Speech Processing Project

Language and Libraries

This project is done primarily using the python programming language with use of Jupyter notebooks as well.

The most important library used is Librosa. Librosa is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems.

Pyaudio is another library used for the audio analysis.

Numpy library is used to work with matrices form of the signal for performing various functions and matplotlib is used to visualize graphs.

Theory Behind

Feature Extraction and analysis of speech signal

Before going to the code we need to understand the theory behind how to analyse and extract the features of a speech signal.

- Sampling : Sound waves are digitized by sampling them at discrete intervals known as the sampling rate. Usually the sampling rate is 44100 Hz.
- Spectrogram : A spectrogram is a visual way of representing the signal strength, or “loudness”, of a signal over time at various frequencies present in a particular waveform. Not only can one see whether there is more or less energy at, for example, 2 Hz vs 10 Hz, but one can also see how energy levels vary over time. A common format is a graph with two geometric dimensions: one axis represents time, and the other axis represents frequency; a third dimension indicating the amplitude of a particular frequency at a particular time is represented by the intensity or color of each point in the image . We use stft (short time fourier transform) to convert a signal to a spectrogram.

Feature Extraction and analysis of speech signal

- Spectral Centroid : The spectral centroid is a measure used in digital signal processing to characterise a spectrum. It indicates where the center of mass of the spectrum is located. Perceptually, it has a robust connection with the impression of brightness of a sound.

$$\text{Spectral centroid} = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n-0}^{N-1} x(n)}, \qquad (3)$$

- Spectral Rolloff : Spectral rolloff is the frequency below which a specified percentage of the total spectral energy, e.g. 85%, lies.
- Spectral Bandwidth : The spectral bandwidth is defined as the width of the band of light at one-half the peak maximum (or full width at half maximum [FWHM]) and is represented by the two vertical red lines and λSB on the wavelength axis.

Feature Extraction and analysis of speech signal

- Zero crossing rate : The zero-crossing rate is the rate of sign-changes along a signal, i.e., the rate at which the signal changes from positive to zero to negative or from negative to zero to positive. A very simple way for measuring the smoothness of a signal is to calculate the number of zero-crossing within a segment of that signal. A voice signal oscillates slowly — for example, a 100 Hz signal will cross zero 100 per second — whereas an unvoiced fricative can have 3000 zero crossings per second.
- Mel-frequency cepstrum (MFCC) : In sound processing, the mel-frequency cepstrum is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. Mel-frequency cepstral coefficients are coefficients that collectively make up an MFC.They are derived from a type of cepstral representation of the audio clip

Speech Enhancement and Noise Removal

In order to perform speech enhancement it is necessary to remove the noise from the audio clip.

There are many ways to do this but basically in three ways : Adaptive , Non adaptive(traditional) , and Machine learning method.

Programming Speech processing: Feature Extraction

Loading the data

There are many formats in which the audio files : mp3 , wav ,etc. Because we are using Librosa the recommended format is wav file. In this slide we load the data and see its default sampling rate. We also observe how the audio signal is converted into a numpy array.

Loading Data

```
In [1]: import librosa

In [2]: audio_file='Stammer2.wav'

In [3]: x,sr=librosa.load(audio_file)

In [4]: x #The wav file has been converted to array of floating point series of time

Out[4]: array([ 0.01439938,  0.02108896,  0.0186679 , ..., -0.01122512,
               -0.01080854, -0.01217357], dtype=float32)

In [5]: sr #The default sampling rate

Out[5]: 22050
```

Visualizing the data

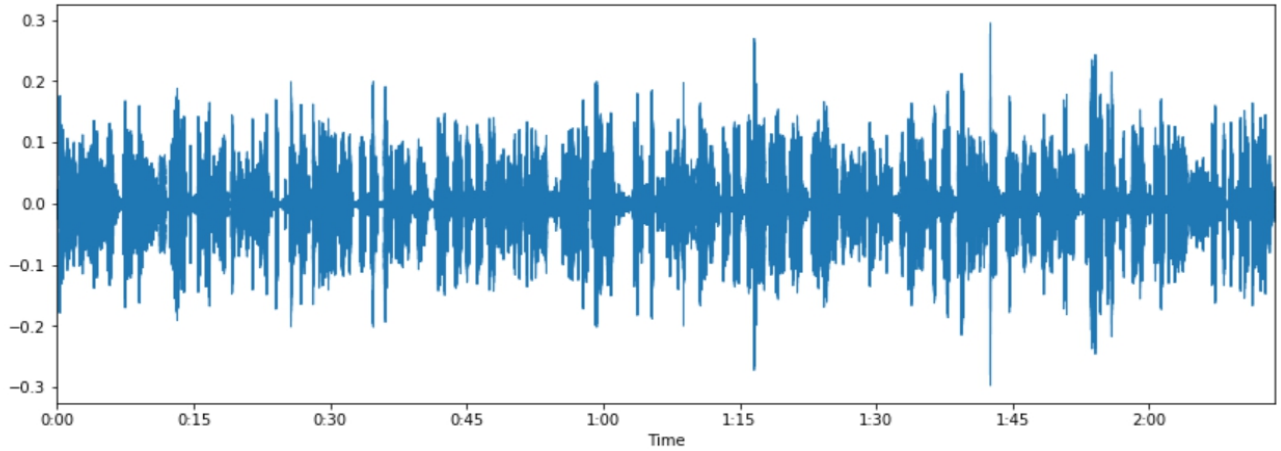
Now that we have loaded the signal let us see how this signal actually looks like.

Visualizing the data

In [6]:
```
import matplotlib.pyplot as plt
```

In [7]:
```
import librosa.display
plt.figure(figsize=(14, 5))
librosa.display.waveplot(x, sr=sr)
```

Out[7]:
```
<matplotlib.collections.PolyCollection at 0x2c0473bcdc8>
```



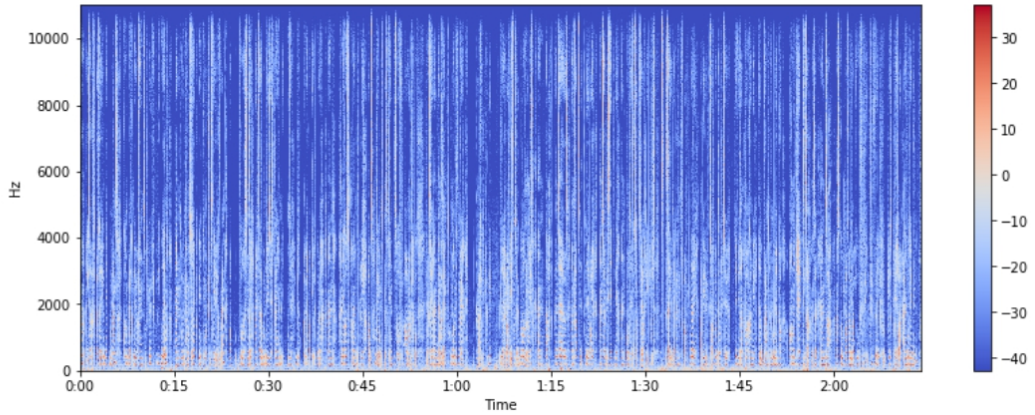
Spectrogram analysis

This slide shows how to convert a signal into spectrogram using the librosa feature librosa.display.specshow.

Spectrogram analysis ¶

In [8]:
```
X = librosa.stft(x)
Xdb = librosa.amplitude_to_db(abs(X))
plt.figure(figsize=(14, 5))
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar()
```

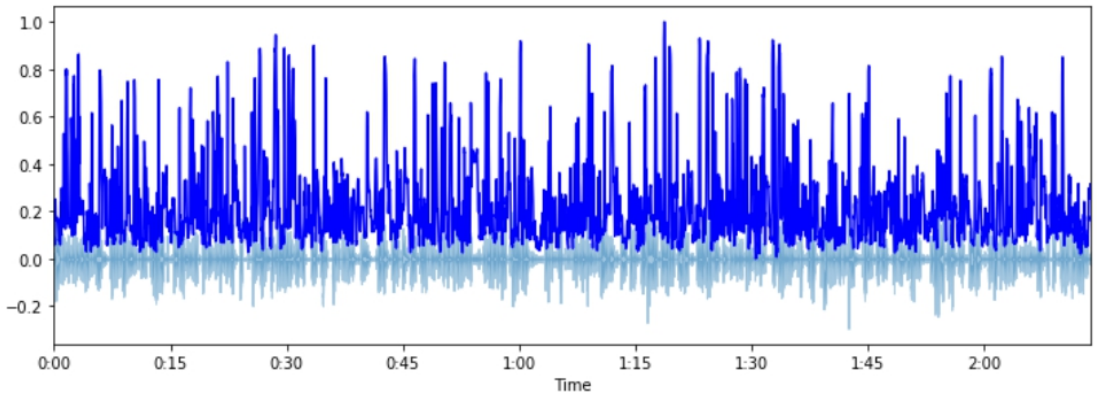
Out[8]: <matplotlib.colorbar.Colorbar at 0x2c0457467c8>



Feature Extraction : Spectral Centroid

In [9]:
```
import sklearn
spectral_centroids = librosa.feature.spectral_centroid(x, sr=sr)[0]
spectral_centroids.shape
(775,)
# Computing the time variable for visualization
plt.figure(figsize=(12, 4))
frames = range(len(spectral_centroids))
t = librosa.frames_to_time(frames)
# Normalising the spectral centroid for visualisation
def normalize(x, axis=0):
    return sklearn.preprocessing.minmax_scale(x, axis=axis)
#Plotting the Spectral Centroid along the waveform
librosa.display.waveplot(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_centroids), color='b')
```

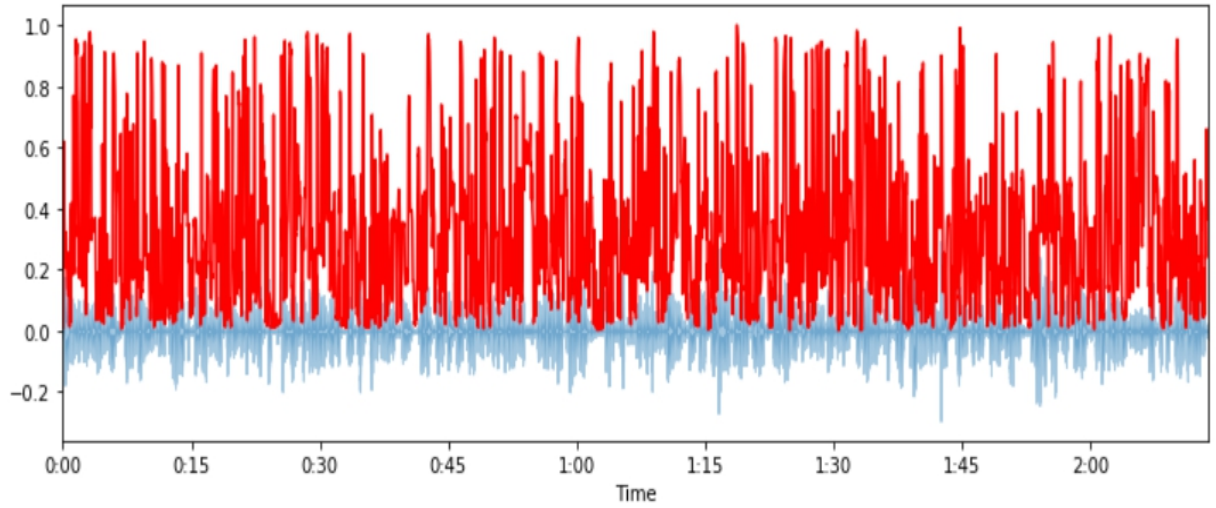
Out[9]:
```
[<matplotlib.lines.Line2D at 0x2c04770f688>]
```



Feature Extraction : Spectral Rolloff

In [10]:
```
spectral_rolloff = librosa.feature.spectral_rolloff(x+0.01, sr=sr)[0]
plt.figure(figsize=(12, 4))
librosa.display.waveplot(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_rolloff), color='r')
```

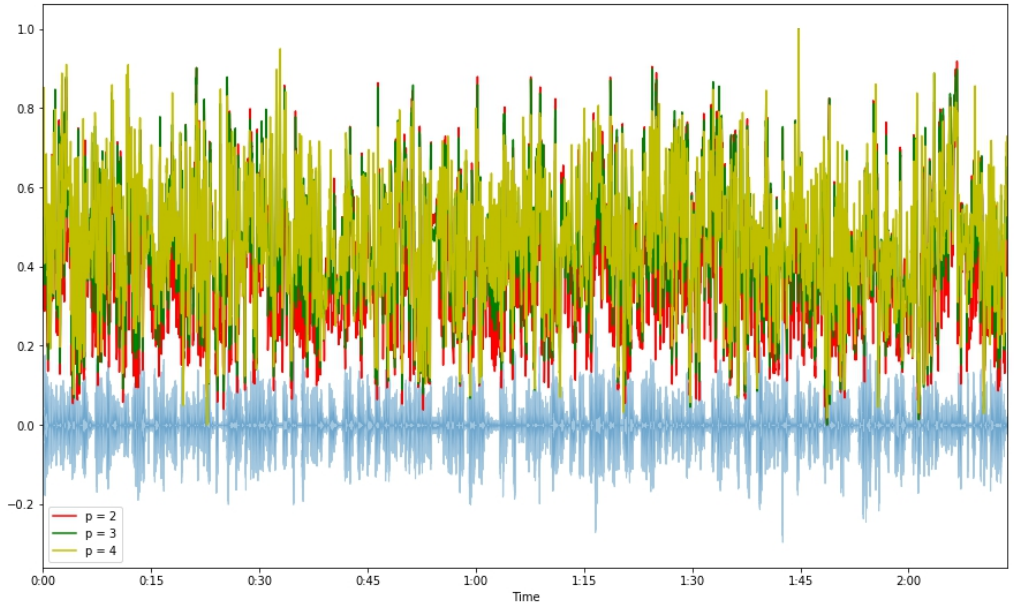
Out[10]:
```
[<matplotlib.lines.Line2D at 0x2c0457b5f48>]
```



Feature Extraction : Spectral Bandwidth

In [11]:
```
spectral_bandwidth_2 = librosa.feature.spectral_bandwidth(x+0.01, sr=sr)[0]
spectral_bandwidth_3 = librosa.feature.spectral_bandwidth(x+0.01, sr=sr, p=3)[0]
spectral_bandwidth_4 = librosa.feature.spectral_bandwidth(x+0.01, sr=sr, p=4)[0]
plt.figure(figsize=(15, 9))
librosa.display.waveplot(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_bandwidth_2), color='r')
plt.plot(t, normalize(spectral_bandwidth_3), color='g')
plt.plot(t, normalize(spectral_bandwidth_4), color='y')
plt.legend(('p = 2', 'p = 3', 'p = 4'))
```

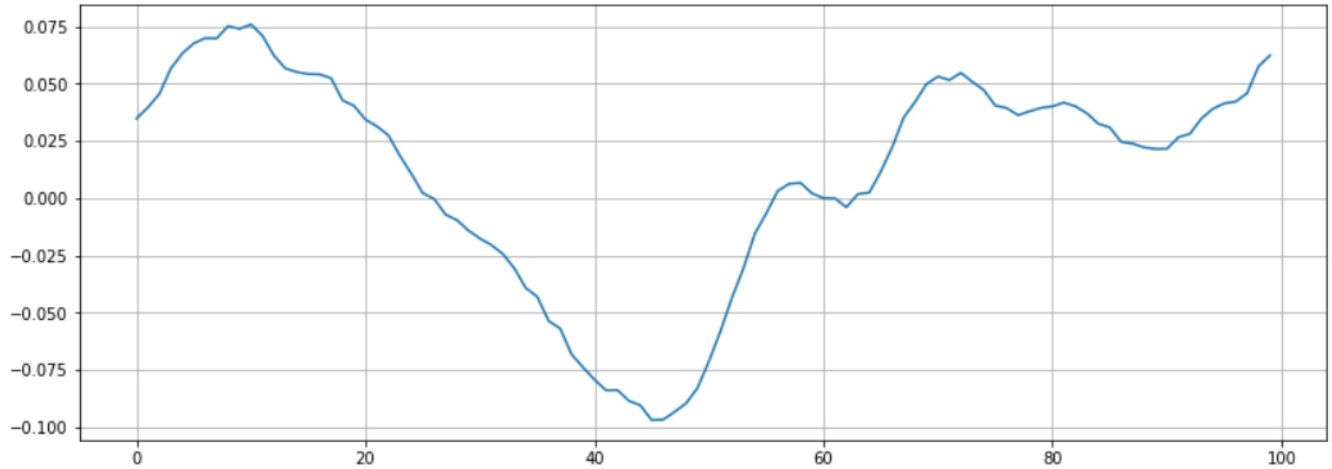
Out[11]: <matplotlib.legend.Legend at 0x2c045a31448>



Feature Extraction : Zero crossing

Zero Crossing

In [13]:
```
n0 = 9000
n1 = 9100
plt.figure(figsize=(14, 5))
plt.plot(x[n0:n1])
plt.grid()
```



In [14]:
```
zero_crossings = librosa.zero_crossings(x[n0:n1], pad=False)
print(sum(zero_crossings))# Number of zero Crossings
```

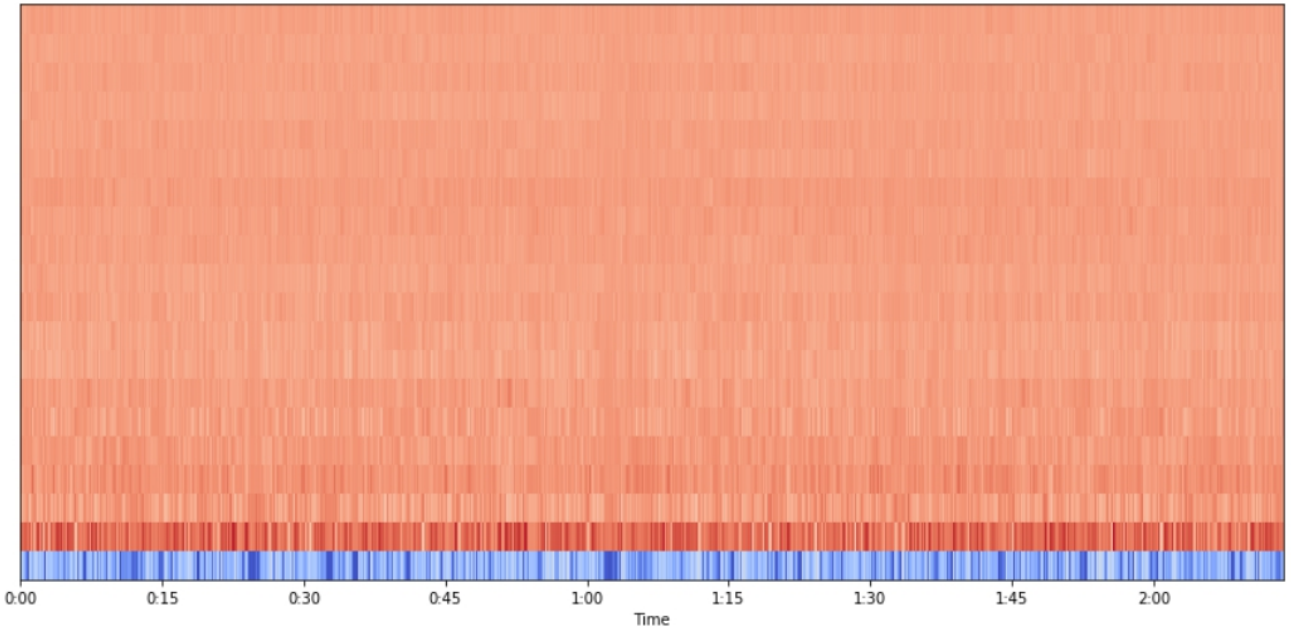
4

Feature Extraction : MFCC

In [12]:
```
mfccs = librosa.feature.mfcc(x, sr=sr)
print(mfccs.shape)
(20, 97)
#Displaying  the MFCCs:
plt.figure(figsize=(15, 7))
librosa.display.specshow(mfccs, sr=sr, x_axis='time')
```

```
(20, 5761)
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2c045a7ec88>
```



Programming Speech Processing: Noise removal (Traditional and Adaptive)

Understanding what has been done

The most basic traditional way in order to denoise a signal is using the features as discussed above to remove the noise. Since for the program we have curbed all of it into a single python script , given below are one liners that explain the function for the noise removal using the traditional methods.

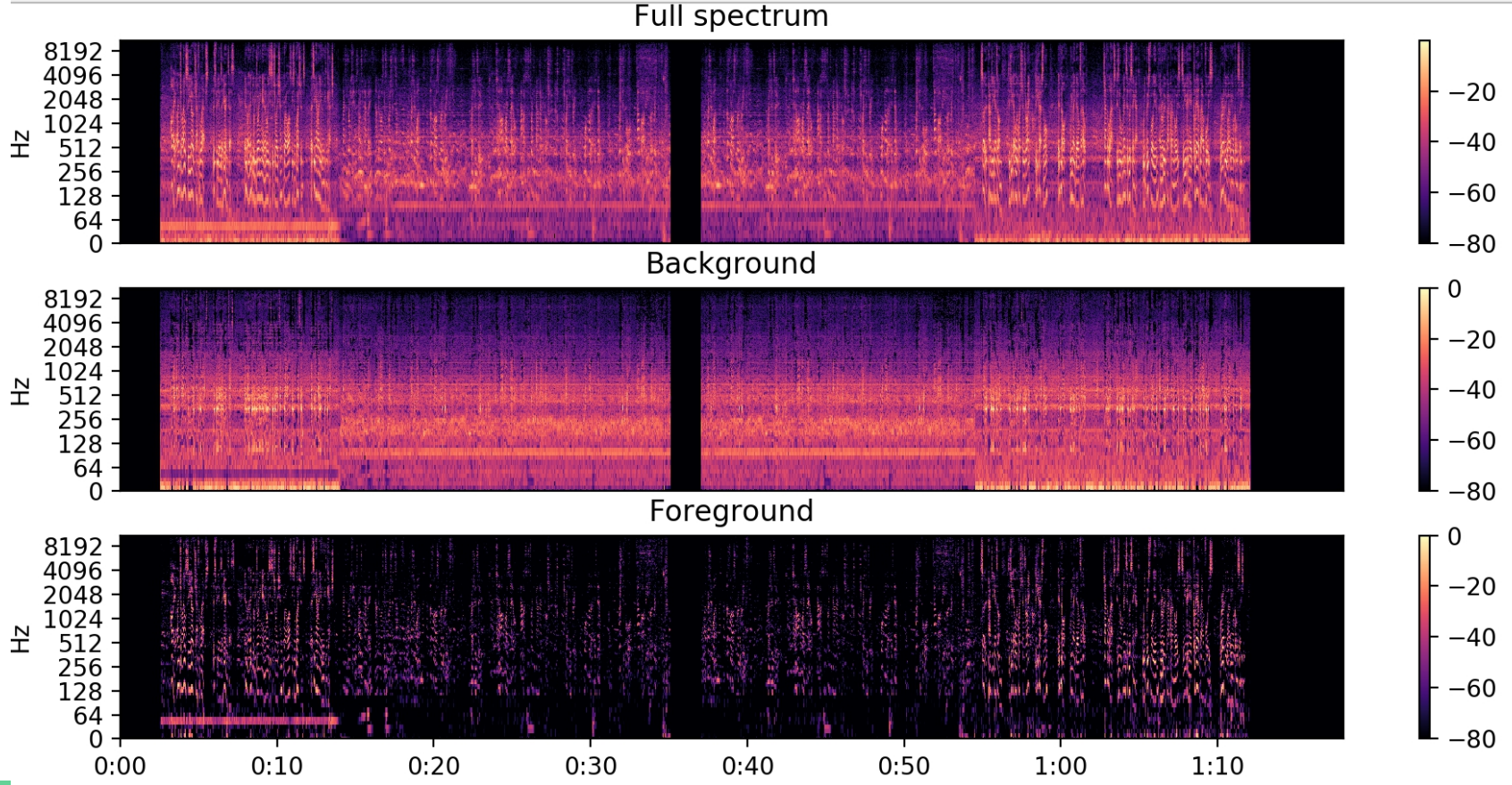
Also the Librosa library comes with an in built in function to remove vocals from a signal which uses the concept of:

This is based on the “REPET-SIM” method of Rafii and Pardo, 2012, but includes a couple of modifications and extensions:

- FFT windows overlap by 1/4, instead of 1/2
- Non-local filtering is converted into a soft mask by Wiener filtering. This is similar in spirit to the soft-masking method used by Fitzgerald, 2012, but is a bit more numerically stable in practice.

Results of librosa vocal seperation(Spectrogram)

The results can be visually seen using the spectrograms obtained. They have also been labelled accordingly.



Next Steps

The above two methods are some of the traditional and adaptive processing done in order to remove noise. As we see it is not very effective.

This call for the usage of Machine learning methods and models to denoise as many techniques like RNN , CNN , autoencoders have proved to be very useful for this process.

References and Acknowledgments

- Reference Papers (sent by Swetha Maam)
- https://www.kdnuggets.com/2020/02/audio-data-analysis-deep-learning-python-part-1.html
- https://librosa.org/doc/latest/index.html
- Wikipedia : For understanding the concepts
- https://librosa.org/librosa_gallery/auto_examples/plot_vocal_separation.html
- https://dodiku.github.io/noise_reduction/
- Kaggle

Thank You