 訓練時間

一開始在訓練Simple Baseline與Strong Baseline的Model因為較為輕量，一個Epoch大約100秒左右，在100Epoch時就可以收斂。

之後使用的CRNN Model，由於修改過切割資料的方法，以及Model較為大型，一個Epoch大約要200秒，而500Epoch左右也不太容易收斂，且Loss會持續上下震盪。

最後使用的VGG Model，訓練一個Epoch需要240秒左右，大約在1800Epoch之後才會收斂。

5. Conclusion

I. 保存更多的features，能夠幫助Model學習的更好。因此，像是Sample Rate 44100在這次的結果上會比24000更好，而切割的長度長一點也會更好，因為這能保留Data的長度這個features。

II. Image Generator在這次題目中仍然非常有用，平移，標準化，以及隨機消除這幾個方法都能夠大幅度的降低Training和Validation之間的Accuracy差異。

III. 在最後預測的時候，比起利用訓練過程中所留下的Model，保留最佳的Model並且利用他持續訓練並且每個Epoch都儲存來產生更多的Model去Ensemble可以得到更好的結果。

IV. 由於整個題目的資料量非常龐大，Model也十分厚重，保留所有中間產物能夠幫助在Memory Error或者其他原因中斷時恢復上次執行的進度，也能省下非常多的時間。

V. Unsupervise Learning在這次題目中很實用，即使是使用Testing Data作為Unlabel的資料加入Training，也能夠得到很好的結果。

VI. Data的品質非常重要，由於官方提供的Unverify Data的Accuracy只有大約60~70%，因此以這Data所Train出來的Model在到達一定程度之後就會遇到瓶頸，我們是最後將這些資料移除，再加入Label後有信心的資料來避免Unbalance的問題，最後也的確能夠提升結果的準確度。

6. Reference

1. Building a Dead Simple Speech Recognition Engine using ConvNet in Keras,https://blog.manash.me/building-a-dead-simple-word-recognition-engine-using-convnet-in-keras-25e72c19c12b
2. https://github.com/daisukelab
3. Very Deep Convolutional Networks for Large-Scale Image Recognition,Karen Simonyan, Andrew Zisserman ,4 Sep 2014,arXiv Computer Vision and Pattern Recognition (cs.CV)
4. Convolutional Recurrent Neural Networks for Music Classification,Keunwoo Choi, George Fazekas, Mark Sandler, Kyunghyun Cho,Neural and Evolutionary Computing (cs.NE); Machine Learning (cs.LG); Multimedia (cs.MM); Sound (cs.SD)