EE603
PROJECT REPORT

Group : 30
Pratik Tidke 190638
Harshit Itondia 190369

Tasks:

1) Audio Event Detection
2) Audio Tagging

Table of Contents:

Project Report

Data Generation

Abstract:

In this part of project, new data is generated from music and speech files which will be used to train the model. Generated data contains mixed audio files each 10s long, containing an interval corresponding to music, an interval corresponding to speech and one or more interval corresponding to silence.

Methodology:

- The music.wav and speech.wav files are loaded with a sampling rate SR = 16000.From these two audio files mixed audio clips will be generated containing each of 10s which will have the data from both music.wav and speech.wav files along with some empty(silence) data interleaved in between.

- In a 10s mixed audio clip, there are 160000 audio samples among these 160000 samples len_m consecutive samples will be of music, len_sp consecutive samples will be of speech and len_si samples of silence are further split into group of samples to be placed between or around music or speech samples. These len_m, len_sp and len_si are selected randomly using Numpy.random.randint() function.

```
music_value = np.random.randint(1,5)
speech_value = np.random.randint(1,5)
zero_value = 10-speech_value-music_value
music_length = music_value*SR
speech_length = speech_value*SR
zero_length = zero_value*SR
```

Randomly generated len_m(music_length), len_sp(speech_length), len_si(zero_length)

- First a random number (r_no1) is selected from the range (1, len_si), if this number is significant then a sequence of 0's of length r_no1 is appended in the newly formed array.

```
r_no = np.random.randint(len(x))
if(abs(round(r_no/SR,4))>=0.02):
  onset.append(0)
  offset.append(round(r_no/SR,4))
  class_.append('silence')
  filename.append(fn)
```

r_no1(r_no)

- After that a sequence of length len_m from music.wav file is inserted in the array. Now the total length of the array is (r_no1 + len_m). Again a number (r_no2) is randomly selected from the range (r_no+len_m, len_si). If the distance from (r_no + len_m) to r_no2 is significant then a sequence of 0's of length (r_no2 - (r_no + len_m)) is appended in the array.

```
onset.append(round(r_no/SR,4))
o = round(r_no/SR,4) + music_value
offset.append(o)
class_.append('music')
filename.append(fn)
x = np.insert(x,r_no,m)
```

```
si = np.random.randint(o*SR,len(x))
if(abs(o-round(si/SR,4))>=0.02):
  onset.append(o)
  offset.append(round(si/SR,4))
  class_.append('silence')
  filename.append(fn)
```

r_no2(si)

- Then the speech signal of length (len_sp) is appended in the array. Now the total length of array is (r_no2 + len_sp).

```
onset.append(round(si/SR,4))
offset.append(round(si/SR,4) + speech_value)
class_.append('speech')
filename.append(fn)
x = np.insert(x,si,s)
```

Caption

- Finally the remaining samples of silence are appended in the array. This lead to the formation of a 10s long mixed audio clip.

```
if(abs(round(si/SR,4) + speech_value-10)>=0.0
  onset.append(round(si/SR,4) + speech_value)
  offset.append(10)
  class_.append('silence')
  filename.append(fn)
```

- Similarly many such audio clips are formed consisting of different sub-clips of music and speech from music.wav and speech.wav files to insert in the different 10s mixed clips.

- Each sub-clip of each 10s clip is defined using a group rows in the table in which each row consist of the:
  Filename: Name of the 10s clip
  Onset : The starting time of each sub-clip.
  Offset. : The ending time of each sub-clip.
  Class : The class which it belongs to, in this case silence, music or speech.

	filename	onset	offset	class
0	sound0_.wav	0.0000	1.6189	silence
1	sound0_.wav	1.6189	3.6189	speech
2	sound0_.wav	3.6189	8.8667	silence
3	sound0_.wav	8.8667	9.8667	music
4	sound0_.wav	9.8667	10.0000	silence

Audio Event Detection:

Abstract:

Our goal in this part is to train the model with data generated in previous section ,predict the class of unknown dataset with the trained model and find accuracy and F-score of the classified data.
With each sub-clip of 10s mixed audio clip as input we have to train the model to predict the class of different intervals in the new audio clips.

Methodology:

Data preprocessing:

- We have the training data which consists of a 10s mixed audio clips which are divided into sub-clips according to the class those sub-clip belong (music, speech, silence).We know the Audio data of each of those sub-clips and the class which they belong to.

- Each sub-clip is fed to librosa.feature.mfcc() function which will frame-wise extract mfcc features of that sub-clip.In our model we are extracting 40 mfcc features for better results.

- To reduce the dimensionality from 2-D to 1-D we compressed all the frames so the value of each mfcc feature is the mean of value of that mfcc feature in all the frames.

- So now we have a 1-D array of 40 mfcc features corresponding to each sub-clip along with the class-name of that sub-clip of a 10s clip for all the 10s clips.Each of the three classes (music , speech, silence) are encoded in three one-hot vector each for a class.
- So now after pre-processing we have mfcc features and one-hot vectors for each sub-clip.

Model:

- Model used for classification is a neural network model with an input layer having 40 neurons which are mfcc features of a single sub-clip, 3 hidden layer with each having 128, 256, 128 neurons respectively and an output layer consisting of 3 neurons.
- ‘Relu’ nonlinearity is used in the hidden layers while softmax nonlinearity is used in the output layer.
- Below is the data for model.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	5248
activation (Activation)	(None, 128)	0
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 256)	33024
activation_1 (Activation)	(None, 256)	0
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32896
activation_2 (Activation)	(None, 128)	0
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 3)	387
activation_3 (Activation)	(None, 3)	0

Total params: 71,555
Trainable params: 71,555
Non-trainable params: 0

Trainable model

- Categorical cross entropy function is used for error/loss calculation and accuracy is used as a metric to judge the model performance after each iteration
- Back-propagation is used as learning algorithm to improve the accuracy of our predictions after every epoch/iteration.

```
Epoch 1/60
13/13 [==============================] - ETA: 0s - loss: 0.1464 - accuracy: 0.9885
Epoch 00001: saving model to /content/drive/MyDrive/Audio_event_detection/Saved_model
13/13 [==============================] - 0s 17ms/step - loss: 0.1464 - accuracy: 0.9885 - val_loss: 0.0180 - val_accuracy: 0.9962
Epoch 2/60
 1/13 [=>............................] - ETA: 0s - loss: 0.7359 - accuracy: 0.9688
Epoch 00002: saving model to /content/drive/MyDrive/Audio_event_detection/Saved_model
13/13 [==============================] - 0s 10ms/step - loss: 0.1858 - accuracy: 0.9847 - val_loss: 0.0170 - val_accuracy: 0.9962
Epoch 3/60
 1/13 [=>............................] - ETA: 0s - loss: 0.0120 - accuracy: 1.0000
Epoch 00003: saving model to /content/drive/MyDrive/Audio_event_detection/Saved_model
13/13 [==============================] - 0s 10ms/step - loss: 0.0965 - accuracy: 0.9911 - val_loss: 0.0166 - val_accuracy: 0.9962
Epoch 4/60
 1/13 [=>............................] - ETA: 0s - loss: 9.7549e-05 - accuracy: 1.0000
Epoch 00004: saving model to /content/drive/MyDrive/Audio_event_detection/Saved_model
13/13 [==============================] - 0s 14ms/step - loss: 0.1195 - accuracy: 0.9898 - val_loss: 0.0148 - val_accuracy: 0.9962
Epoch 5/60
 1/13 [=>............................] - ETA: 0s - loss: 0.0042 - accuracy: 1.0000
Epoch 00005: saving model to /content/drive/MyDrive/Audio_event_detection/Saved_model
13/13 [==============================] - 0s 12ms/step - loss: 0.1438 - accuracy: 0.9860 - val_loss: 0.0128 - val_accuracy: 0.9962
Epoch 6/60
 1/13 [=>............................] - ETA: 0s - loss: 0.7494 - accuracy: 0.9688
Epoch 00006: saving model to /content/drive/MyDrive/Audio_event_detection/Saved_model
13/13 [==============================] - 0s 12ms/step - loss: 0.1265 - accuracy: 0.9834 - val_loss: 0.0095 - val_accuracy: 0.9962
Epoch 7/60
13/13 [==============================] - ETA: 0s - loss: 0.0933 - accuracy: 0.9872
Epoch 00007: saving model to /content/drive/MyDrive/Audio_event_detection/Saved_model
13/13 [==============================] - 0s 16ms/step - loss: 0.0933 - accuracy: 0.9872 - val_loss: 0.0079 - val_accuracy: 0.9962
Epoch 8/60
 1/13 [=>............................] - ETA: 0s - loss: 0.0083 - accuracy: 1.0000
Epoch 00008: saving model to /content/drive/MyDrive/Audio_event_detection/Saved_model
13/13 [==============================] - 0s 11ms/step - loss: 0.1263 - accuracy: 0.9847 - val_loss: 0.0084 - val_accuracy: 0.9962
Epoch 9/60
```

Accuracy and loss after every epoch

Prediction using audio files:

- Now as the model is trained, we can test the model performance by with unknown data, which is in this audio files.
- The test data is given to us in the format of audio clips each of length 10s.

```
check

['S001',
 'S002',
 'S003',
 'S004',
 'music_noisy1',
 'music_noisy2',
 'music_noisy3',
 'music_noisy4',
 'music+speech_noisy1',
 'music+speech_noisy2',
 'music+speech_noisy3',
 'music+speech_noisy4']
```

Audio files

- Each of these 10s long audio clips are split into sub-clips where each sub-clip has a sample length of 2500 samples so with a sampling rate of 16000, each sub-clip is of 0.15625s. So total 64 sub-clips are formed from each 10s clip.
- Mfcc features are extracted from each of those sub-clips.
- Here also 40 mfcc features are extracted frame-wise and the all the frames are compressed by taking mean along the frame axis to make the feature array 1-D.

0	S0010	[-499.80484, 97.405014, -6.242565, 29.702097, ...
1	S0011	[-501.80035, 103.22322, -4.691728, 31.590801, ...
2	S0012	[-511.67734, 102.85643, 3.918148, 26.303196, 8...
3	S0013	[-521.1769, 105.81555, 6.26138, 27.474567, 8.4...
4	S0014	[-463.32495, 110.38657, 16.875725, 25.956945, ...
5	S0015	[-241.54382, 113.09074, -17.936165, 41.87146, ...
6	S0016	[-261.03754, 101.65785, 5.5851126, 24.882952, ...
7	S0017	[-259.40717, 124.6599, 3.2924523, 32.076763, -...
8	S0018	[-337.75558, 127.10234, 17.054003, 44.520363, ...
9	S0019	[-283.3834, 116.027306, 56.556385, 4.9752417, ...
10	S00110	[-412.67877, 138.44751, 46.317863, 25.527258, ...
11	S00111	[-501.67255, 130.56061, 17.712553, 33.96024, 9...
12	S00112	[-516.5272, 120.54533, 11.750652, 31.629002, 1...
13	S00113	[-520.0316, 116.01216, 9.071032, 30.107952, 13...
14	S00114	[-525.6825, 115.063194, 4.707974, 27.366674, 1...
15	S00115	[-519.7355, 114.811295, 3.8564599, 27.688946, ...

First 15 sub-clips of clip S00

- The features of each sub-clips are inserted in the trained neural network model as input vectors and the output of the neural network will be a vector of length 3. The 3 values of output vector corresponds to the dominance or weight of each class (music, speech, silence) in that sub-clip.
- Now simplest approach to determine which class among the three the sub-clip belong to is to find the value which is most dominant among those three output vector value. Which ever class has the highest value will be the class of that sub-clip.

```
result= []
for i in range(len(X)):
  a = model.predict(X[i].reshape(1,-1))
  result.append(a[0])
```

```
[ ] res = np.array(result)
    result_list = []
    for i in range(res.shape[0]):
      arr = res[i]
      index = np.where(arr == np.amax(arr))[0][0]

      if index == 0:
        result_list.append("music")
      elif index == 1:
        result_list.append("silence")
      else:
        result_list.append("speech")
```

```
result_list
```

```
['speech',
 'speech',
 'speech',
 'speech',
 'speech',
 'music',
 'music',
 'music',
 'speech',
 'music',
 'speech',
 'speech',
 'speech',
 'speech',
 'speech',
 'speech',
 'speech',
 'speech',
 'speech',
 'music',
 'music',
 'music',
 'music',
 'music',
 'music',
 'music',
```

Class-name of sub-clips

Convolution using silence padding:

- In the result_list we have class name for each of the sub-clip of duration 0.15625s.
- Now what if there is a sub-clip having a class-name silence in between two clips of class-name music? It's class name should be music as it can be a silence interval of musical tone.
- So to avoid this, convolution is done to eliminate some small silence intervals in between a music or speech.
- For convolution the window size is tuned from 3-10 to get better performance
- For example if a window (size = 4) is convolved with the array of sub-clips. 4 consecutive sub-clips are considered at a time and whichever class-name has the maximum frequency among the class-names of those 4 sub-clips, will be the modified class-name of the first sub-clip (among those 4 sub-clips). Padding of 3 silence class-name is done for the last 3 sub-clips to convolve fully with window.

```
def max_window(index, array, window_size):
  count_music = 0
  count_silence = 0
  count_speech = 0
  for i in range(window_size):
    if array[i+index]=="music":
      count_music +=1
    elif array[i+index]=="speech":
      count_speech +=1
    else:
      count_silence +=1
  largest = maximum(count_music, count_silence, count_speech
  if count_music == largest:
    return "music"
  elif count_speech == largest:
    return "speech"
  else:
    return "silence"
```

Code for convolution

```
modified_result
```

```
['speech',
 'speech',
 'speech',
 'music',
 'music',
 'music',
 'music',
 'music',
 'speech',
 'speech',
 'speech',
 'speech',
 'speech',
 'speech',
 'speech',
 'speech',
 'speech',
 'music',
 'music',
 'music',
 'music',
 'music',
 'music',
 'music',
 'music',
 'speech',
 'speech',
 'speech',
```

Class-name of each sub-clip after convolution

- Now for the final result we have to merge consecutive sub-clips with same class-name and also ensure that all those sub-clips belong to the same 10s clip.

```
final_result = []
count = 0
onset_time = 0
offset_time = 0
frame = 2500/16000

for i in range(modified_result_array.shape[0]-1):
  var = modified_result_array[i]
  if modified_result_array[i+1] == modified_result_array[i]:
    offset_time += frame
  else:
    if (offset_time - onset_time)>0.5 and onset_time != offset_time and modified_result_array[i] != "silence":
      final_result.append([check[count], modified_result_array[i], onset_time, offset_time])
    offset_time += frame
    onset_time = offset_time

  if offset_time >= 10:
    dummy = offset_time - 10
    offset_time = 10
    if onset_time == offset_time or modified_result_array[i] == "silence":
      continue
    if (offset_time - onset_time)>0.5 and onset_time != offset_time and modified_result_array[i] != "silence":
      final_result.append([check[count], modified_result_array[i], onset_time, offset_time])
    count +=1
    onset_time = 0
    offset_time = dummy
```

Code for merging consecutive clips with same class-nam

```
final_result[26:30]
```

```
[['music_noisy1', 'music', 0, 1.25],
 ['music_noisy1', 'music', 2.8125, 4.21875],
 ['music_noisy1', 'music', 6.09375, 7.34375],
 ['music_noisy2', 'music', 0, 1.09375]]
```

Final result

Audio Tagging

Abstract :

The modified array which we formed by convolving a window with the result array (extracted from the output of neural network), consist of sub-clips of all 10s clips with their predicted class-name (for each sub-clip). Now our goal is to predict the class-name of complete the 10s clip.

Methodology:

- We know there are 64 sub-clips of each 10s clip. From these 64 sub-clips of a clip we will count the number of time each class-name (music and speech) has arrived in those 64 sub-clips.
- Now we will define a limit which is a tuneable parameter and if the count of music class-name or count of speech class-name in those 64 sub-clips is greater than that limit then 10s clip corresponding to those sub-clips will have that class-name.
- So the there can be two class-name of that particular 10s clip.

```
[ ]  output[4:9]
```

```
[['music_noisy1', array([1, 1])],
 ['music_noisy2', array([1, 1])],
 ['music_noisy3', array([1, 0])],
 ['music_noisy4', array([1, 0])],
 ['music+speech_noisy1', array([1, 1])]]
```

Here ‘music_noise’1 is both music and speech, but ‘music_noise3’ is only music