

SOC Midterm Report


21B030035- Sripradheep K

1.Installing Magenta

When trying to use the google collab for installing Magenta, the code didn't work because collab failed to create an environment of magenta to continue installing the library. So the github repo of Magenta was used. The code still faced a lot of issues while running it in Ubuntu.Starting from installing g++ and then moving on to install rtmidi in itself was a big task.python-rtmidi required python 3.9 or above and it also required some missing packages to be installed separately. Those commands with screenshots are attached below. Thanks to GPT.


- **Ubuntu/Debian:**

bash

 Copy code

```
sudo apt update  
sudo apt install g++
```

bash


 Copy code

```
sudo apt update  
sudo apt install build-essential libasound2-dev
```

3. Use precompiled wheels:

If available, use precompiled wheels to avoid building from source:


sh

 Copy code

```
pip install wheel
```

1. Add the LLVM repository:

sh

 Copy code

```
wget https://apt.llvm.org/llvm.sh
chmod +x llvm.sh
sudo ./llvm.sh 8
```

2) Study on libraries available in Python for audio processing

1) Librosa:

It's a music library to import audio files, process them and plot the signals as a Mel Spectrogram. The code for the functionalities is shown below.

Reading in Audio Files

There are many types of audio files: `mp3`, `wav`, `m4a`, `flac`, `ogg`

```
In [2]: audio_files = glob('../input/ravdess-emotional-speech-audio/**/*.wav')
```

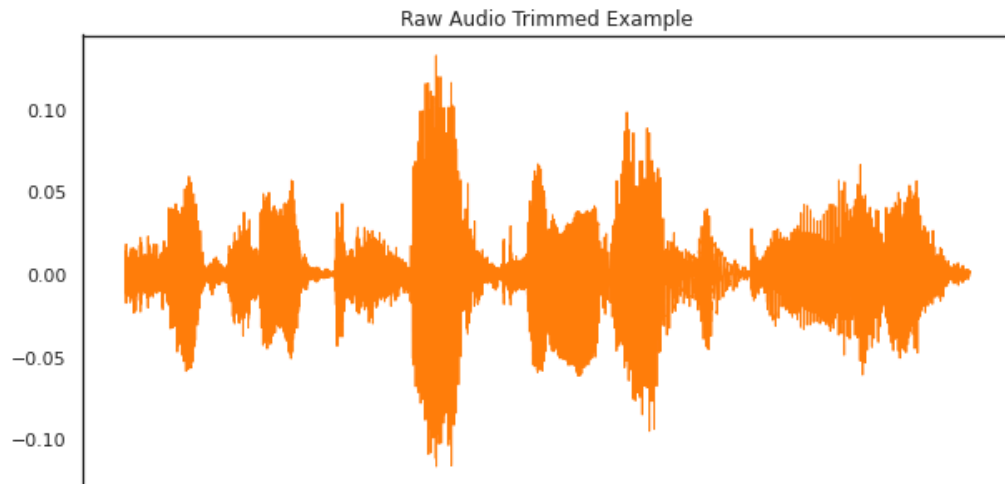
```
In [3]: # Play audio file
        ipd.Audio(audio_files[0])
```

Out[3]:

▶ 0:00 / 0:03 ————— 🔊 ⋮

This also includes chopping off the silence in the audio signals and import them to create spectrograms.

```
In [6]: # Trimming leading/lagging silence
y_trimmed, _ = librosa.effects.trim(y, top_db=20)
pd.Series(y_trimmed).plot(figsize=(10, 5),
                        lw=1,
                        title='Raw Audio Trimmed Example',
                        color=color_pal[1])
plt.show()
```



Spectrogram

```
In [8]: D = librosa.stft(y)
S_db = librosa.amplitude_to_db(np.abs(D), ref=np.max)
S_db.shape
```

```
Out[8]: (1025, 153)
```

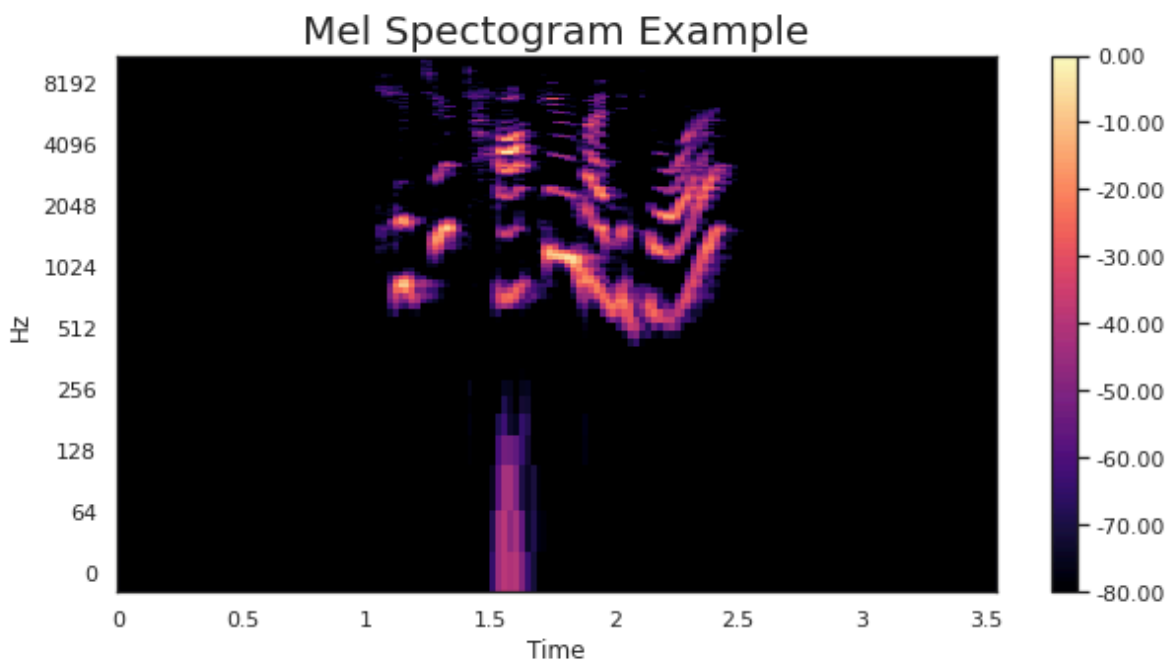
```
In [9]: # Plot the transformed audio data
fig, ax = plt.subplots(figsize=(10, 5))
img = librosa.display.specshow(S_db,
                                x_axis='time',
                                y_axis='log',
                                ax=ax)
ax.set_title('Spectrogram Example', fontsize=20)
fig.colorbar(img, ax=ax, format=f'%0.2f')
plt.show()
```



Mel Spectrogram

```
In [10]: S = librosa.feature.melspectrogram(y=y,
                                             sr=sr,
                                             n_mels=128 * 2,)
S_db_mel = librosa.amplitude_to_db(S, ref=np.max)
```

```
In [11]: fig, ax = plt.subplots(figsize=(10, 5))
# Plot the mel spectrogram
img = librosa.display.specshow(S_db_mel,
                                x_axis='time',
                                y_axis='log',
                                ax=ax)
ax.set_title('Mel Spectrogram Example', fontsize=20)
fig.colorbar(img, ax=ax, format=f'%0.2f')
plt.show()
```



Kaggle link: <https://www.kaggle.com/code/scratchpad/notebookd6549b967c/edit>

Source: <https://www.youtube.com/watch?v=ZqpSb5p1xQo>

2)Essentia

This library is also used to pre-process audio files and fit them into machine learning models.

- **Equalizer**: an **EasyLoader** that applies an equal readiness matching to the audio

```
# we start by instantiating the audio loader:
loader = essentia.standard.MonoLoader(filename='.././../test/audio/recorded/dubstep.wav')

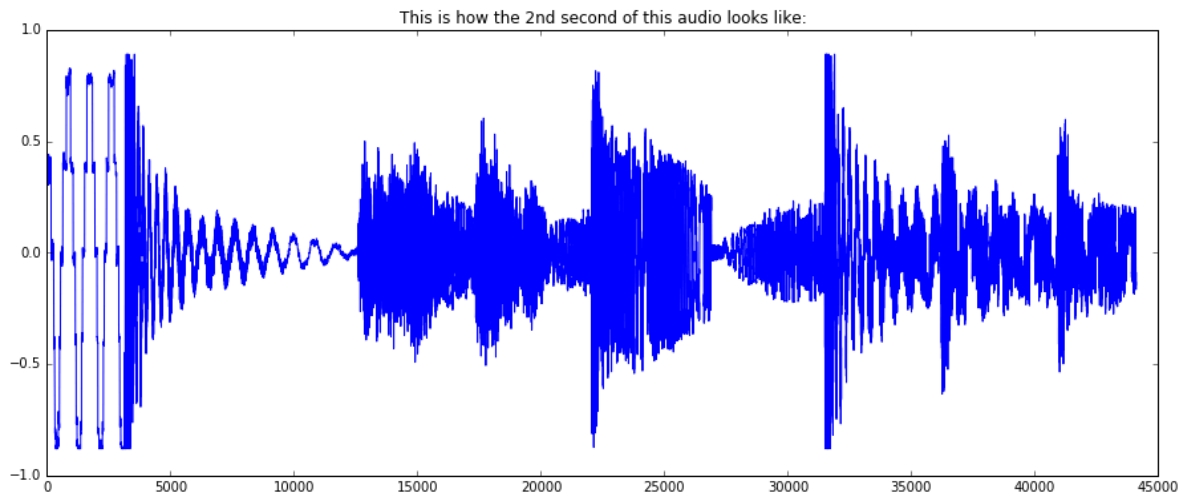
# and then we actually perform the loading:
audio = loader()
```

```
# This is how the audio we want to process sounds like
import IPython
IPython.display.Audio('.././../test/audio/recorded/dubstep.wav')
```

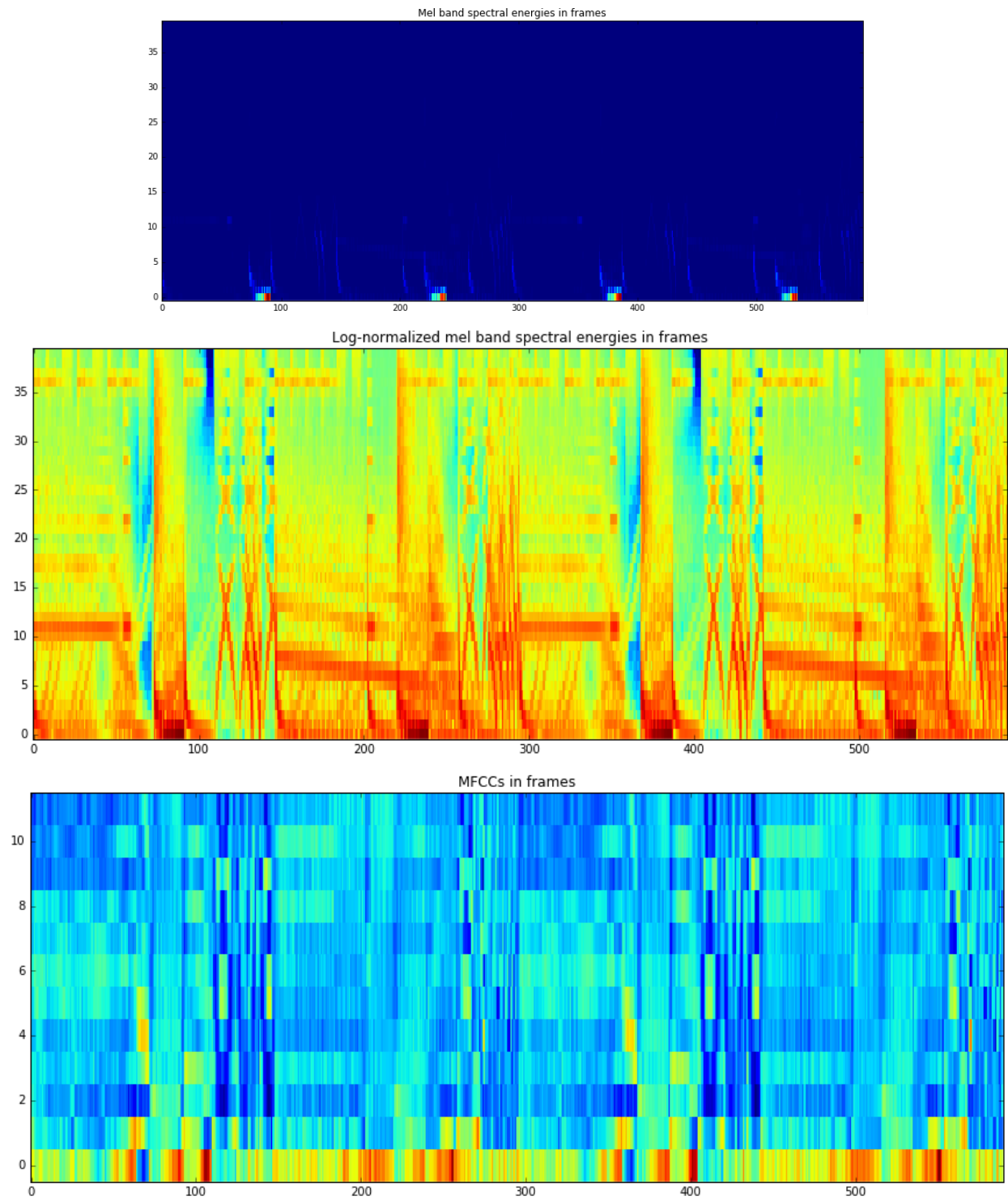
▶ 0:00 / 0:06 🔊 ⋮

```
# pylab contains the plot() function, as well as figure, etc... (same names as Matlab)
from pylab import plot, show, figure, imshow
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (15, 6) # set plot sizes to something larger than default

plot(audio[1*44100:2*44100])
plt.title("This is how the 2nd second of this audio looks like:")
show()
```



But this one generates much detailed Spectrograms as shown below



This is used for complex audio analysis and it also provides data for machine learning algorithms which is our primary need.

3) Pedalboard

This is also an audio extracting library with which data analysis and sata pre processing can be done.

Source: <https://www.youtube.com/watch?v=fl8cbEsxz8I>

Havent completed it fully yet.