ECE539 Proposal

Overview and background

This project focuses on the classification and segmentation of audio, particularly for distinguishing between singing and speech. The primary goal is to develop an accurate audio classifier and segmentation algorithm that can handle a mixture of speech and singing.

Dataset

- gtzan_music_speech is an open source dataset
- There are video on bilibili in which a person only talk (no singing) and in which the same person only sing (no talking).
- The videos are crawled down (using Python, selenium and bilili), converted to audio, sliced into 30-second pieces, resampled at 22050KHz, and mixed into mono sound. (using Python and ffmpeg). This results in 661500 samples each piece.

Below is the summary of dataset

```
$ ls xxm_singing/
0.wav    105.wav  112.wav  12.wav   127.wav  134.wav  19.wav   26.wav   33.wav
40.wav   48.wav   55.wav   62.wav   7.wav    77.wav   84.wav   91.wav   99.wav
1.wav    106.wav  113.wav  120.wav  128.wav  135.wav  2.wav    27.wav   34.wav
41.wav   49.wav   56.wav   63.wav   70.wav   78.wav   85.wav   92.wav
10.wav   107.wav  114.wav  121.wav  129.wav  136.wav  20.wav   28.wav   35.wav
42.wav   5.wav    57.wav   64.wav   71.wav   79.wav   86.wav   93.wav
100.wav  108.wav  115.wav  122.wav  13.wav   14.wav   21.wav   29.wav   36.wav
43.wav   50.wav   58.wav   65.wav   72.wav   8.wav    87.wav   94.wav
101.wav  109.wav  116.wav  123.wav  130.wav  15.wav   22.wav   3.wav    37.wav
44.wav   51.wav   59.wav   66.wav   73.wav   80.wav   88.wav   95.wav
102.wav  11.wav   117.wav  124.wav  131.wav  16.wav   23.wav   30.wav   38.wav
45.wav   52.wav   6.wav    67.wav   74.wav   81.wav   89.wav   96.wav
103.wav  110.wav  118.wav  125.wav  132.wav  17.wav   24.wav   31.wav   39.wav
46.wav   53.wav   60.wav   68.wav   75.wav   82.wav   9.wav    97.wav
104.wav  111.wav  119.wav  126.wav  133.wav  18.wav   25.wav   32.wav   4.wav
47.wav   54.wav   61.wav   69.wav   76.wav   83.wav   90.wav   98.wav

$ ls xxm_speech/
0_0.wav   0_17.wav  0_5.wav   1_4.wav  2_5.wav   3_16.wav  3_24.wav  3_9.wav
4_16.wav  4_24.wav  5_0.wav   5_3.wav  6_2.wav   7_10.wav  7_9.wav   8_8.wav
0_1.wav   0_18.wav  0_6.wav   1_5.wav  3_0.wav   3_17.wav  3_25.wav  4_0.wav
4_17.wav  4_25.wav  5_1.wav   5_4.wav  6_3.wav   7_11.wav  8_0.wav   8_9.wav
0_10.wav  0_19.wav  0_7.wav   1_6.wav  3_1.wav   3_18.wav  3_26.wav  4_1.wav
4_18.wav  4_3.wav   5_10.wav  5_5.wav  6_4.wav   7_2.wav   8_1.wav
0_11.wav  0_2.wav   0_8.wav   1_7.wav  3_10.wav  3_19.wav  3_3.wav   4_10.wav
4_19.wav  4_4.wav   5_11.wav  5_6.wav  6_5.wav   7_3.wav   8_2.wav
0_12.wav  0_20.wav  0_9.wav   2_0.wav  3_11.wav  3_2.wav   3_4.wav   4_11.wav
4_2.wav   4_5.wav   5_12.wav  5_7.wav  6_6.wav   7_4.wav   8_3.wav
```

```
0_13.wav  0_21.wav  1_0.wav  2_1.wav  3_12.wav  3_20.wav  3_5.wav   4_12.wav
4_20.wav  4_6.wav   5_13.wav  5_8.wav  6_7.wav  7_5.wav   8_4.wav
0_14.wav  0_22.wav  1_1.wav  2_2.wav  3_13.wav  3_21.wav  3_6.wav   4_13.wav
4_21.wav  4_7.wav   5_14.wav  5_9.wav  6_8.wav  7_6.wav   8_5.wav
0_15.wav  0_3.wav   1_2.wav  2_3.wav  3_14.wav  3_22.wav  3_7.wav   4_14.wav
4_22.wav  4_8.wav   5_15.wav  6_0.wav  7_0.wav  7_7.wav   8_6.wav
0_16.wav  0_4.wav   1_3.wav  2_4.wav  3_15.wav  3_23.wav  3_8.wav   4_15.wav
4_23.wav  4_9.wav   5_2.wav   6_1.wav  7_1.wav  7_8.wav   8_7.wav

$ ls xxm_singing/ | wc -w; ls xxm_speech/ | wc -w
137
137

$ ffprobe -i xxm_singing/0.wav
  ......
  Duration: 00:00:30.00, bitrate: 352 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 22050 Hz, 1 channels,
s16, 352 kb/s
```

- There are also long (not cropped) videos in which a person sometimes talks and sometimes sings. There are human-labeled timestamps of starts of each singing. These labels are crawled, parsed, and stored in hh,mm,ss format as shown below

```
0, 19, 23
0, 23, 22
0, 30, 28
0, 38, 20
0, 45, 9
1, 1, 37
1, 4, 10
1, 8, 45
1, 13, 27
1, 19, 12
1, 22, 15
1, 25, 15
1, 30, 34
1, 33, 0
1, 38, 27
1, 42, 32
1, 49, 17
1, 53, 2
1, 58, 47
2, 1, 50
```

- They are used to test the performance of our final program.

Others' work

[Classifying Music and Speech with Machine Learning]()

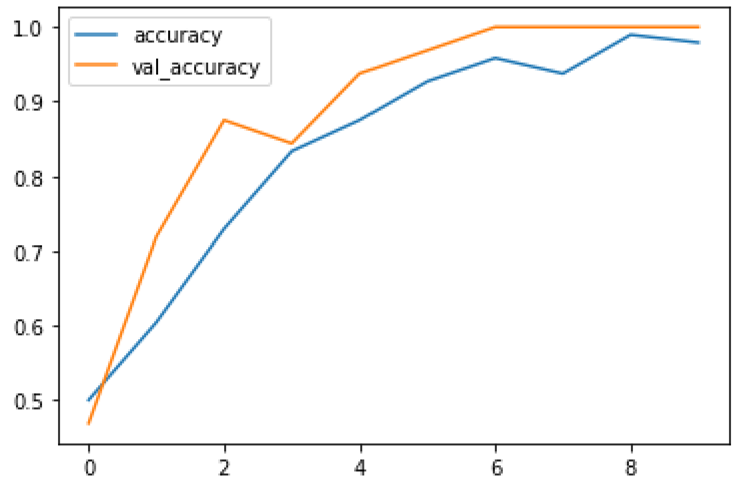
- Data preprocessing: FFT

Transfer each 30-second audio (as a whole, no further chunking) into frequency domain (using FFT) and normalize the amplitude.

- Model: CNN

```
model = models.Sequential([
    layers.Input(shape=input_shape),
    preprocessing.Resizing(64, 64),
    norm_layer,
    layers.Conv2D(32, 3, activation='relu'),
    layers.Conv2D(64, 3, activation='relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.25),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(num_labels),
])
```

- Result: below is from the reference

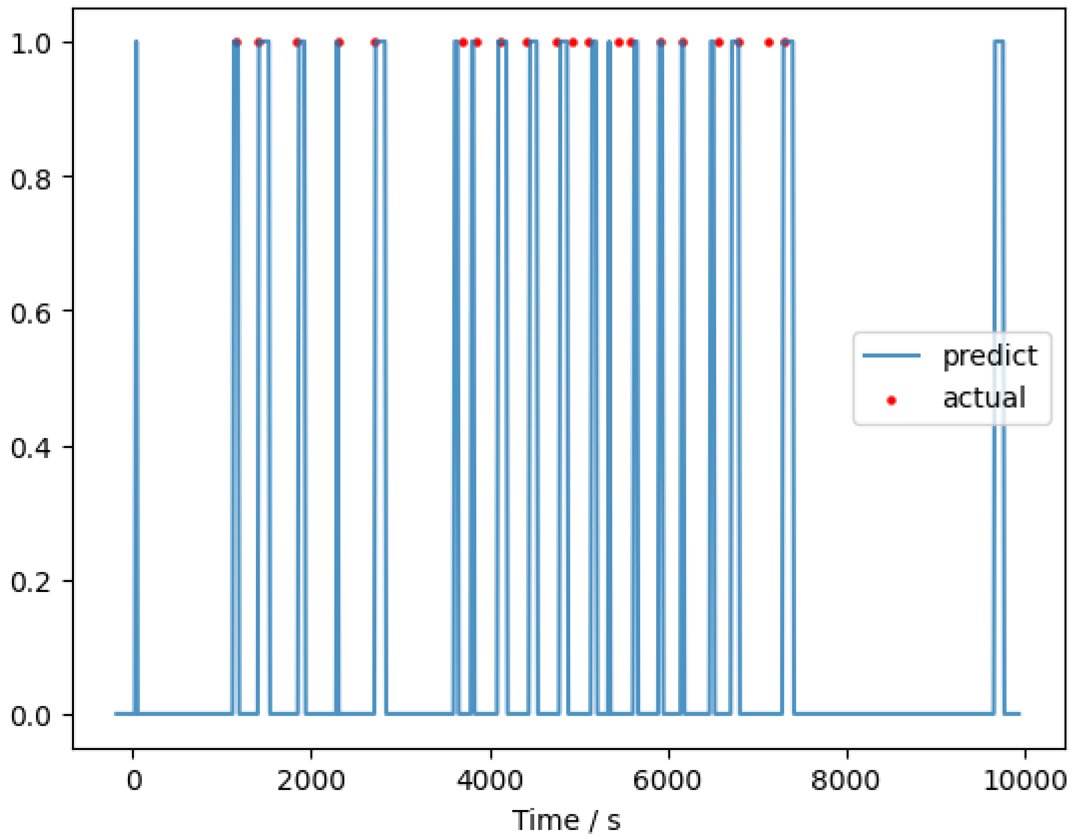


Reproduction of above work

I have reproduced the above work, and achieved similar accuracy. Then I applied the model to long mixed audio, and below is the block diagram of implementation:

```
    [Long audio]-->[CHUNK 1][CHUNK 2] ... [CHUNK N]
                    |
                    V
                    (FFT)-->[Spectrum]-->(CNN)-->[Probability of each chunk]
                                                                          |
                                                                          V
[Predict outcome]<--(Thredhold)<--[Filtered probability]<--(FIR)
```

Bwlow is the result we currently have, blue line is the predict outcome from above model, red dot is from actual label. (1 for singing, 0 for speech)



From the outcome, we can see the FPR is high as sometimes there is BGM but it is actually speech not singing. VPR will be applied to fix this issue.

Methods

- FFT
- CNN
- FIR

Computing recourses

- Tesla T4 from Google Colab