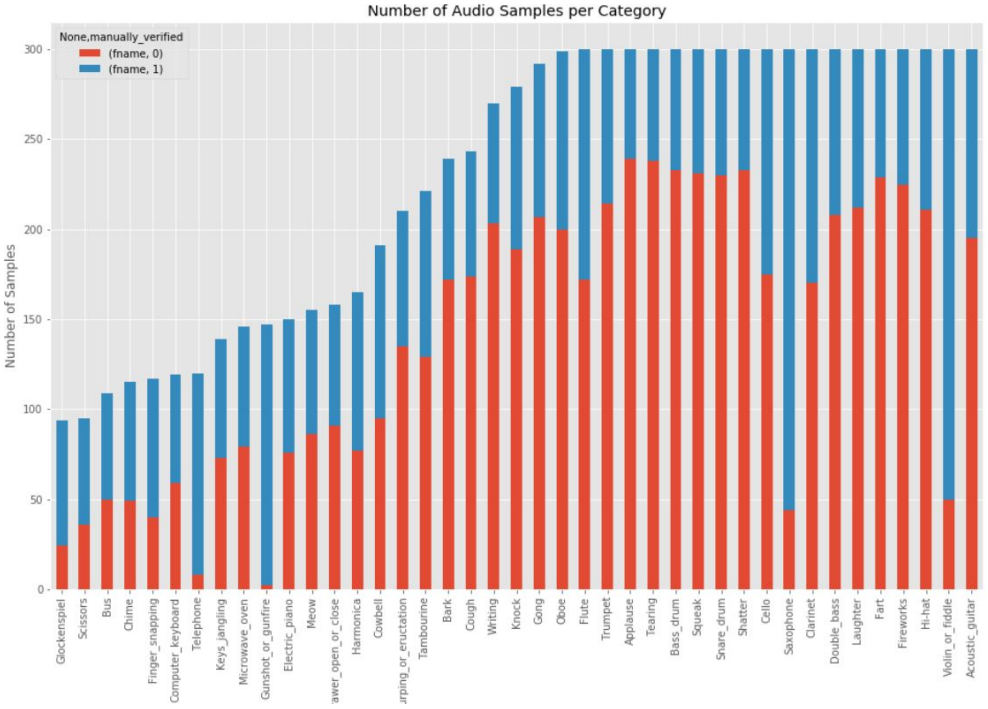
Freesound General-Purpose Audio Tagging Challeng

NTU_b04901020_算名師下去啦(解正平 吳倉永 黃平瑋 張問寬)

I. Introduction & Motivation

A. Task Description

利用包含在真實世界中收集到的各種聲音的dataset， 訓練出一個能自動將input sound做tagging的model，其中training set大約有9400筆資料，經過model的預測，能對聲音能從41種category中做classification。(圖 from Zafar)




```
"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"
```

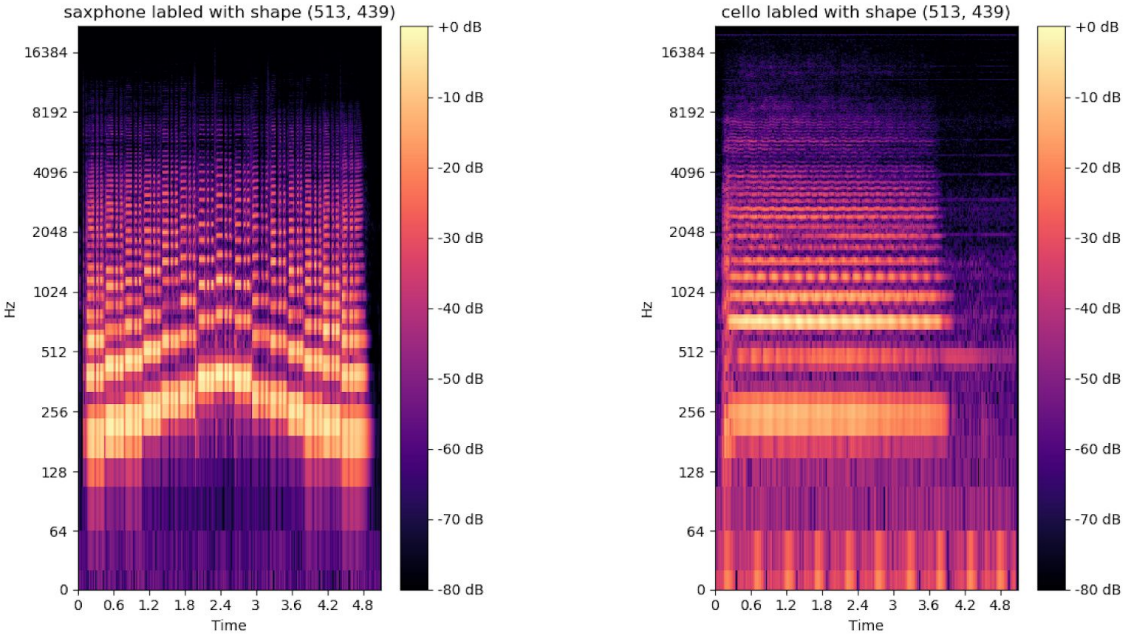
B. Motivation

因為同時有修語音相關的課程，所以final project決定作與聲音辨識相關的task，如此一來能將本課程所學的一些machine learning的技術與語音相關課程互相結合。

II. Data Preprocessing & Feature Engineering

A. Spectrogram

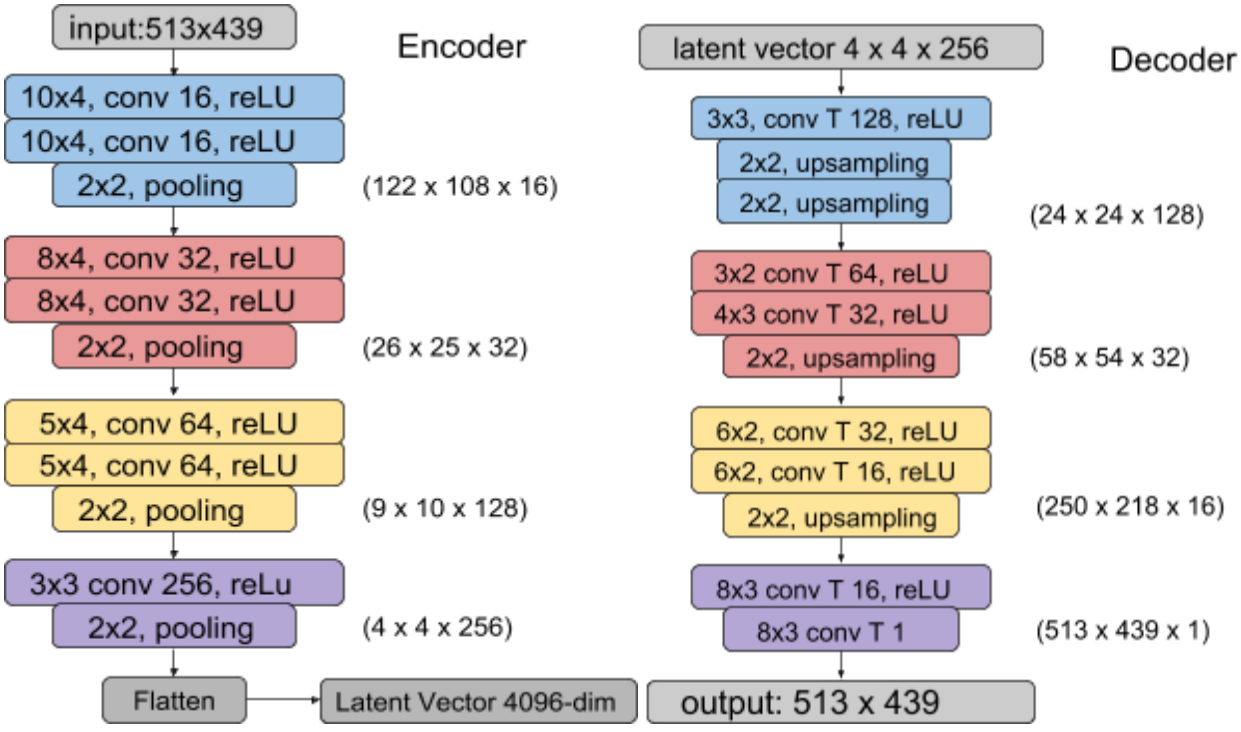
Spectrogram的作法是給定一段音訊檔，用一段window將所對印到的input作Short-time Fourier transform，即可得到一個時間對頻率的一個2維的圖片，每個點範圍在0到-80之間，表示在該時間特定頻率的強度。由於training data中的音訊長度不盡相同，所以得到的spectrogram大小也不太一樣，我們橫軸取出平均的長度513，將所有的圖片resize成相同的長度，如此一來，就能將這個題目轉換成圖片的分類問題，可以透過CNN抽取不同spectrogram的特徵，以達到訊練的目的。



可以看出不同樂器的spectrogram明顯在頻率的分佈上不太一樣

B. Dimension Reduction

將原本input為513 x439的spectrogram先除以(-80)做scaling, 再利用autoencoder降到4096維, model如下



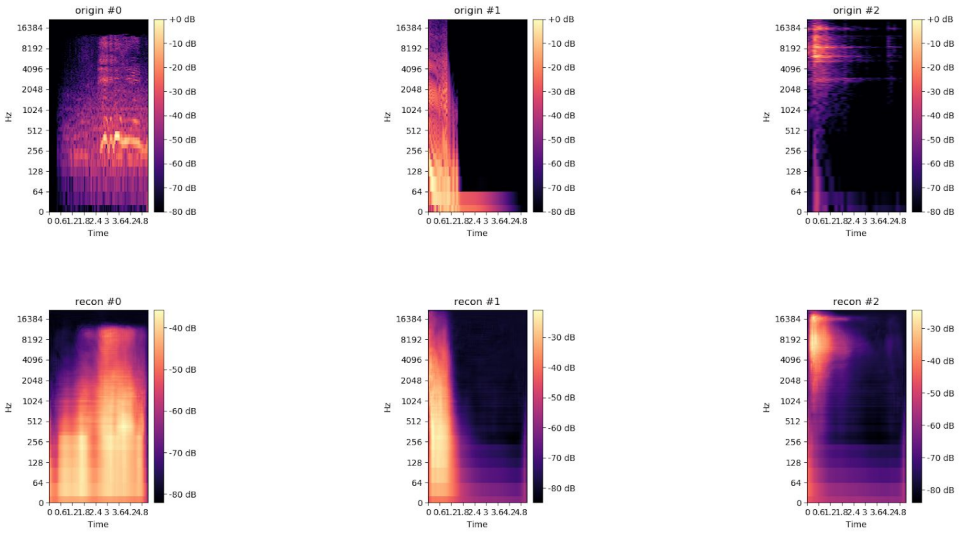
model training的參數： optimizer = Adam，learning rate為10^-5

loss function 採用min square error

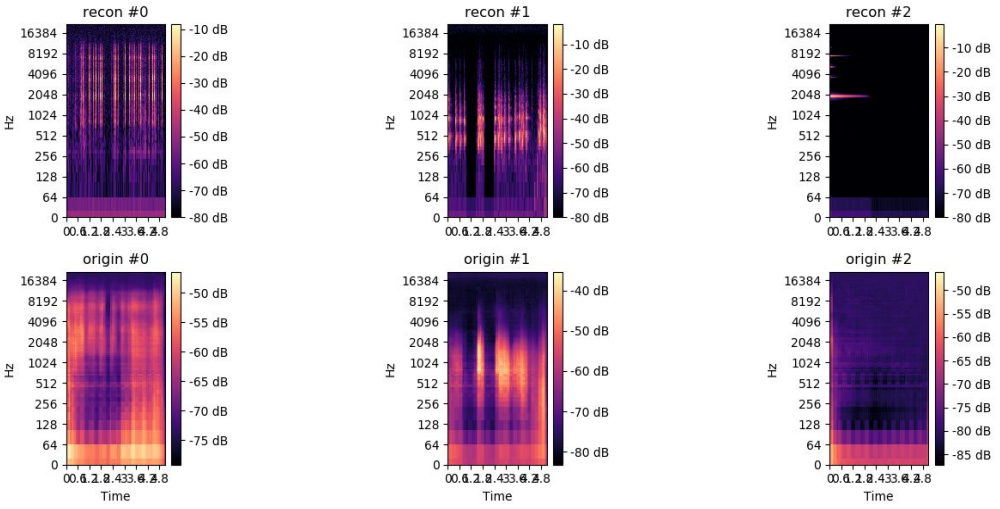
batch size = 16，epoch =100，有early stopping

loss 最低可以到 0.006

此外，我們還用latent vector去重建原本的圖，以比較降維方法的好壞，以下是幾張圖重現的結果



上排的是原圖, 下排是重現後的結果



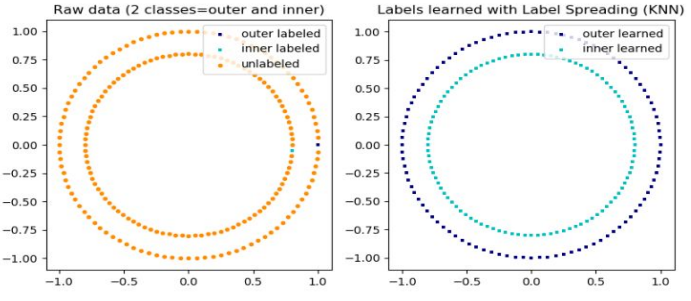
由上面的幾張圖可以發現, 在處理較大塊的面積或是直條狀的區塊時, 重現的效果較好

C. Labelspreading

由於原本9473個data，verified的只有3710個，unverified有5763個，如果只拿verified的做訓練會太少data，可是拿全部的會有不好的data導致model訓練不起來，因此我們想到用labelspreading的方式做cluster，判斷unverified的data是否可用，目的增加可用的traing data數量，而本部分分為兩個步驟，前面提到的降維以及分類。

1. methods

每一個聲音訊號spectrogram經過降維之後，我們透過labelspreading的方式，把unverified的訊號來標上new label，再依據原本的old label來判斷，如果兩者相同我們就，將這份data標成verified，已增加我們的資料量。



2. Labelspreading
   先用簡單的model來分類不同類別的data，並且標上新的label。
   - sklearn.semi_supervised.LabelSpreading
   - kernel='rbf',gamma=20,alpha=0.2,max_iter=100000,tol=0.001

3. Class similarity
   由於每個類別的聲音會有相似性，所以在做分類的可以事先人工將已有的label分組，好處是讓model在分類的時候較不會因為類別太多而容易有錯誤，在屬於自己的特定class判斷，可以有效提高verified data；壞處是因為這是由人工分組，可能會造成明明聲音相近似的data卻分不同區，這樣無法判斷出正確的label，而有一些誤差。以下是分不同class中可以增加多少verified　data的統計，以及各個class中原始verifed跟unverid的數量及比例。
   - 1 group (class：41all)

	41 class
all data	9473
verified	3710 (39.16%)
new verified	5174 (54.62%)
all verified	5174 (39.16% => 54.62%)

   - 2 group (class：19music and 22non-music)

	music	non music
all data	4827	4626
verified	2069 (42.86%)	1642 (35.34%)
new verified	3054 (63.27%)	2395 (51.55%)
all verified	5449 (39.16% => 57.52%)	

   - 3 group (class：19music, 10creature and 12non-human)

	music	creature	non human
all data	4827	2464	2182
verified	2069 (42.86%)	708 (28.73%)	934 (42.80%)
new verified	3054 (63.27%)	1340 (54.38%)	1334 (61.14%)
all verified	5728 (39.16% => 60.47%)		

- 5 group (class：4strings, 7musical, 8percussion, 10creature, 12non-human)

	musical	strings	percussion	creature	non human
all data	1779	1200	1848	2464	2182
verified	853 (47.95%)	572 (47.67%)	644 (34.85%)	708 (28.73%)	934 (42.80%)
new verified	1295 (72.79%)	811 (67.58%)	1162 (62.88%)	1340 (54.38%)	1334 (61.14%)
all verified	5942 (39.16% => 62.73%)				

從以上五種分析可以發現分組更為詳細可以使得labelspreading的結果較為突出，原因是data可以選擇的class更少，容易在人工限制的群組裡面選擇最為相近的label，然而可成會影響spreading不到真正的class因此我們並未採用。再來如果我們將class分成3組group，會發現data較不balance，這樣子去做spreading會使得原本data較多的class增加更多，使得資料更不平衡。綜上所述，我們只選擇分開music跟non-music的做spreading的data，可以看見我們能將原本verified 39.16%的data有效提升到57.52%，對於training會有很大幫助，至於還有約50%的data我們沒有用到，後面我們會介紹利用semi-supervisied learning的方式做訓練。

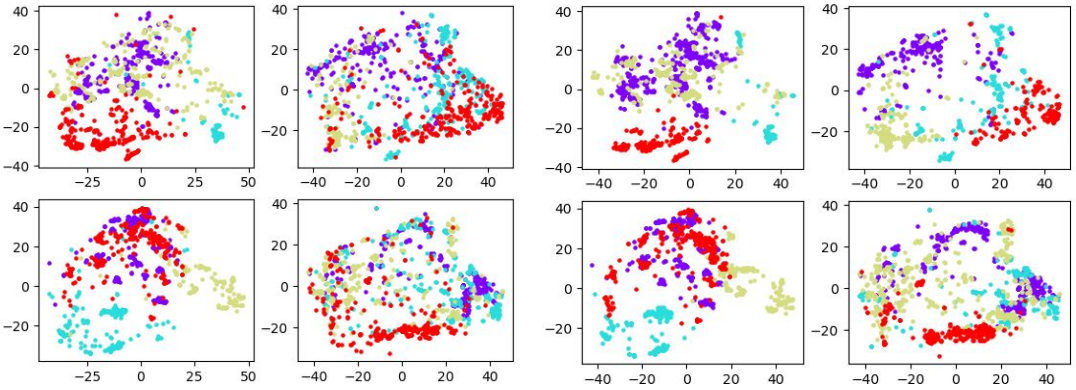
4. TSNE

由於我們透過labelspreading的方式增加verified資料，但是並未確定是否能夠有效分類為正確的class，因此我們使用TSNE的方式將資料visualize，觀察分為兩種group (music and non music) 的效果為何，發現可以有效準確的增加每個class的資料量，下列圖片會分為4張是因為一共有41個class我們將其分開來畫圖。

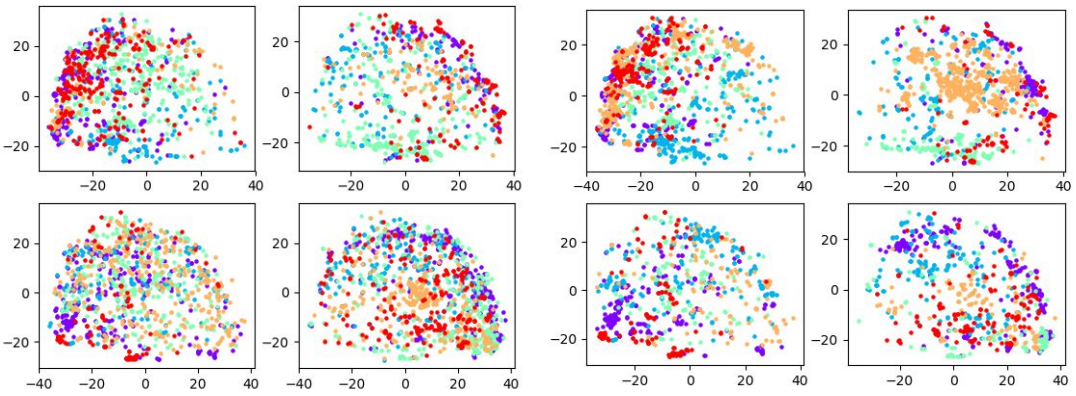
- music

  左邊的圖為未經過spreading的資料分布，右邊的圖是經過spreading。

  從圖可以發現右邊可以將資料分開，代表spreading的結果可以使得data有效增加；左邊的圖很雜代表原始標得label會因為unverified而影響分布。



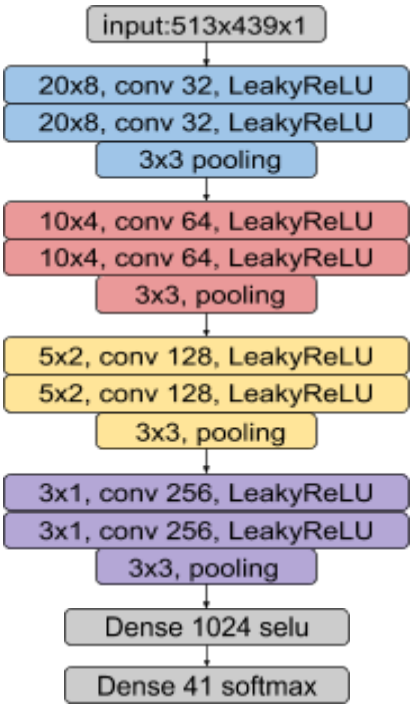
- non-music
  左邊的圖為未經過spreading的資料分布，右邊的圖是經過spreading。
  兩張圖都很難區分出每個class的資料，推測是因為non-music的聲音本來就很難區分，不像music都是有很明顯的特徵，經過spreading之後，可以看到部分class有被分開來，尤其右邊右下角的圖可以看出與左邊右下角圖的差異。

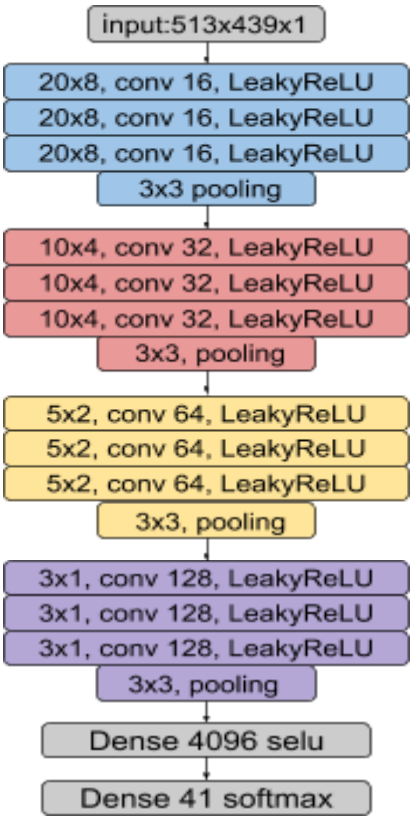


III. Model Description

A. Model structure

類似VGG16的做法，左邊是每一層CNN都是兩層，使用LeakyReLU activation方法，最後再用1024fully connected層搭配selu activation。右邊是每一層CNN都是三層，使用較多參數，最後是4096fully connected層搭配selu activation。會用VGG16的原因是這個model專門訓練圖片，可以適用在spectrogram的辨識中。





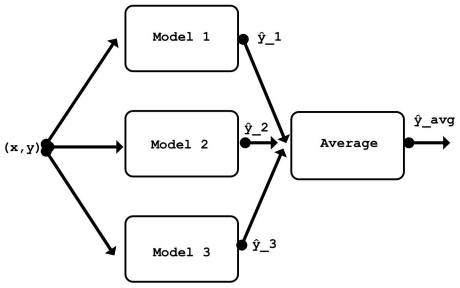
B. Data generator

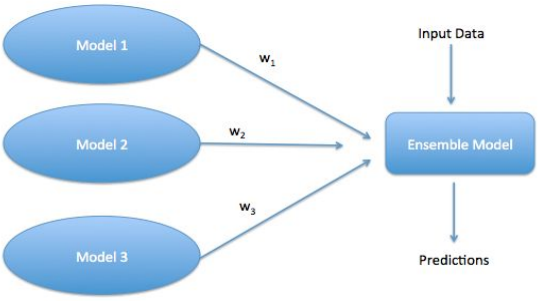
使用data generator的原因是因為記憶體不夠，不可能一次就把所有data都loading進來等待訓練，必須只能現在需要training的資料，因此我們使用generator在每個epoch產生每筆資料，只會暫存一點點data，然而不管是preprocess還是function都要小心處理，因為會使得時間過久，訓練的效率較很低。

```
class DataGenerator(keras.utils.Sequence):
    'Generates data for Keras'
    def __init__(self, list_IDs, labels, batch_size=32, dim=(32,32,32), n_channels=1,
                 n_classes=41, shuffle=True):
        'Initialization'
        self.dim = dim
        self.batch_size = batch_size
        self.labels = labels
        self.list_IDs = list_IDs
        self.n_channels = n_channels
        self.n_classes = n_classes
        self.shuffle = shuffle
        self.on_epoch_end()
```

C. Ensemble

由於訓練出來的模型會因為bias跟variance而有誤差，因此需要透過ensemble的方式將每個model predict的機率平均起來，方式有算術平均、幾何平均或是加權平均，每種方式出來的結果不盡相同。





D. Semisupervisied

因為data有分是否verified，造成不是全部資料都能當training data，雖然我們已經有實做labelspreading來增加數量，但是希望還是以semisupervisied learning的方法來改善資料不夠的問題，好處是可以改善model運用所有data，壞處是需要很久的時間，因此我們想了以下方法來改善。

1. tradeoff between iteration and epoch

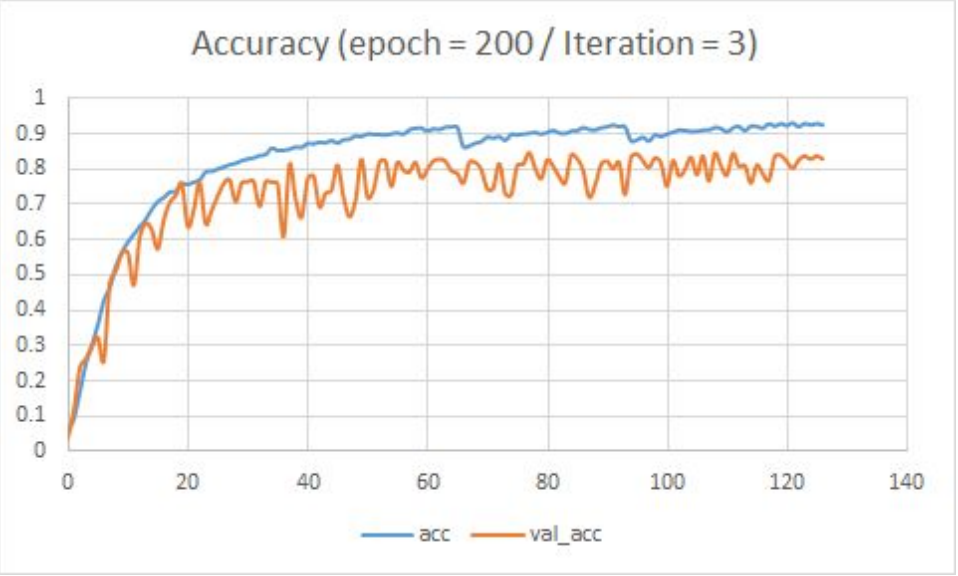
   由於每個epoch訓練的時間大約需要15分鐘，因此必須考慮怎麼樣有效率訓練好model，共取決於兩個因素，一個是epoch數量代表我模型要訓練到多好再去加入semi data增加資料量；一個是iteration數量代表我要取幾次semi data。如果epoch高就不需要太大iteration，因為model其實已經train好只是再增加data微調；但是我們可以也選擇不用訓練太好就先引入部分資料，這樣可以讓model學到比較多資料，雖然epoch少但是就要有較大的iteration。

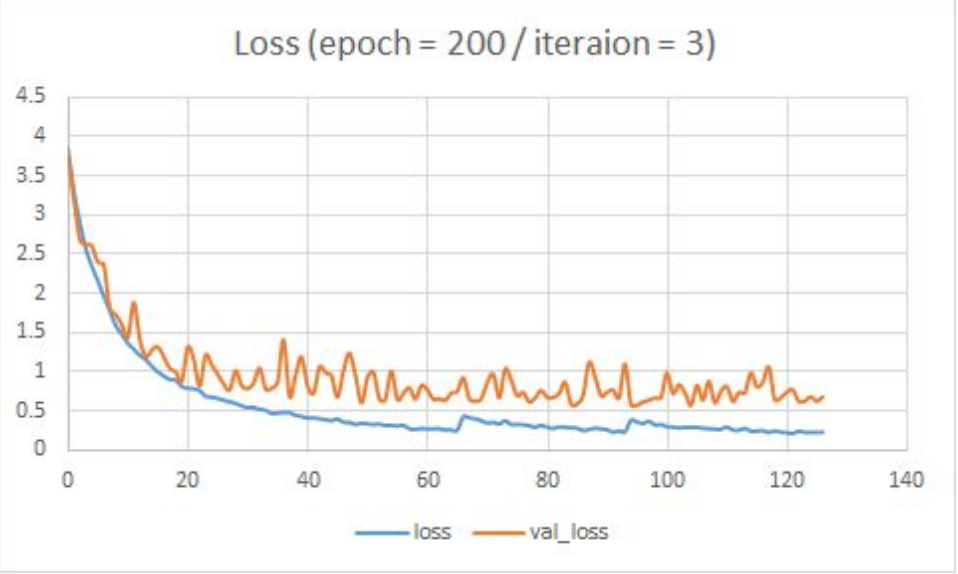
   下頁是我們的兩種設定方式，我們設定earlystop (15) 及 checkpoint：

- 先細再一點微調(epoch = 200 / iteration = 3)

  觀察圖可以發現，我們先把model訓練好，val acc約為0.829，之後才開始加入新的data，在epoch為66跟94時分別加入新的semi資料進入下個iteration，造成train acc下降，但是很快model又會學好，val acc有逐漸增加的趨勢。

iteration	epoch	data	train acc	val acc
1	0	5449	0.5687	0.0368
2	0	6118	0.8635	0.7610
3	0	6109	0.8801	0.8346
2	11	6118	0.9012	0.8474

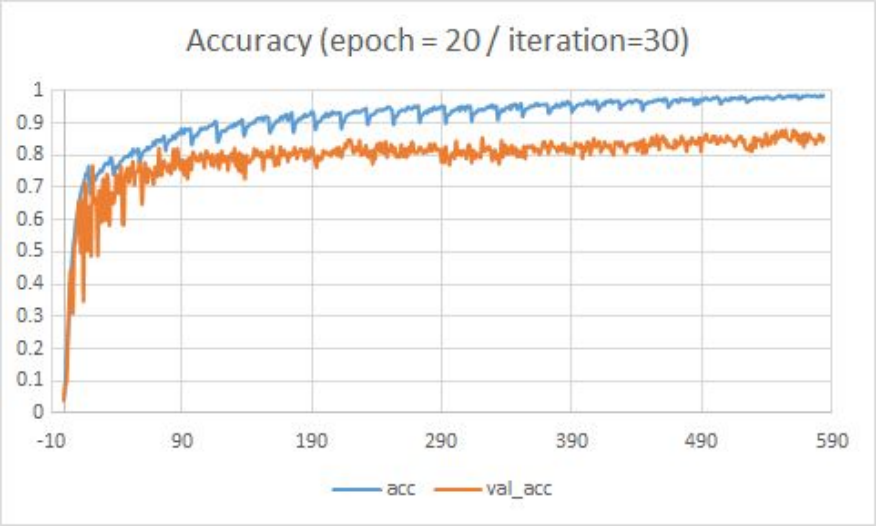


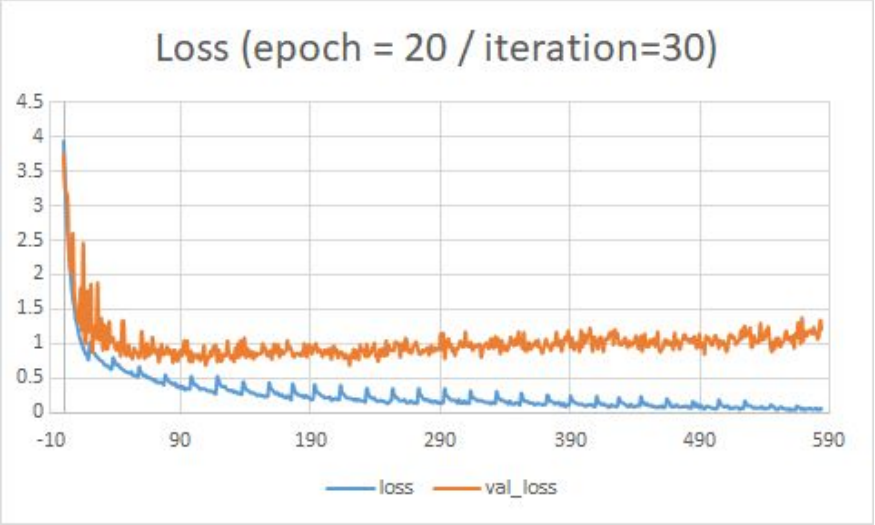


- 先粗再持續微調(epoch = 20 / iteration = 30)

  觀察圖可以發現，我們先把model以epoch=20訓練完，再開始加入新的data，從表可以發現每一次iteration通常val acc都有上升，直到train完30次iteration都還是有上升的趨勢。

iteration	epoch	data	train acc	val acc
1	0	8042	0.0440	0.0368
2	0	7539	0.6810	0.6397
3	0	7287	0.7398	0.6489
4	0	7052	0.7752	0.7776
5	0	6721	0.8147	0.7647
6	0	6569	0.8329	0.7537
7	0	6437	0.8406	0.8015
8	0	6267	0.8606	0.7647
9	0	6057	0.8697	0.7923
10	0	6079	0.8712	0.8033
11	0	5868	0.8799	0.7757
12	0	5879	0.8819	0.7923
13	0	5778	0.8961	0.7978
14	0	5649	0.8949	0.8070
15	0	5634	0.9021	0.8254
16	0	5625	0.8996	0.8272
17	0	5601	0.9059	0.7757
18	0	5644	0.9091	0.8015
19	0	5459	0.9206	0.8033
20	0	5509	0.9292	0.8051
21	0	5445	0.9327	0.8088
22	0	5383	0.9394	0.8309
23	0	5338	0.9422	0.8254
24	0	5330	0.9390	0.8051
25	0	5282	0.9482	0.8346
26	0	5224	0.9557	0.8456
27	0	5150	0.9585	0.8566
28	0	5111	0.9638	0.8180
29	0	5057	0.9725	0.8585
30	0	5022	0.9732	0.8640
28	11	5111	0.9798	0.8787





2. scheduled threshold

選擇semi data是否能當作training data餵給模型是看predict出來的機率數值，通常會直接設個threshold，機率大於某個值就代表是正確的(非黑即白)。然而我們需要防止overfitting，所以threshold有上限及下限，下限是為了表示label是夠確定的才能選，上限是為了怕已經訓練太好的資料還繼續訓練，造成overfitting。

除了上限及下限，我們認為每個iteration選擇多少機率水準下的data不應該是固定的，我們使用scheduled threshold來線性調整下限，讓model剛開始因為下限較低會先學較多的資料量，之後隨著model變強j我們使下限上修，這樣就不應該再選到機率不高的資料，怕造成bias，而且選擇的資料量會越來越少，有點像是先學大量再慢慢學精隨；先大的調整再來為調。

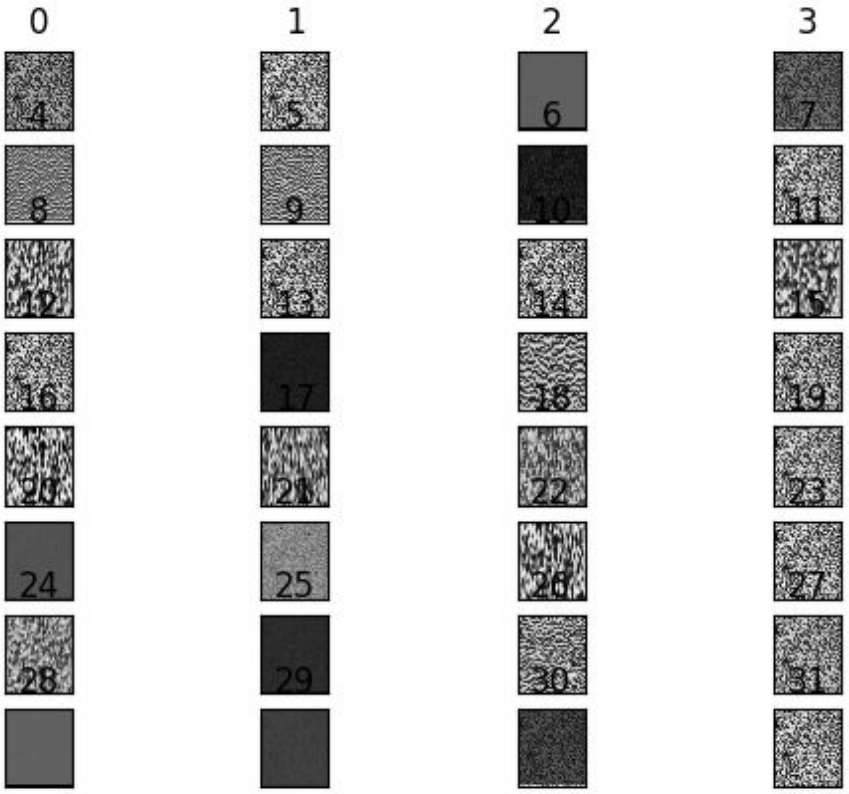
程式碼範例如下：可以發現我們原本下限是0.35，之後會變成0.65。

```
for i in range(30):
    threshold = (0.65 - i * 0.01)
```

IV. Experiment and Discussion (部分實驗已在partII介紹)

A. layer feature

下圖是VGG16較少參數的model，它第一層CNN filter中32個unit的Feature圖，可以發現有很多明暗變化，是因為spectrogram的圖就是很多不同amplitude不同亮度，部分unit是針對顏色深淺也就是數值大小，部分unit是針對細微紋路也就是數值分布，從第一層layer可以看出model知道如何解析spectrogram的圖。



B. performance in different methods

除了用將signal轉成spectrogram的方式來training，我們還多做了兩種方式，第一種是直接將row data數值餵給1D CNN；第二種是用MFCC將原訊號轉換，再以2D CNN的架構來訓練，下表為kaggle的分數，與未經過優化的spectrogram結果比較，發現再沒有任何前處理的狀態下spectrogram分數最高，但三者分數都很接近，推測可能會因為model或是參數設定影響，但此表可以發現raw data其實也訓練得起來，但是沒有甚麼能進步的方向。

methods	Public score
row data in 1D CNN	66.9%
MFCC in 2D CNN	67.1%
Spectrogram in 2D CNN	67,5%

C. performance advancement

models (Spectrogram)	Public score
all data	65.8%
all verified data	67.5%
with longer frequency window size	68.8%
vgg (more parameters)	77.5%
vgg (more parameters) in train music	78.5%
vgg (less paramenters) in train music	83.5% and 84.3%
vgg (less paprameters)(autoencoder)	87.9%
vgg (less parameters)(long iter semi)	86.8%
ensemble six data > 0.8	89.7%
ensemble four semi+train data	90.3%
ensemble with GM	91.1%

V. Conclusion

在這次作業中，我們嘗試了各種不同的前處理，如mfcc, spectrogram或甚至是未處理的raw data，在過程中發現了spectrogram的表現最佳，因此往後都是建立在spectrogram之上。而因為原始有verified的資料太少(約3千多筆)，我們又將降維後的spectrogram作clustering，讓有verifiied的資料做labelspreading，判斷unverified的資料是否可以使用，希望能達到增加training data的目的，而最後的結果也十分顯著，training data可以從39.16%增加到最高62.13%，2對往後模型的訓練有不少助益。

由於還是有將近一半的資料是unverfied的，所以我們採用semi-supervised的方式訓練，模型採用經典的vgg-16 和 vgg-19，比起我們自己設計的model，在準確度上有顯著的提昇，最後將Kaggle上accuracy超過0.8的模型作ensemble，最後的結果可以達到將近90%的準確度。

VI. Reference

[1] Hershey, Shawn, et al. "CNN architectures for large-scale audio classification." *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017.

[2] Zafar, Beginner's Guide to Audio Data
https://www.kaggle.com/fizzbuzz/beginner-s-guide-to-audio-data