

Machine Learning Final Project Report

Freesound General-Purpose Audio Tagging Challenge

NTU_b04901009_正妹隊長

b04901009 林孟瑾 b04901058 鍾興寰 b04901164 莫絲羽 b05901058 陳帝宇

2018.07.05

1. Introduction & Motivation

在這個題目中，我們需要建立一個自動標記音檔類別的系統，目標是標記出測試音檔最有可能對應的三種類別。約為9500筆的訓練資料音檔包含了人聲、樂器、動物以及一些生活中所接觸到的聲音，其中61.05%的資料是沒有被人工確認的，剩下的38.95%則為經過人工確認，且他們的長度不一定相同，從300毫秒到30秒皆有分佈。每個音檔都歸類於41個類別的其中之一，然而，這些音檔為不均勻的分佈，一個類別中含有的訓練資料最少有94筆，最多有300筆。

在建立標記類別的模型前，我們觀察音檔資料，決定對資料做預先處理，包括長度的裁減、資料的轉換等等。另外，由於是分類的問題，我們採用CNN模型的架構來訓練，並有使用semi-supervising去對model做retrain的動作，最後將許多不同參數的模型統整起來(ensemble)對測試音檔做預測，以提升準確度，降低過擬合的情況。

2. Data Preprocessing/Feature Engineering

(1)MFCC

我們首先嘗試了第一種方法：對音檔做傅立葉變換，再將所得頻譜通過梅爾濾波器。所採用的套件亦為librosa，使用其中的librosa.feature.mfcc函數。音檔經過轉換所得的頻譜維度為 $40 \times t$ ， t 與時間長短有關，由於training時，同一個mini batch中的形狀必須是固定的，因此將頻譜padding至 40×1300 的大小，方法如圖所示：

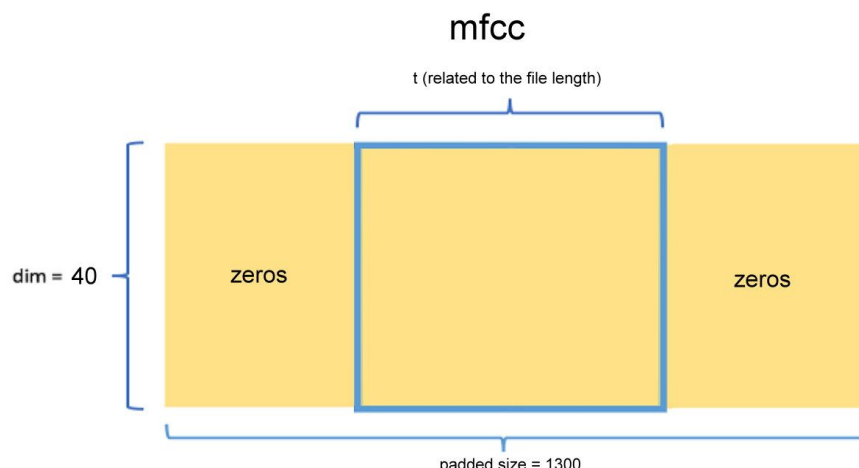


Fig. 1 mfcc data preprocessing

將音訊的前後皆補上相同長度的0，相當於補上相同長度的靜音片段。不過由於本次比賽提供的音檔長短差距過大(30毫秒~300秒皆有分佈)，padding的0造成分辨的困難，預測結果不甚理想，因此改用下列的資料預處理方法。

(2) Amvidia

我們實際聽一些音訊資料，發現其中有許多音訊的尾部是靜音的。為了不讓這些靜音的部分影響預測，我們使用Amvidia的MP3-Normalizer處理這些音檔，包括：

- A. 切除首尾靜音。
- B. 調整所有音檔的音量到一致。

(3) Mel-spectrogram

處理過這些音檔之後，我們可以開始對音檔進行short-time fourier transform，轉成一個mel-scaled的頻譜。我們採用的套件為 librosa當中的 librosa.feature.melspectrogram。音檔經過轉換得到的頻譜維度為 $128 \times t$ ， t 由音檔的長度決定。由於之後使用的模型只能吃固定形狀的輸入，我們在 preprocess 時將頻譜裁切成固定的大小。裁切方式如下圖所示：

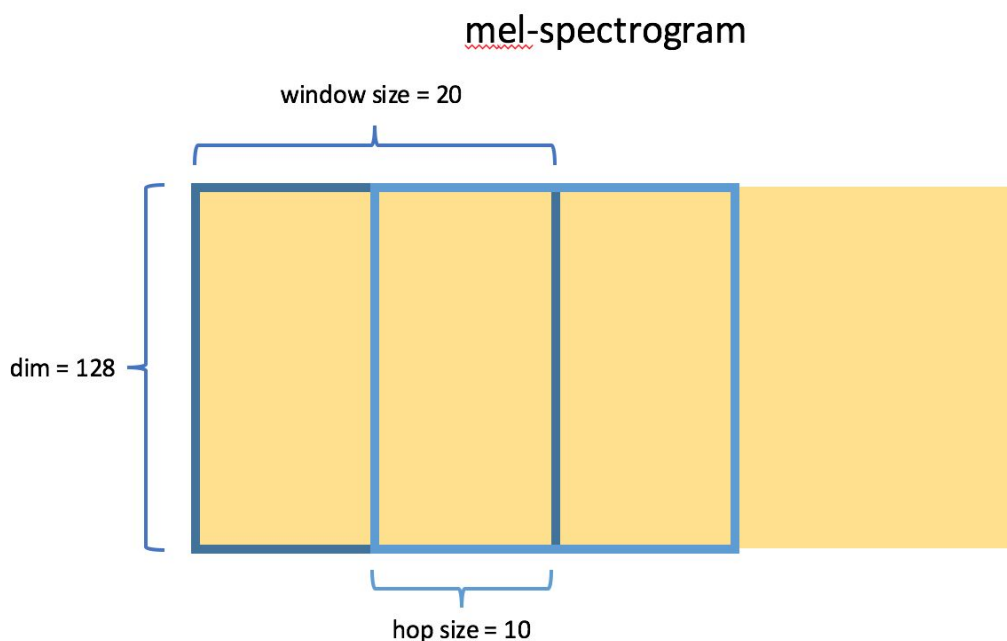


Fig. 2 mel-spectrogram data preprocessing

我們從每一個長度不一的 mel-spectrogram 中裁切長度 20 的片段，再維持同樣的 window size 右移 10 個單位裁切下一段音訊。本次比賽提供的某些音檔轉成 mel-spectrogram 長度不及20，我們會複製他們接到後面以延長

spectrogram 的長度，再進行上述的裁切過程。

(4) one-hot encoding

音檔對應的label部分，我們將個別の種類資訊轉換成one-hot vector。如果使用integer-encoding來處理類別的資訊，模型會以為不同類別之間存在著順序性，可能導致預測結果不佳且不符合期望。而使用one-hot encoding，便能去除類別之間的順序性。

(5) 過濾non-verified資料

除此之外，比賽主辦單位提供的音訊資料還分為 verified 和 non-verified 兩類。主辦單位表示，verified 的資料有經過人工驗證，non-verified 則無。其中 non-verified 資料估計約有 65% 到 70% 的機率是正確的。在後續訓練模型時若要提升表現，勢必不能只用 verified 過的資料，也不能把所有 non-verified 的資料都用進去，因此我們必須對 non-verified 的資料進行過濾，留下可信度比較高的資料提供模型學習。我們過濾資料的方法是先利用 verified 的資料訓練一個模型，再利用這個模型去預測 non-verified 的資料。假設 non-verified 的 label 和我們預測出的結果一致的話，我們就保留這筆 non-verified 資料。這些過濾後的 non-verified 資料能夠幫助模型的準度更上層樓。

3. Model Description

我們使用Keras作為建立預測模型的API。進行訓練前，將聲音由單一維度轉換成二維的spectrogram，所以模型主要的層為影像分類時所使用到的二維卷積層(Conv2D)。以下分別為我們嘗試的兩種訓練模型；其中，以VGG-16所建立之CNN模型，加上使用未經人工確認的資料去做semi-supervising的retraining，在Kaggle上取得較高的準確率。

(1) CNN (training data: only verified data)

輸入資料的大小為(128,10,1)，對應到音檔的window大小為10的訓練資料。此類模型所使用的訓練資料不包含未經人工確認(non-verified)的音檔，所以準確度低於往後所述的第二種模型。以下是模型堆疊的情況：

層	相關參數
Input	input_shape = (128,10,1)
Conv2D	卷積核數目 = 64，卷積核大小 = (10,3)
PReLU	激活層
BatchNormalization	axis = -1
MaxPooling2D	pool_size = (2,2)

Dropout	rate = 0.2
Conv2D	卷積核數目 = 128, 卷積核大小 = (10,3)
PReLU	激活層
BatchNormalization	axis = -1
MaxPooling2D	pool_size = (2,2)
Dropout	rate = 0.3
Conv2D	卷積核數目 = 256, 卷積核大小 = (10,3)
PReLU	激活層
BatchNormalization	axis = -1
MaxPooling2D	pool_size = (2,2)
Dropout	rate = 0.35
Conv2D	卷積核數目 = 512, 卷積核大小 = (10,3)
PReLU	激活層
BatchNormalization	axis = -1
Dropout	rate = 0.3

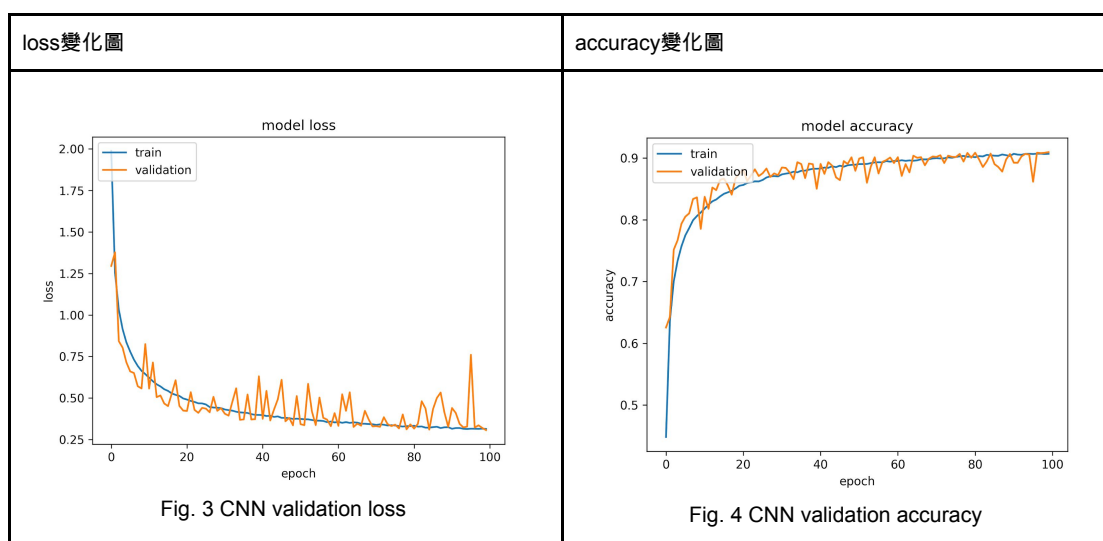
經過以上這些層後，再經過一層Flatten層，接著最後進入以下這些層：

層	相關參數
Dense	units = 256
PReLU	激活層
BatchNormalization	axis = -1
Dropout	rate = 0.25
Dense	units = 100
PReLU	激活層
BatchNormalization	axis = -1
Dropout	rate = 0.25
Dense	units = 41 (共有41個類別)

以下為模型參數的數量：

Total parameters	7444165
Trainable parameters	7441533
non-trainable parameters	2632

以下則為模型訓練時的loss和accuracy的變化圖：



其中，為了使測試資料分佈平均，避免訓練時過度擬合的狀況發生，隨機生成一段長度為訓練資料的排列的陣列，再將訓練資料重新排列，如此一來訓練的效果變好許多。

```
np.random.seed(1000)
index = np.random.permutation(len(train_X0))
train_X0 = train_X0[index]
```

Fig. 5 data shuffling

另外，因為單一個模型的預測效果普通，我們嘗試了將14個模型ensemble，把每個模型預測出的數值加總取平均後，選出前三大預測值的類別，預測效果比單一模型進步許多，但有其極限，多訓練了許多類似但不同參數的模型，準確度依然無法再往上提升，後來決定加入VGG-16和未經人工確認的資料來做訓練。

(2) CNN with VGG-16 (training data: verified data + non-verified data)

輸入資料的大小為(128,20,1)，對應到音檔的window大小為20的訓練資料，音檔與音檔之間的hop size為10，意思就是每兩個音檔之間會有長度為10

的overlapping。

模型使用Karen Simonyan, Andrew Zisserman在2014年發表的論文《Very Deep Convolutional Networks for Large-Scale Image Recognition》裡頭所使用的VGG-16模型，專門拿來做圖片辨識與分類。以VGG-16為基礎，訓練出多個參數各與彼此有些微差異的模型後，再將每個model拿出來，以原來verified的資料加上過慮後的non-verified資料，繼續做semi-supervised的參數訓練。以我們的經驗來看，大概只能再多訓練一、兩個epoch，再往後accuracy也不會再提升了。值得一提的是，accuracy往往在第一個epoch結束時比先前使用verified資料的還要高出3%，可以明顯看到semi-supervised的效果。以下為模型堆疊的情況：

層	相關參數
Input	input_shape = (128,20,1)
ZeroPadding	padding = (1,1)
Conv2D	卷積核數目 = 64，卷積核大小 = (3,3)
ReLU	激活層
ZeroPadding	padding = (1,1)
Conv2D	卷積核數目 = 64，卷積核大小 = (3,3)
ReLU	激活層
MaxPooling2D	pool_size = (2,2), strides = (2,2)
ZeroPadding	padding = (1,1)
Conv2D	卷積核數目 = 128，卷積核大小 = (3,3)
ReLU	激活層
ZeroPadding	padding = (1,1)
Conv2D	卷積核數目 = 128，卷積核大小 = (3,3)
ReLU	激活層
MaxPooling2D	pool_size = (2,2), strides = (2,2)
ZeroPadding	padding = (1,1)
Conv2D	卷積核數目 = 256，卷積核大小 = (3,3)
ReLU	激活層
ZeroPadding	padding = (1,1)
Conv2D	卷積核數目 = 256，卷積核大小 = (3,3)

ReLU	激活層
ZeroPadding	padding = (1,1)
Conv2D	卷積核數目 = 256, 卷積核大小 = (3,3)
ReLU	激活層
MaxPooling2D	pool_size = (2,2), strides = (2,2)
ZeroPadding	padding = (1,1)
Conv2D	卷積核數目 = 512 , 卷積核大小 = (3,3)
ReLU	激活層
ZeroPadding	padding = (1,1)
Conv2D	卷積核數目 = 512, 卷積核大小 = (3,3)
ReLU	激活層
ZeroPadding	padding = (1,1)
Conv2D	卷積核數目 = 512, 卷積核大小 = (3,3)
ReLU	激活層
BatchNormalization	
ZeroPadding	padding = (1,1)
Conv2D	卷積核數目 = 512 , 卷積核大小 = (3,3)
ReLU	激活層
ZeroPadding	padding = (1,1)
Conv2D	卷積核數目 = 512, 卷積核大小 = (3,3)
ReLU	激活層
ZeroPadding	padding = (1,1)
Conv2D	卷積核數目 = 512, 卷積核大小 = (3,3)
ReLU	激活層
BatchNormalization	

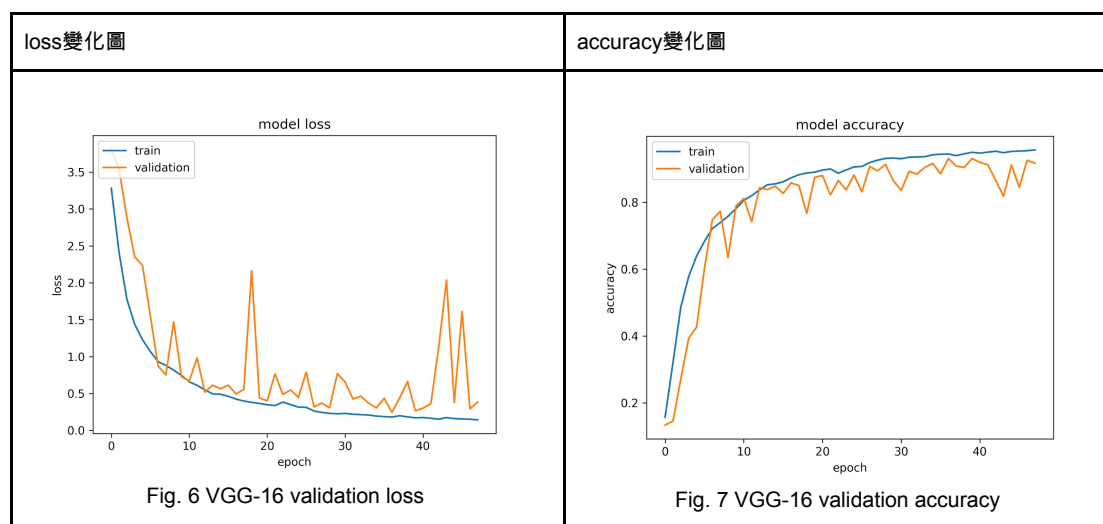
經過以上這些層後，再經過一層Flatten層，接著最後進入以下這些層：

層	相關參數
Dense	units = 1024
ReLU	激活層
Dropout	rate = 0.25
Dense	units = 256
ReLU	激活層
Dropout	rate = 0.5
Dense	units = 41 (共有41個類別)

以下為模型參數的數量：

Total parameters	31769833
Trainable parameters	31767273
non-trainable parameters	2560

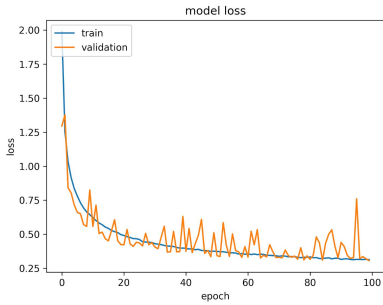
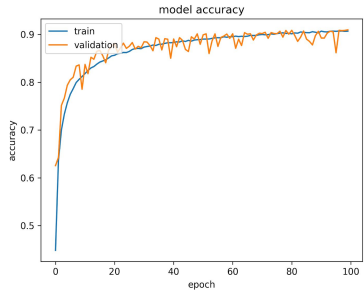
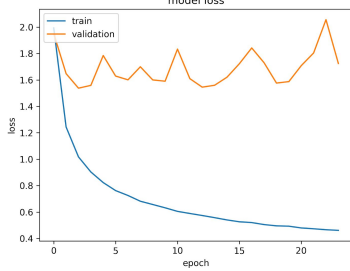
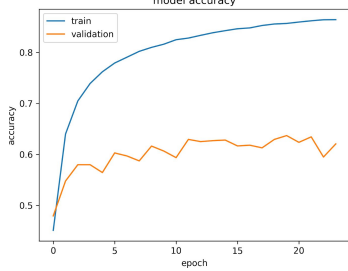
以下分別為模型訓練時的loss和accuracy的變化圖：



3. Experiment and Discussion

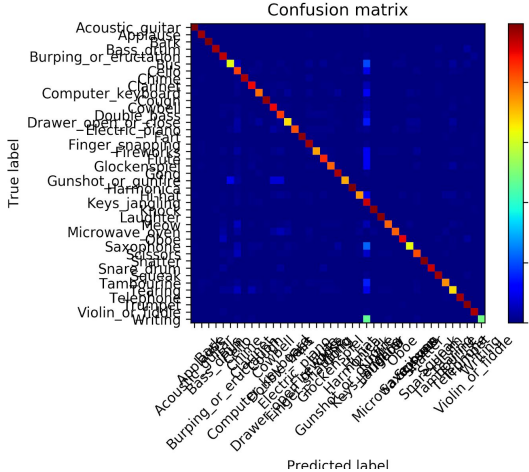
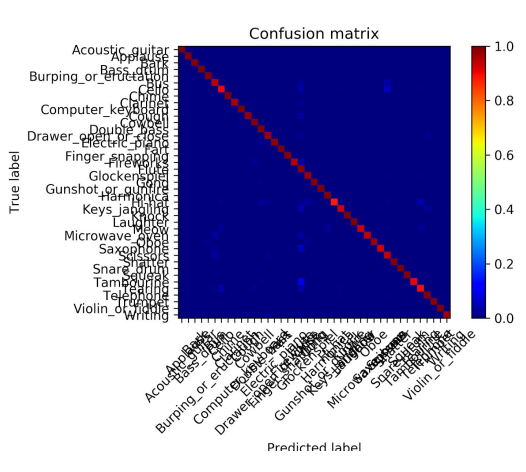
(1) 資料是否經過隨機打亂 (shuffle)

以下分別使用單純CNN模型，訓練資料有經過shuffle與沒有經過shuffle的loss和accuracy變化圖。訓練時的batch size為128，epoch為100，EarlyStopping的patience為20。

	loss變化圖	accuracy變化圖
有經過shuffle	 <p>Fig. 8 shuffle loss</p>	 <p>Fig. 9 shuffle accuracy</p>
未經過shuffle	 <p>Fig. 10 non-shuffle loss</p>	 <p>Fig. 11 non-shuffle accuracy</p>

由以上表格可以看出，資料未經shuffle的訓練過程中，validation loss一直無法下降且上下起伏，表示極不穩定；而準確率也一直無法提升，訓練效果非常不佳，具有過擬和 (over-fitting) 的情況發生。由於loss未有效下降，epoch超過patience的數目後，便earlystop，在20多個epoch時訓練過程即停止。將資料隨機打亂後，效果明顯變好，完整地跑完全部的100個epoch，且loss的趨勢大致穩定下降，accuracy也大致穩定上升，表示將訓練資料隨機打亂，較能避免過擬和的現象，且能提升訓練效果。

(2) 上述兩種不同模型預測validation set的混淆矩陣 (confusion matrix)

CNN	CNN with VGG-16
 <p>Fig. 12 CNN confusion matrix</p>	 <p>Fig. 13 VGG-16 confusion matrix</p>

備註：所有音檔的41個種類如下：

"Acoustic_guitar", "Applause", "Bark", "Bass_drum", "Burping_or_eructation", "Bus", "Cello", "Chime", "Clarinet", "Computer_keyboard", "Cough", "Cowbell", "Double_bass", "Drawer_open_or_close", "Electric_piano", "Fart", "Finger_snapping", "Fireworks", "Flute", "Glockenspiel", "Gong", "Gunshot_or_gunfire", "Harmonica", "Hi-hat", "Keys_jangling", "Knock", "Laughter", "Meow", "Microwave_oven", "Oboe", "Saxophone", "Scissors", "Shatter", "Snare_drum", "Squeak", "Tambourine", "Tearing", "Telephone", "Trumpet", "Violin_or_fiddle", "Writing".

單純只有用CNN架構訓練出來的model，其confusion matrix分佈已經滿平均的了，由左上到右下的斜線可以看出，有三個類別的準確率為六成，呈現黃色；而最右下角的方塊呈現綠色，表示正確label為Writing種類的音檔僅有約四成判斷正確，在途中水平方向往左對照過去，可以發現其餘有四成誤判成Keys_jangling。在predicted label為Keys_jangling的地方垂直方向往上對照，可以看出有許多方塊為淺藍色，表示許多true label為其他種類的音檔，如果判斷錯誤的話，通常此模型會將之歸類成Keys_jangling。表示Keys_jangling在此模型的理解中，充滿許多其他音檔種類具有的因素，為較為尋常、普通、無特徵的種類。

以VGG-16加上過濾的non-verified資料訓練出來的模型所做出的confusion matrix，則又顯得更為準確了。整張圖幾乎只有自己對自己label的格子有暖色系，並且顏色都偏向大紅，正確率都有九成以上。對角線以外的區域，則很明顯地都是代表沒有資料分佈的深藍色，代表著幾乎只有小於一成的資料被label錯誤。以此判斷，此模型及訓練方式，確實比第一種方法又更上一層樓。

(3) 有無做non-verified資料的retraining

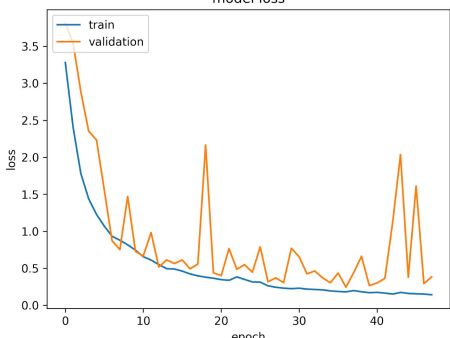
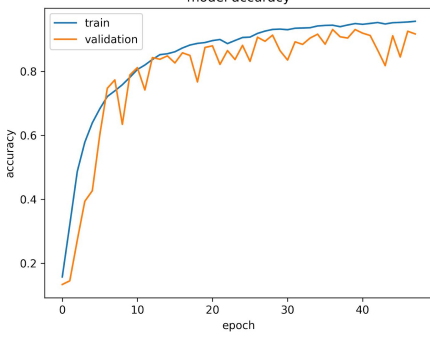
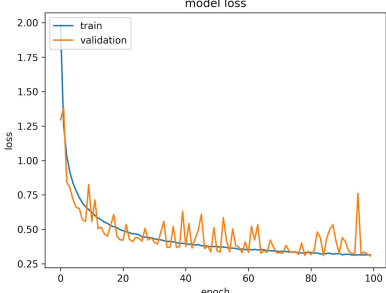
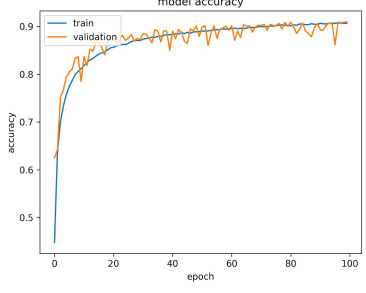
前面有提到我們過濾了non-verified data後，加進原本的verified data再對模型做retraining，會讓正確率提升3%左右。這邊以七種類似於VGG-16架構的CNN，用verified data所train出來的模型，和此模型經過retrain後的validation accuracy做比較。而由於validation accuracy在retrain的階段通常只有第一個epoch會有顯著的成長，後續都是持續往下掉，因此本題直接以數字表示，而非趨勢圖。

模型	validation accuracy	
	無retrain	有retrain
VGG-7, kernel=(8,3)	0.898	0.919
VGG-7, kernel=(3,3)	0.891	0.920
VGG-10, kernel=(8,3)	0.928	0.933
VGG-10, kernel=(3,3)	0.926	0.931
VGG-13, kernel=(8,3)	0.892	0.931
VGG-13, kernel=(3,3)	0.926	0.932
VGG-16, kernel=(3,3)	0.959	0.968

由以上七筆數據，我們可以看出，經過retrain的模型，準確率都有明顯的提升。上升的幅度大概為1%~4%不等，而這些數據全部都是retrain後第一個或第二個epoch結束時所紀錄的準確率。因此較好的訓練模型方法應為，以VGG-16, kernel=(3,3)訓練後，再已過濾的non-verified資料加上原有資料，訓練一或二epoch。

(4)音檔轉mel-spectrogram時的取樣頻率(sampling rate)的影響

Librosa的mel-spectrogram功能提供使用者設定我們想要以多大的取樣頻率來計算這個音檔的傅利葉轉換，參數名字是'sr'，也就是sampling rate的縮寫；預設值為22050Hz，也是我們用來產生音檔長度為20的取樣頻率；由於各種音色的頻率可能超過22050Hz，我們也試著以44100Hz做了實驗，而取樣頻率增大，頻譜長度當然也要跟著加寬，維持原有資料密度，我們以長度40來做。以下是分別用兩種採樣頻率，搭配VGG-16模型，所訓練出的loss與accuracy趨勢圖。

sampling rate	loss變化圖	accuracy變化圖
22050Hz	 <p>Fig. 14 sr=22050Hz loss</p>	 <p>Fig. 16 sr=22050Hz accuracy</p>
44100Hz	 <p>Fig. 15 sr=44100Hz loss</p>	 <p>Fig. 17 sr=44100Hz accuracy</p>

我們可以發現，取樣頻率為22050Hz的擁有較高的準確率，和我們原本預測的不同。估計原因是，可能沒有太多的音色擁有如此高頻的聲音，因此 over sampling反而造成模型訓練無法快速收斂。故最後選擇sr=22050Hz，頻譜長度為20的資料去訓練，以獲得較高的準確率。

4. Conclusion

由以上的實驗和分析，我們發現類似於VGG-16架構的CNN且加入 non-verified data的半監督式學習 (semi-supervised learning)，不論在避免過擬和的效果上，還是在各類別的分辨上 (由confusion matrix可以看出)，都表現得較單純的CNN模型好許多。

而經過這次機器學習的期末專題，也讓我們對這門領域有更進一步的認識。長達三個月的競賽，從一開始的徬徨，以及各種資料預處理的失敗，到後來漸漸找到適合的模型架構，並在最後衝刺時做了半監督式學習，使學習效率大幅提升，還是挺有成就感的，儘管Kaggle上的排名實在是蠻慘的。在有限的資源下 (GPU)，能有這樣的學習效果，相信我們還是能繼續踏上擁擠的ML之路的。

5. Reference

- (1)Amviida : <https://amvidia.com/mp3-normalizer>
- (2)Mfcc : <https://librosa.github.io/librosa/generated/librosa.feature.mfcc.html>
- (3)Mel-spectrogram :
<https://librosa.github.io/librosa/generated/librosa.feature.melspectrogram.html?highlight=melspectro#librosa.feature.melspectrogram>
- (4)VGG-16模型論文 : <https://arxiv.org/abs/1409.1556>
- (5)confusion matrix : <https://sunprinces.github.io/ML-Assignment3/p3.html>
- (6)loss和accuracy變化圖 :
<https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/>
- (7)freesound-audio-tagging :
<https://www.kaggle.com/c/freesound-audio-tagging/>