[IIAI30003] Digital Speech Processing

Homework 2

Po-Chuan, Chen
Student ID: 311511052
present90308.ee11@nycu.edu.tw

National Yang Ming Chiao Tung University
October 6, 2023

Contents

1 Speech Analysis & Feature Extraction

First, I record the audio file, and the program will output the waveform.

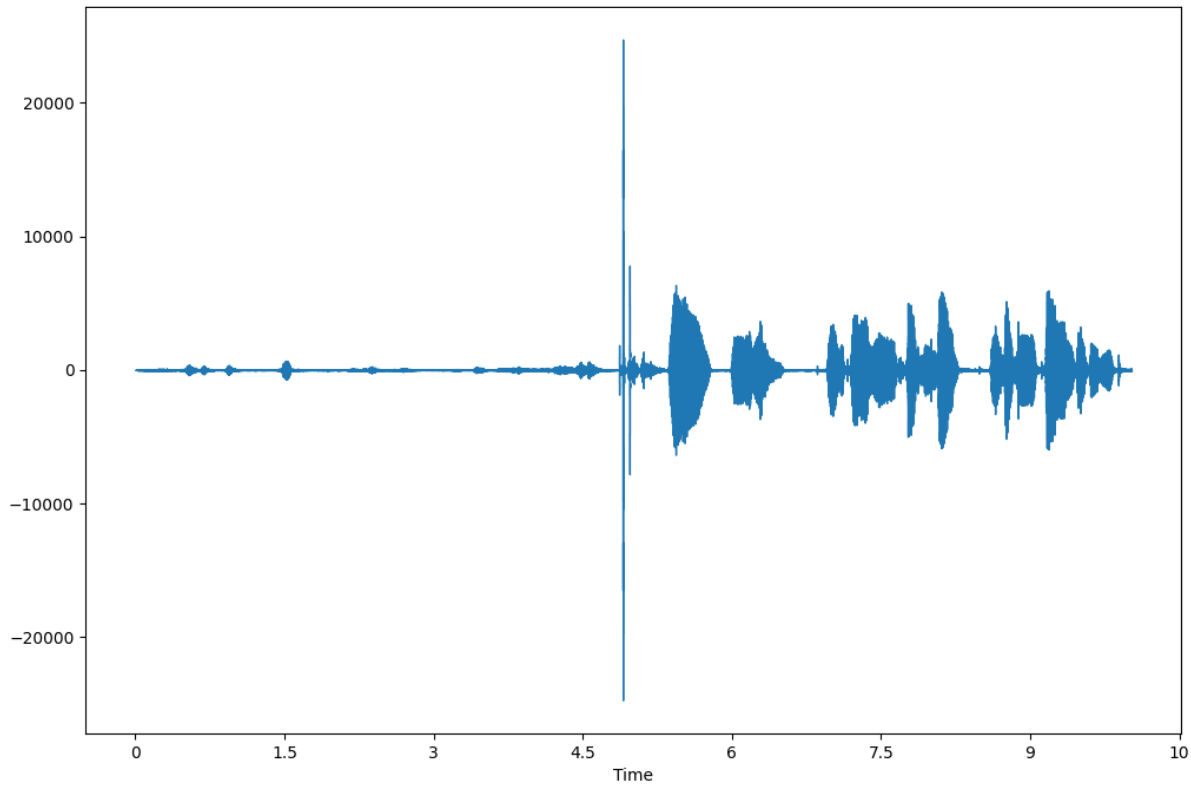

Figure 1: Waveform

With the `librosa.stft`, we can use Short-Time Fourier Transform convert a time-domain signal into its frequency-domain representation.

```
x = librosa.stft(audio, n_fft=2048, hop_length=480) # sampling rate
    =48000, frame_size=32 ms, frame_shift=10 ms
plt.figure(figsize=(12, 8))
librosa.display.specshow(librosa.amplitude_to_db(np.abs(x)), sr=sr,
    y_axis='linear', x_axis='time')
```

Listing 1: Code section for Spectogram

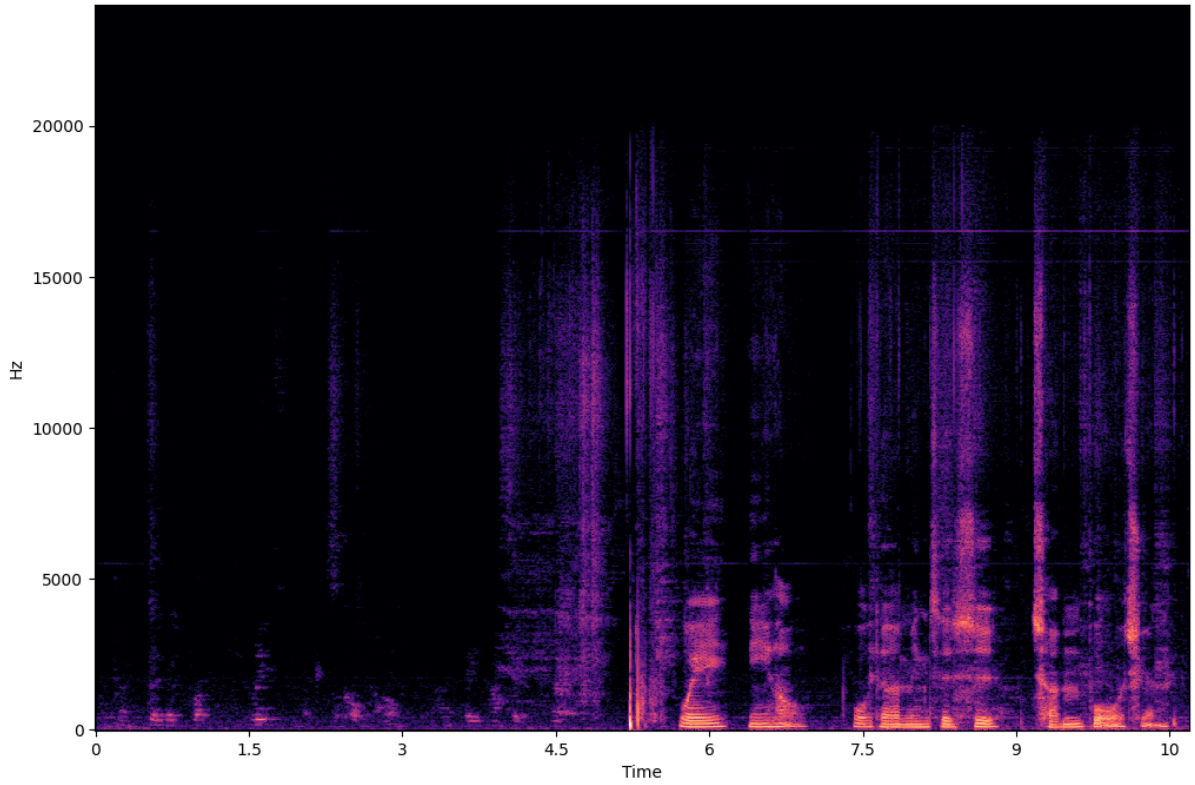

Figure 2: Spectrogram

The left subplot displays the entire pitch information estimated from the audio signal, while the right subplot shows a closer view of the first 100 rows of the pitch information.

```
pitches, magnitudes = librosa.piptrack(y=audio, sr=sr)
plt.figure(figsize=(28, 8))
plt.subplot(1,2,1)
plt.imshow(pitches[:, :], aspect="auto", interpolation="nearest",
    origin="lower")
plt.subplot(1,2,2)
plt.imshow(pitches[:100, :], aspect="auto", interpolation="nearest",
    origin="lower")
```

Listing 2: Code section for Pitch Tracking

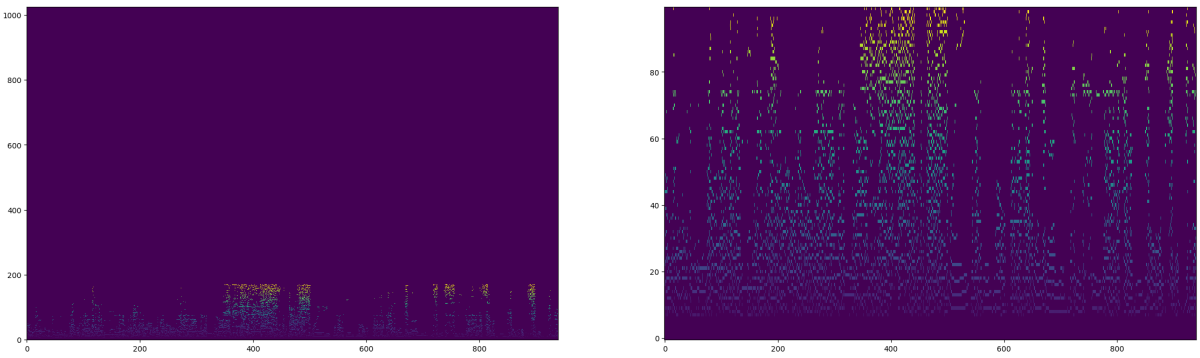

Figure 3: Pitch Tracking

Here, we use the Librosa library to perform some audio processing and visualization tasks. It generates and displays two spectrograms side by side: one for the original audio signal and one for the resampled audio signal.

```
plt.figure(figsize=(24, 8))

plt.subplot(1,2,1)
x = librosa.stft(audio, n_fft=2048, hop_length=480) # # sampling rate
    =16000, frame_size=32 ms, frame_shift=10 ms
librosa.display.specshow(librosa.amplitude_to_db(np.abs(x)), sr=sr,
    y_axis='linear', x_axis='time')

# resampling
target_sr = 16000
audio = librosa.resample(audio, orig_sr=sr, target_sr=target_sr)
sr = target_sr

plt.subplot(1,2,2)
x = librosa.stft(audio, n_fft=2048, hop_length=480) # # sampling rate
    =16000, frame_size=32 ms, frame_shift=10 ms
librosa.display.specshow(librosa.amplitude_to_db(np.abs(x)), sr=sr,
    y_axis='linear', x_axis='time')
```

Listing 3: Code section for Resampling

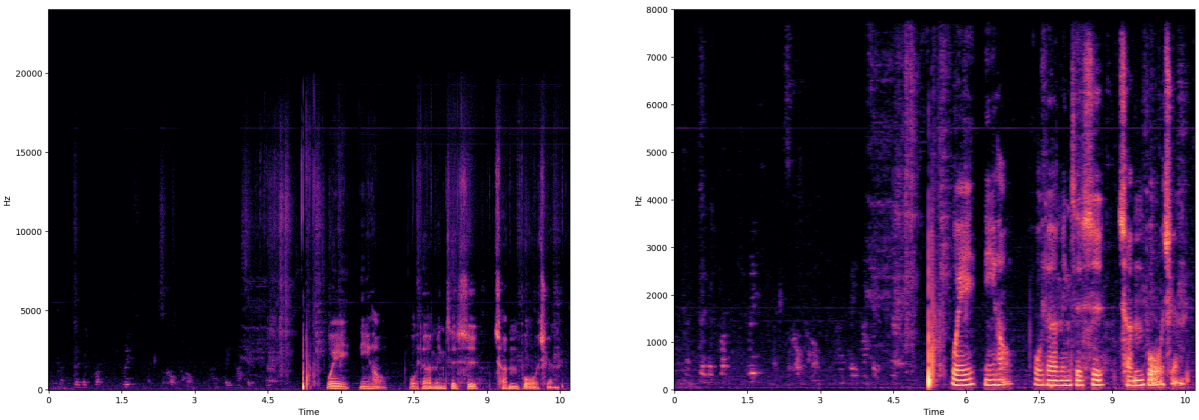

Figure 4: Resampling

The left subplot has a larger `n_fft` value, which provides higher frequency resolution but potentially lower time resolution, while the right subplot has a smaller `n_fft` value, providing higher time resolution but lower frequency resolution.

```
plt.figure(figsize=(24, 8))

plt.subplot(1,2,1)
x = librosa.stft(audio, n_fft=2048, hop_length=80) # # sampling rate
    =16000, frame_size=32 ms, frame_shift=10 ms
librosa.display.specshow(librosa.amplitude_to_db(np.abs(x)), sr=sr,
    y_axis='linear', x_axis='time')

```

```
plt.subplot(1,2,2)
x = librosa.stft(audio, n_fft=128, hop_length=80) # # sampling rate
    =16000, frame_size=32 ms, frame_shift=10 ms
librosa.display.specshow(librosa.amplitude_to_db(np.abs(x)), sr=sr,
    y_axis='linear', x_axis='time')
```

Listing 4: Code section for NarrowBand and WideBand Spectogram

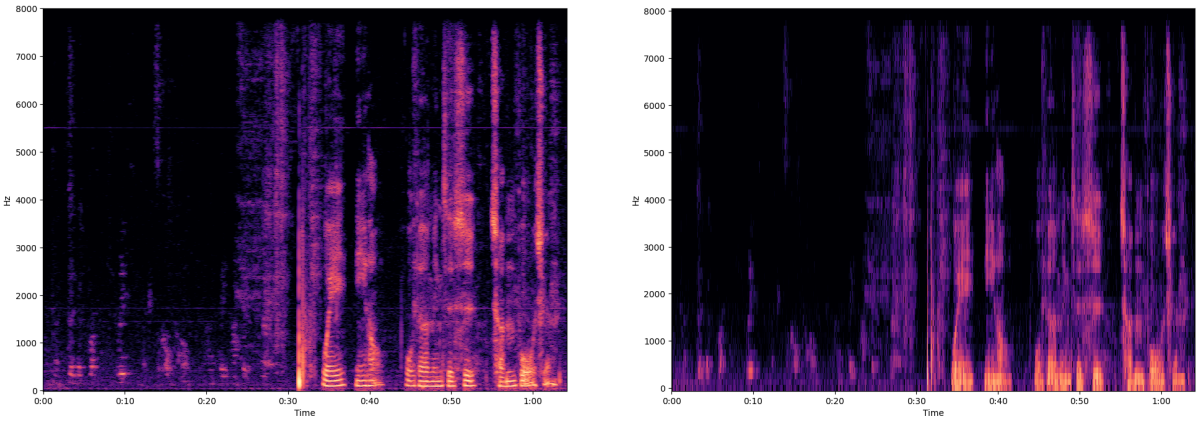

Figure 5: NarrowBand and WideBand Spectogram

Using the Parselmouth library to perform some audio analysis and visualization.

```
snd = parselmouth.Sound(human_sound_file)
snd.resample(new_frequency=16000)
pitch = snd.to_pitch()
# If desired, pre-emphasize the sound fragment before calculating the
    spectrogram
pre_emphasized_snd = snd.copy()
pre_emphasized_snd.pre_emphasize()
spectrogram = pre_emphasized_snd.to_spectrogram(window_length=0.03,
    maximum_frequency=8000)

plt.figure(figsize=(12, 8))
draw_spectrogram(spectrogram)
plt.twinx()
draw_pitch(pitch)
plt.xlim([snd.xmin, snd.xmax])
plt.show() # or plt.savefig("spectrogram_0.03.pdf")
```

Listing 5: Code section for Pitch Contour Extraction

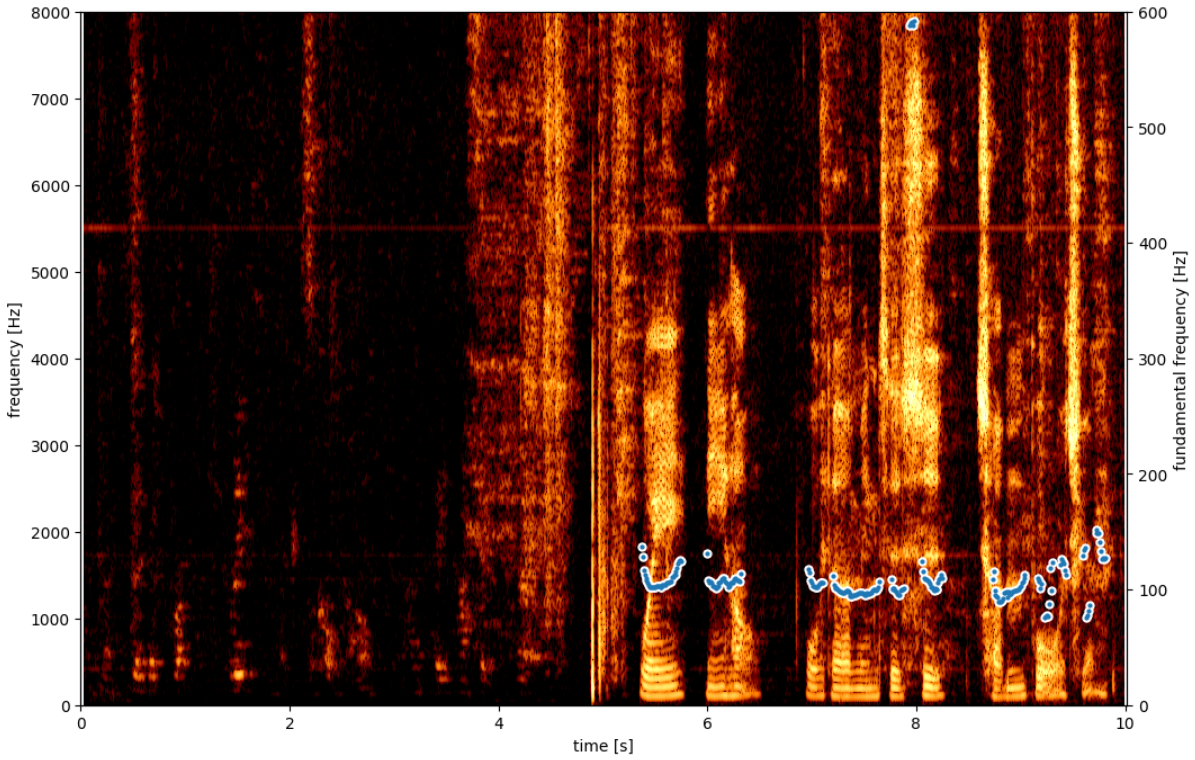

Figure 6: Pitch Contour Extraction

Here, the code generates and displays two spectrograms side by side: a linear power spectrogram on the left and a mel-scaled power spectrogram on the right.

```
plt.figure(figsize=(24, 8))

plt.subplot(1,2,1)
x = librosa.stft(audio, n_fft=2048, hop_length=80) # # sampling rate
    =16000, frame_size=32 ms, frame_shift=10 ms
librosa.display.specshow(librosa.amplitude_to_db(np.abs(x)), sr=sr,
    y_axis='linear', x_axis='time')
# Put a descriptive title on the plot
plt.title('linear power spectrogram')
# draw a color bar
plt.colorbar(format='%+02.0f dB')


plt.subplot(1,2,2)
# Let's make and display a mel-scaled power (energy-squared)
    spectrogram
S = librosa.feature.melspectrogram(y=audio, sr=sr, n_mels=128)
# Convert to log scale (dB). We'll use the peak power (max) as
    reference.
log_S = librosa.power_to_db(S, ref=np.max)
# Display the spectrogram on a mel scale
# sample rate and hop length parameters are used to render the time
    axis
librosa.display.specshow(log_S, sr=sr, x_axis='time', y_axis='mel')
# Put a descriptive title on the plot
plt.title('mel power spectrogram')
```

```
# draw a color bar
plt.colorbar(format='%+02.0f dB')

# Make the figure layout compact
plt.tight_layout()
```

Listing 6: Code section for Mel-Scaled Spectogram

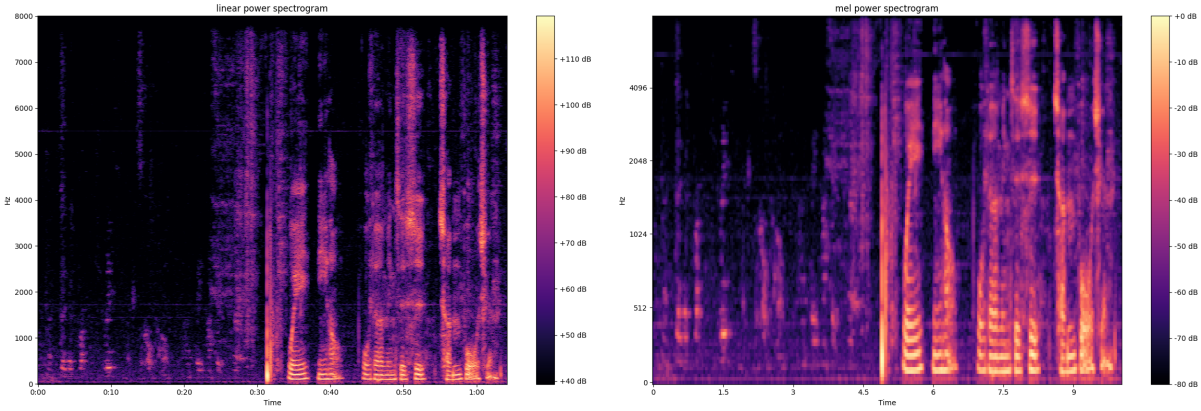

Figure 7: Mel-Scaled Spectogram

Now it focusing on extracting Mel-frequency cepstral coefficients (MFCCs) and their derivatives. The resulting coefficients are standardized, and visualizations of these coefficients are displayed in separate subplots.

```
# Next, we'll extract the top 13 Mel-frequency cepstral coefficients (
    MFCCs)
mfcc        = librosa.feature.mfcc(S=log_S, n_mfcc=13)
print(mfcc.shape)
print(mfcc)

# zero mean, unit variance
mfcc        = preprocessing.scale(mfcc,axis=1)

# Let's pad on the first and second deltas while we're at it
delta_mfcc  = librosa.feature.delta(mfcc)
delta2_mfcc = librosa.feature.delta(mfcc, order=2)

# How do they look?  We'll show each in its own subplot
plt.figure(figsize=(12.5, 12))

plt.subplot(3,1,1)
librosa.display.specshow(mfcc)
plt.ylabel('MFCC')
plt.colorbar()

plt.subplot(3,1,2)
librosa.display.specshow(delta_mfcc)
plt.ylabel('MFCC-$\Delta$')
plt.colorbar()
```

```
25
26 plt.subplot(3,1,3)
27 librosa.display.specshow(delta2_mfcc, sr=sr, x_axis='time')
28 plt.ylabel('MFCC-$\Delta^2$')
29 plt.colorbar()
30
31 plt.tight_layout()
32
33 # For future use, we'll stack these together into one matrix
34 M = np.vstack([mfcc, delta_mfcc, delta2_mfcc])
```

Listing 7: Code section for MFCCs

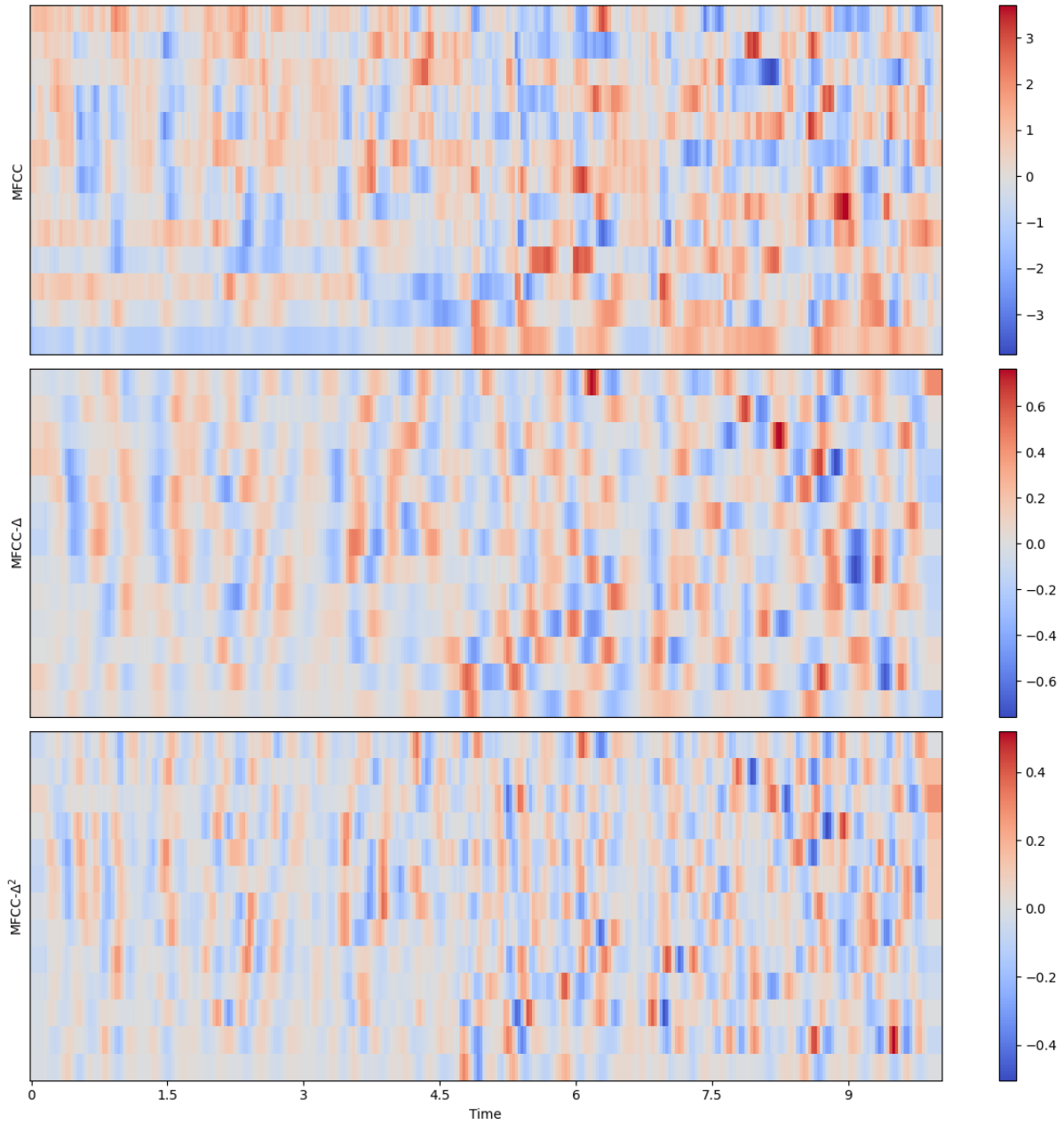

Figure 8: MFCCs

The last step, it uses the Parselmouth library to perform audio analysis and visualization for two different audio files. Two of the plots are a spectrogram and pitch contour for a different audio file.

```
plt.figure(figsize=(28, 8))

plt.subplot(1,2,1)
snd = parselmouth.Sound(human_sound_file)
pitch = snd.to_pitch()
# If desired, pre-emphasize the sound fragment before calculating the spectrogram
pre_emphasized_snd = snd.copy()
pre_emphasized_snd.pre_emphasize()
spectrogram = pre_emphasized_snd.to_spectrogram(window_length=0.03, maximum_frequency=8000)

draw_spectrogram(spectrogram)
plt.twinx()
draw_pitch(pitch)
plt.xlim([snd.xmin, snd.xmax])

plt.subplot(1,2,2)
snd = parselmouth.Sound('output.wav')
pitch = snd.to_pitch()
# If desired, pre-emphasize the sound fragment before calculating the spectrogram
pre_emphasized_snd = snd.copy()
pre_emphasized_snd.pre_emphasize()
spectrogram = pre_emphasized_snd.to_spectrogram(window_length=0.03, maximum_frequency=8000)

draw_spectrogram(spectrogram)
plt.twinx()
draw_pitch(pitch)
plt.xlim([snd.xmin, snd.xmax])
```

Listing 8: Code section for Rubber Band

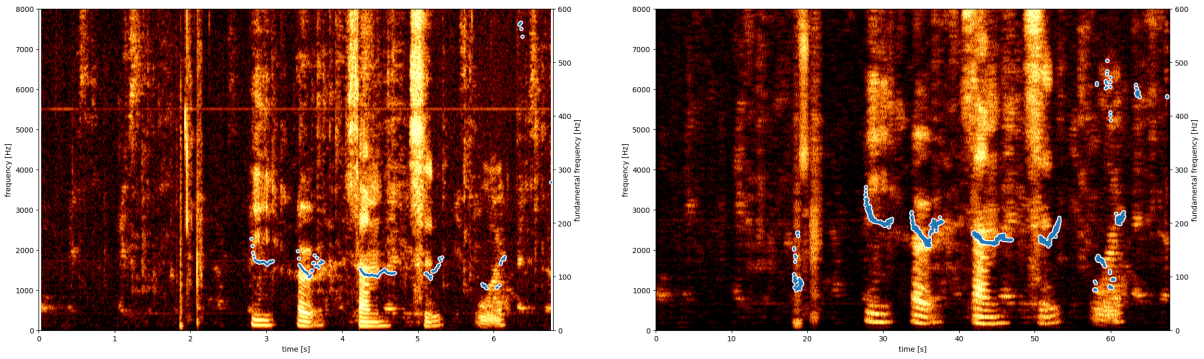

Figure 9: Rubber Band

2 Audio Data Augmentation

The source of the audio is Jay Chou's song, In the Name of the Father.

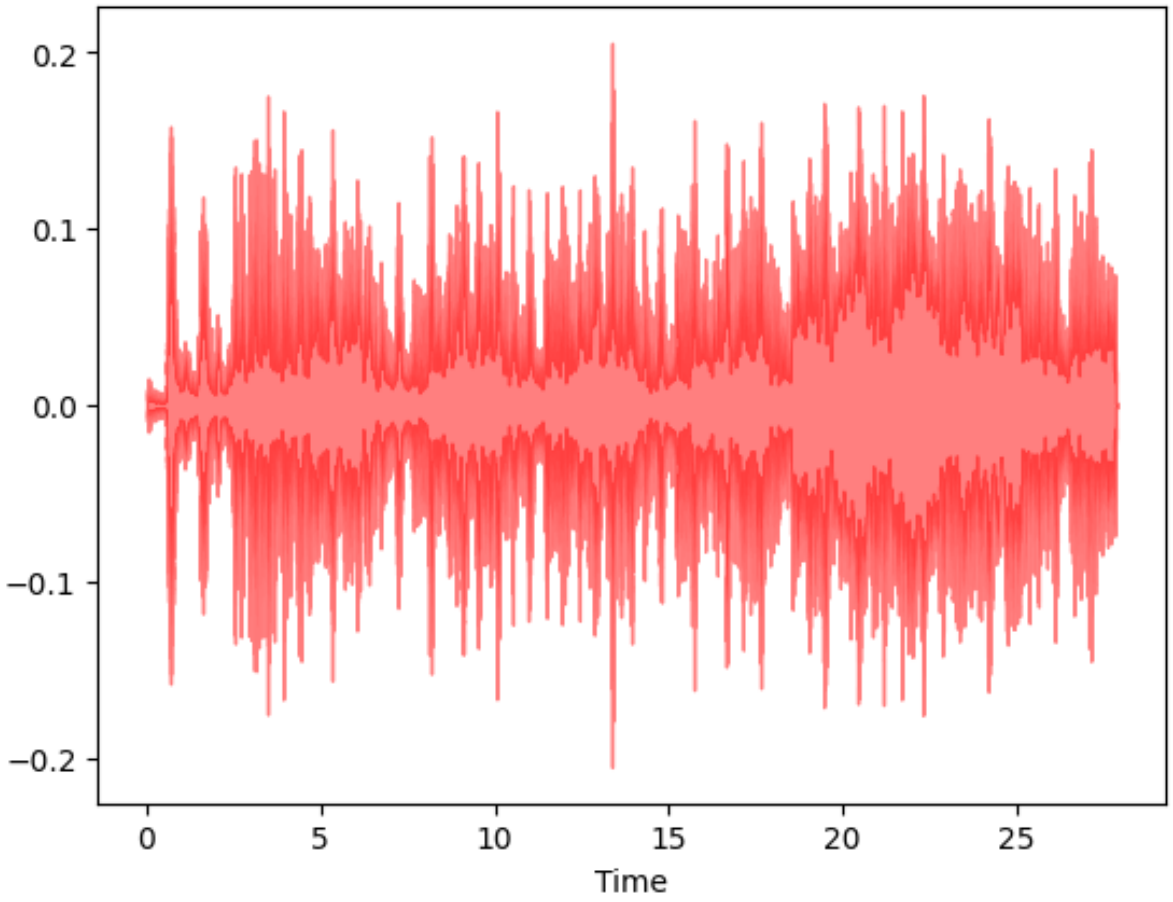

Figure 10: Waveform

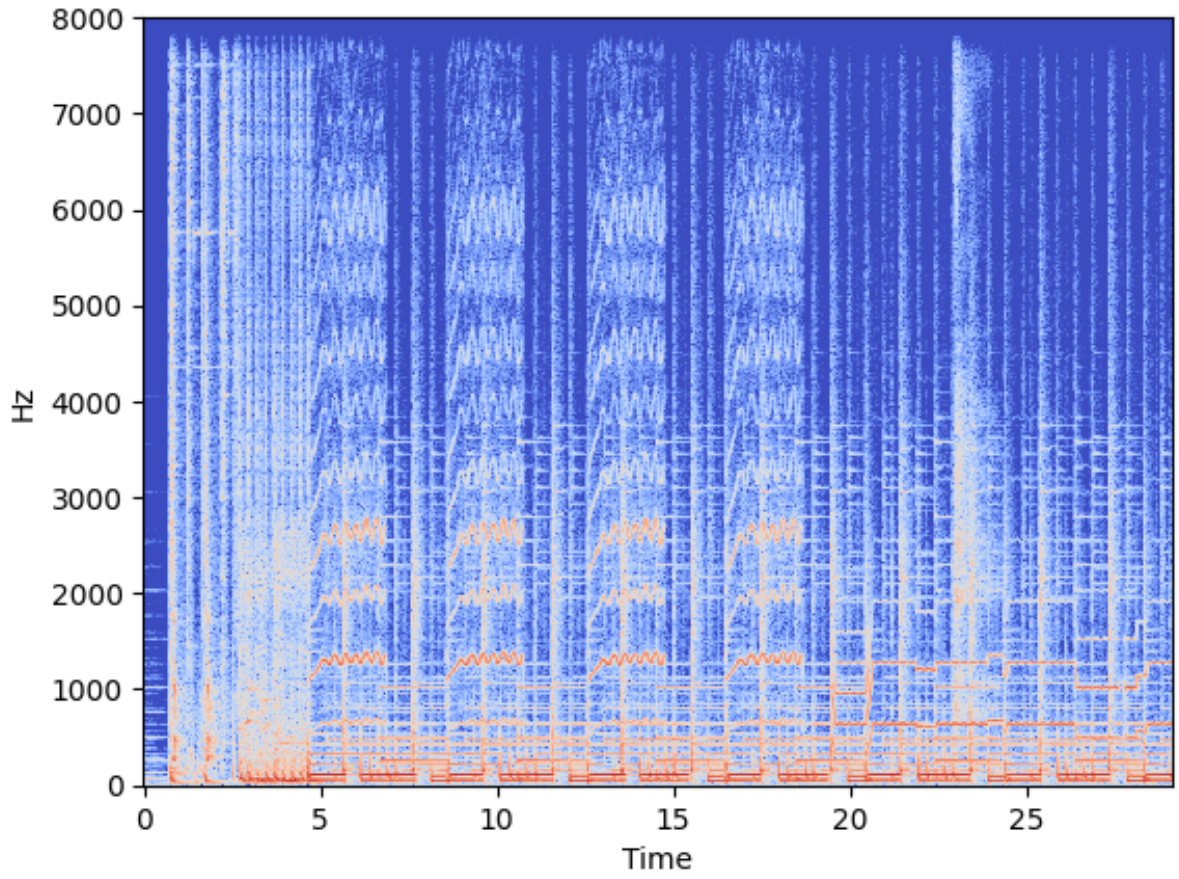

Figure 11: Spectrogram

2.1 Nlpaug

PitchAug

```
aug = naa.PitchAug(sampling_rate=sr, factor=(2,3))
augmented_data = np.array(aug.augment(data))

librosa_display.waveshow(data, sr=sr, color='b',alpha=0.5)
librosa_display.waveshow(augmented_data, sr=sr, color='g', alpha=0.25)
write_wav("jay-PitchAug.wav", sr, augmented_data)
plt.tight_layout()
plt.show()
```

Listing 9: Code section for PitchAug

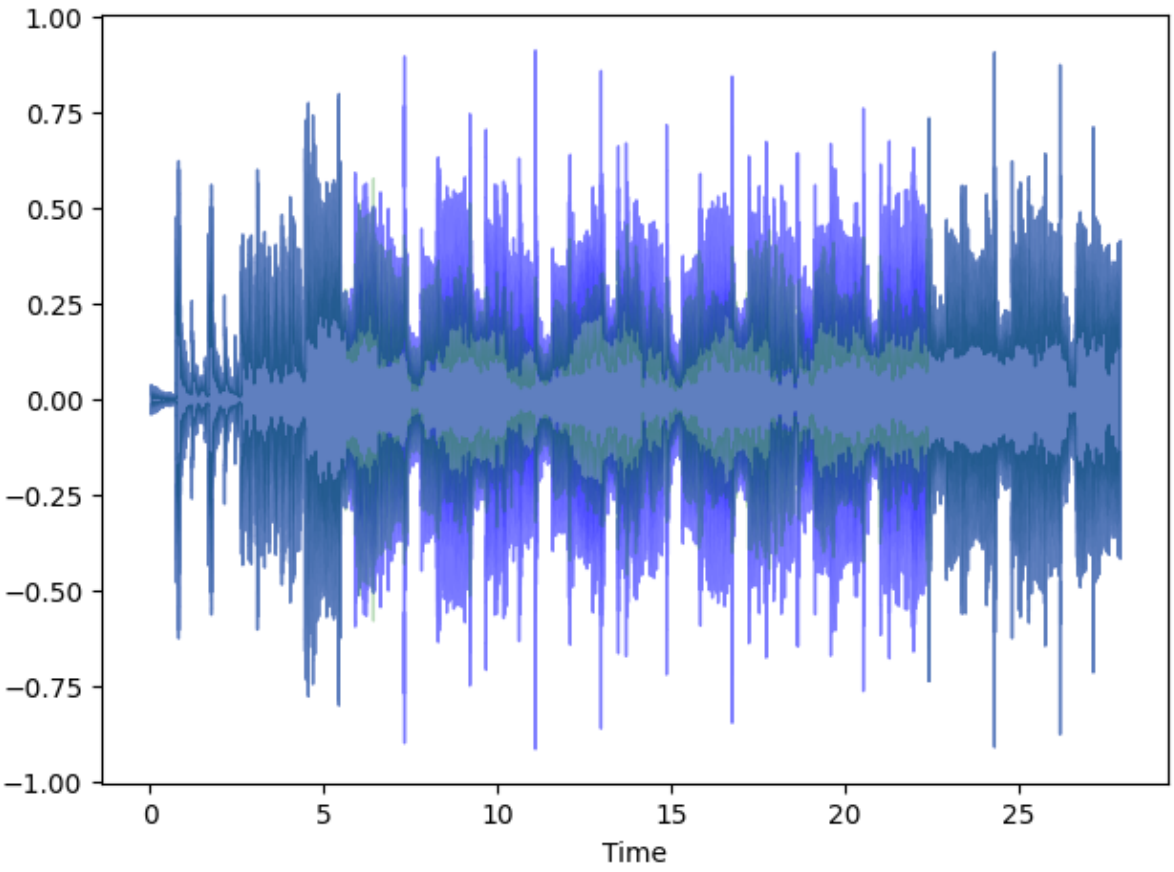

Figure 12: PitchAug

ShiftAug

```
aug = naa.ShiftAug(sampling_rate=sr)
augmented_data = np.array(aug.augment(data))

librosa_display.waveshow(data, sr=sr,color='b', alpha=0.5)
librosa_display.waveshow(augmented_data, sr=sr, color='g', alpha=0.25)
write_wav("jay-ShiftAug.wav", sr, augmented_data)
plt.tight_layout()
plt.show()
```

Listing 10: Code section for ShiftAug

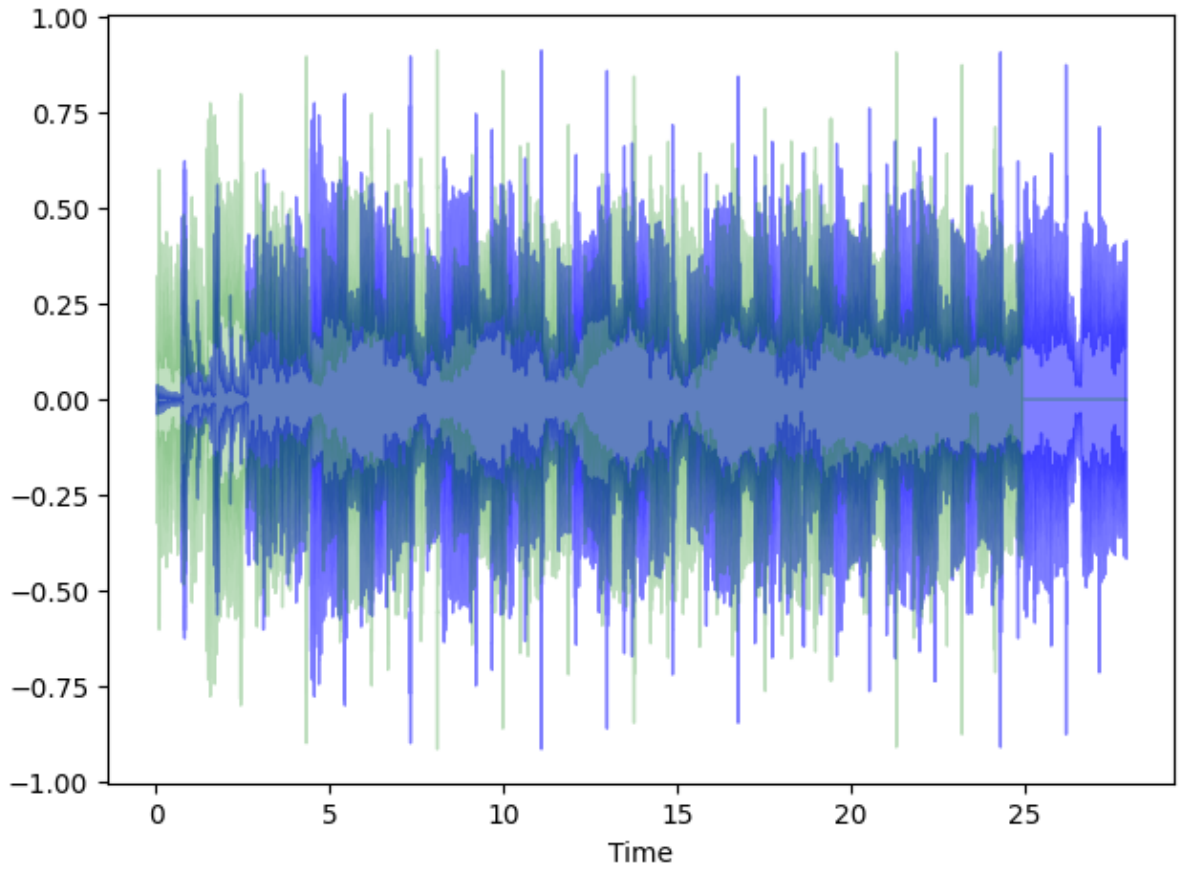

Figure 13: ShiftAug

SpeedAug

```
aug = naa.SpeedAug()
augmented_data = np.array(aug.augment(data))

librosa_display.waveshow(data, sr=sr, color='b',alpha=0.5)
librosa_display.waveshow(augmented_data, sr=sr, color='g', alpha=0.5)
write_wav("jay-SpeedAug.wav", sr, augmented_data)
plt.tight_layout()
plt.show()
```

Listing 11: Code section for SpeedAug

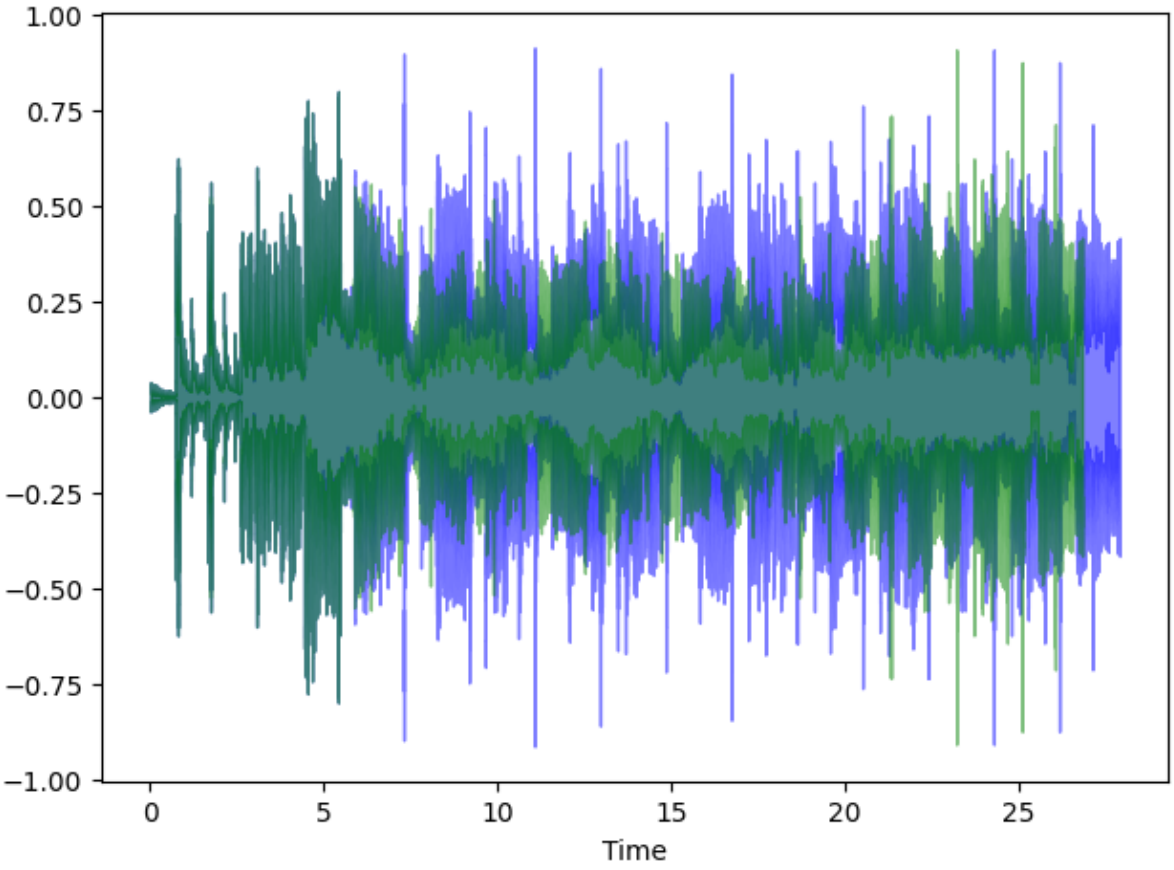

Figure 14: Rubber Band

VtlpAug

```
aug = naa.VtlpAug(sampling_rate=sr)
augmented_data = np.array(aug.augment(data))
write_wav("jay-VtlpAug.wav", sr, augmented_data)
AudioVisualizer.freq_power('VTLP Augmenter', data, sr ,augmented_data)
```

Listing 12: Code section for VtlpAug

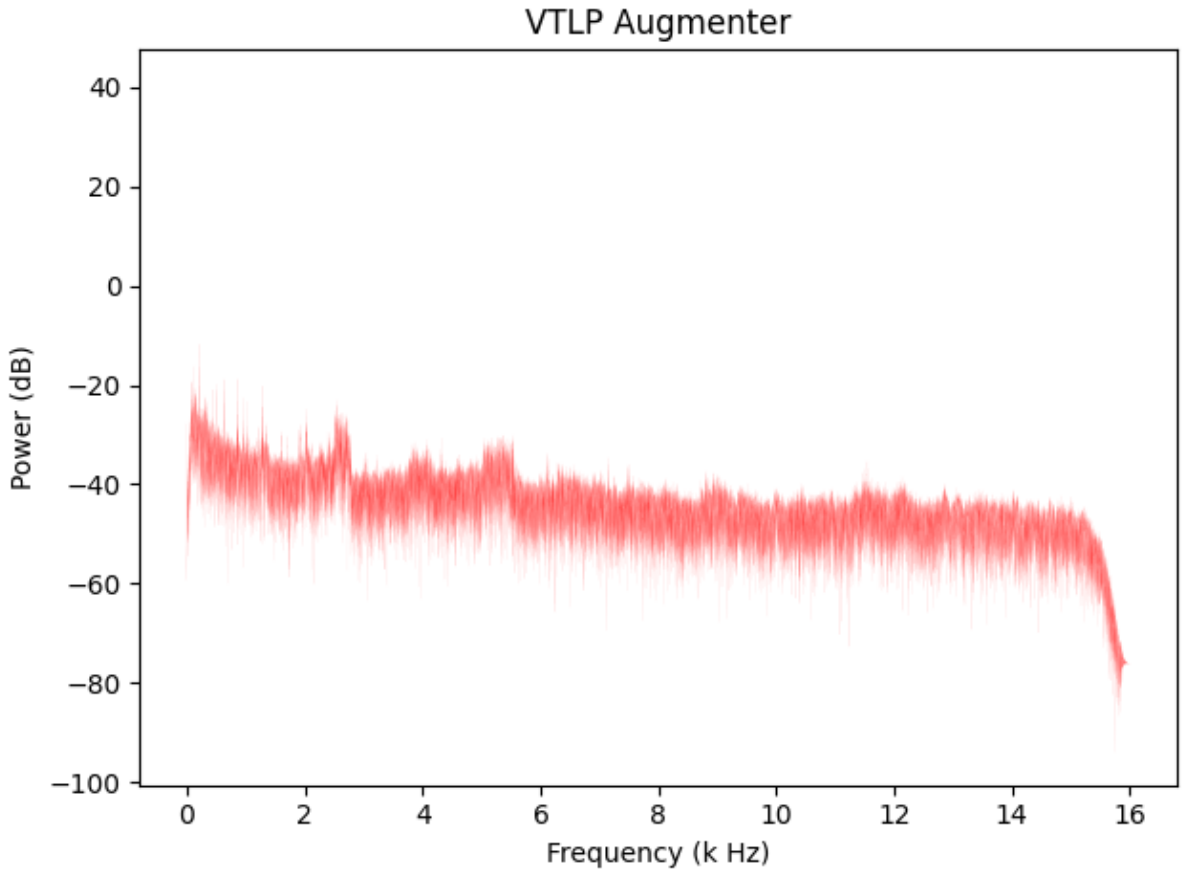

Figure 15: VtlpAug

2.2 Audiomentations

TimeMask

```
from audiomentations import TimeMask

transform = TimeMask(
    min_band_part=0.1,
    max_band_part=0.15,
    fade=True,
    p=1.0,
)

my_waveform_ndarray = np.array(data)

augmented_sound = transform(my_waveform_ndarray, sample_rate=16000)

librosa_display.waveshow(augmented_sound, sr=sr, color='b', alpha=0.25)
write_wav("jay-TimeMask.wav", sr, augmented_sound)
plt.tight_layout()
plt.show()
```

Listing 13: Code section for TimeMask

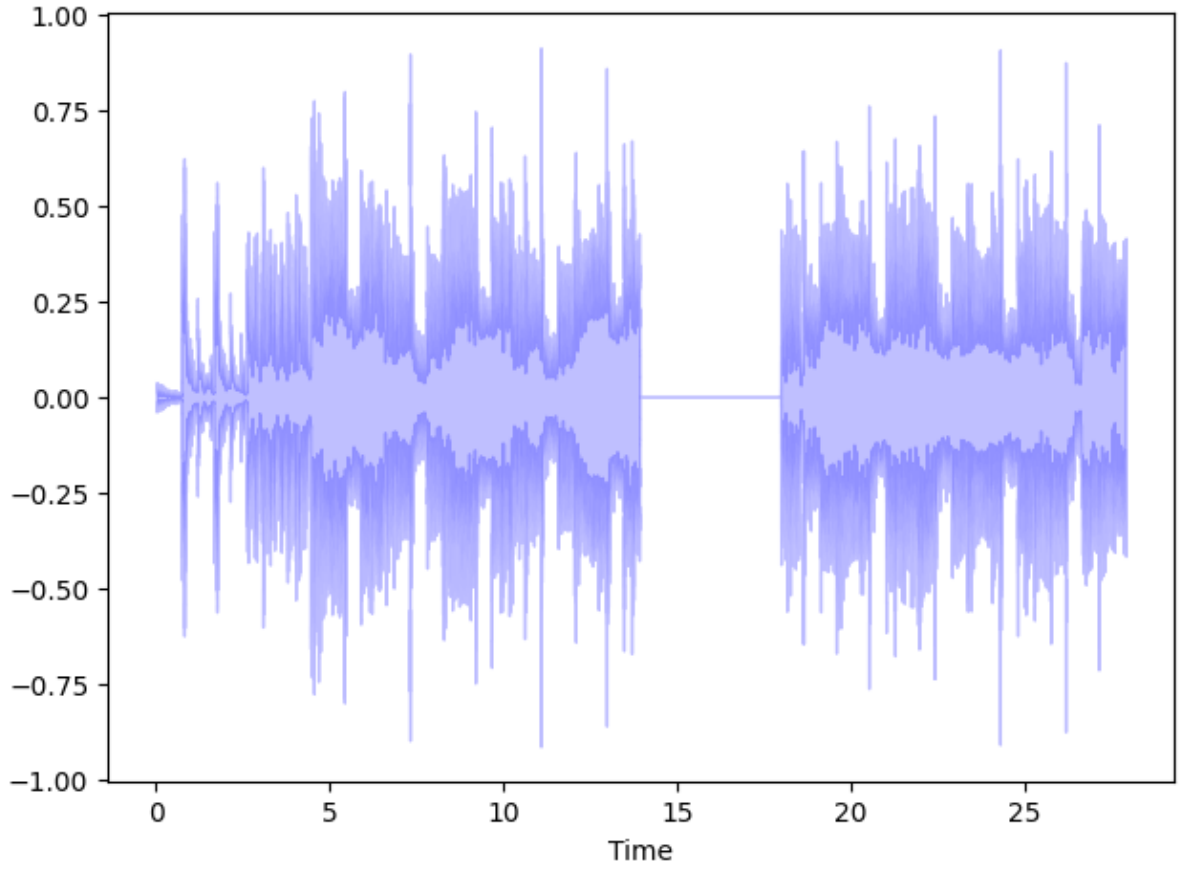

Figure 16: TimeMask

FrequencyMask

Here we use SpecFrequencyMask.

```
from audiomentations import SpecFrequencyMask
import numpy as np

transform = SpecFrequencyMask(p=0.5)

my_waveform_ndarray = np.array(data)

# Augment/transform/perturb the spectrogram
augmented_spectrogram = transform(my_waveform_ndarray)

librosa_display.waveshow(augmented_spectrogram, sr=sr, color='b', alpha
    =0.25)
write_wav("jay-SpecFrequencyMask.wav", sr, augmented_spectrogram)
plt.tight_layout()
plt.show()
```

Listing 14: Code section for FrequencyMask

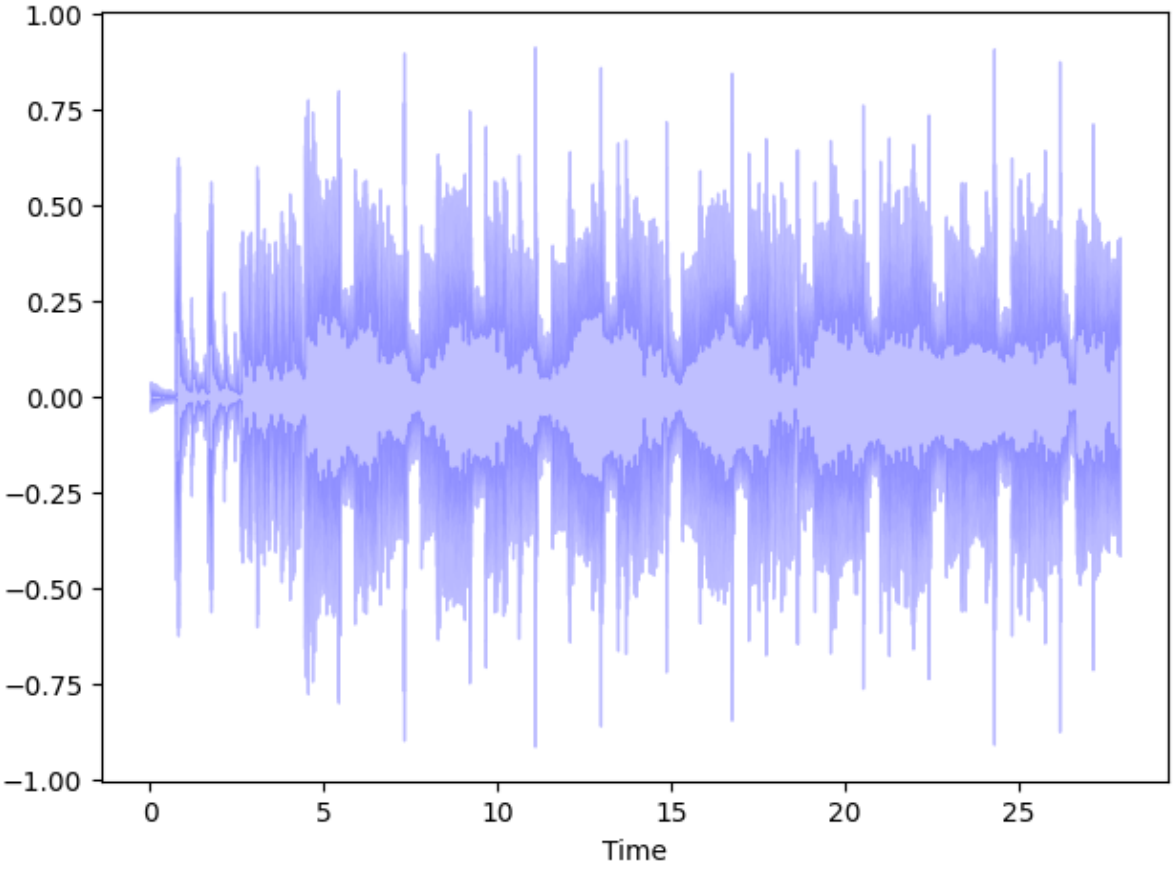

Figure 17: FrequencyMask

ApplyImpulseResponse

```
from audiomentations import ApplyImpulseResponse

my_waveform_ndarray = np.array(data)

transform = ApplyImpulseResponse(ir_path=path_2, p=1.0)

augmented_sound = transform(my_waveform_ndarray, sample_rate=48000)

librosa_display.waveshow(augmented_sound, sr=sr, color='b', alpha=0.25)
write_wav("jay-ApplyImpulseResponse.wav", sr, augmented_sound)
plt.tight_layout()
plt.show()
```

Listing 15: Code section for ApplyImpulseResponse

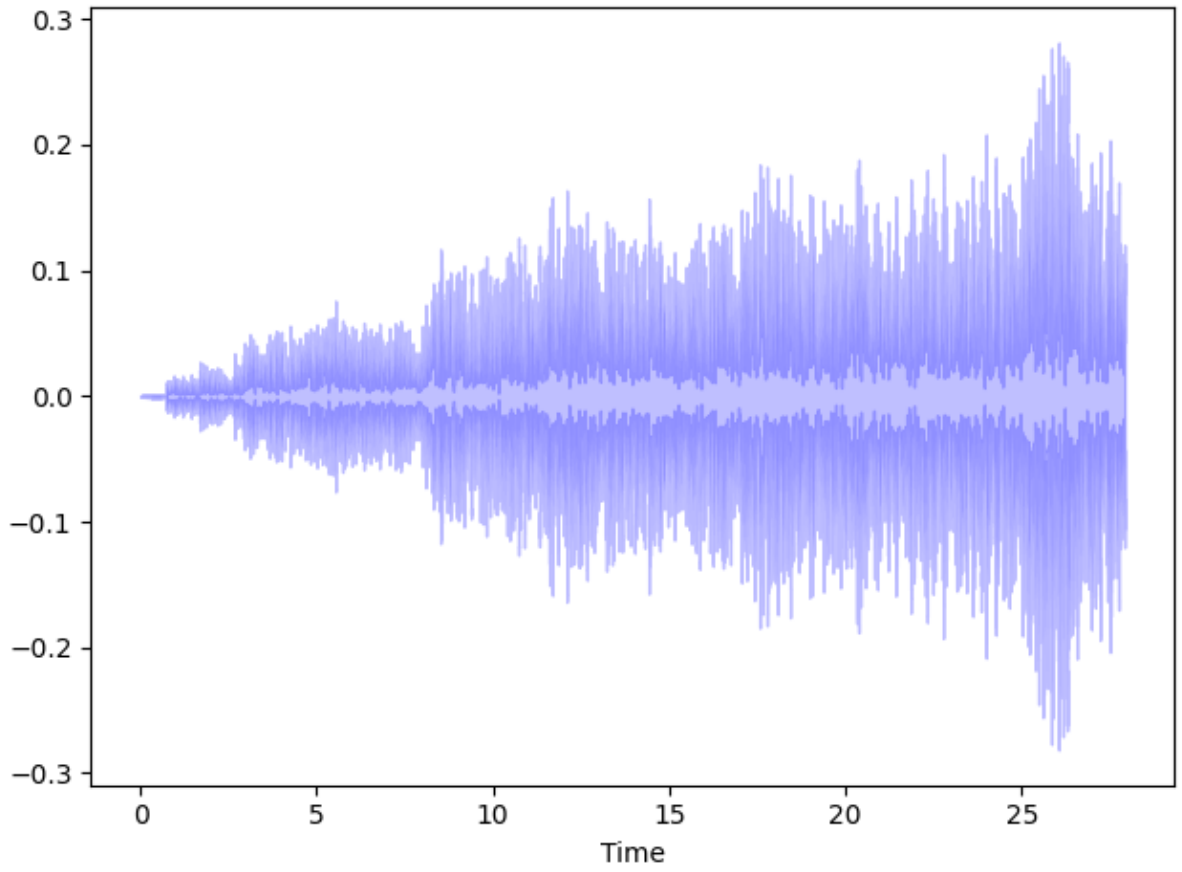

Figure 18: ApplyImpulseResponse

AddBackgroundNoise

```
from audiomentations import AddBackgroundNoise, PolarityInversion

my_waveform_ndarray = np.array(data)

transform = AddBackgroundNoise(
    sounds_path=path_2,
    min_snr_in_db=3.0,
    max_snr_in_db=30.0,
    noise_rms="relative",
    noise_transform=PolarityInversion(),
    p=0.4
)

augmented_sound = transform(my_waveform_ndarray, sample_rate=16000)

librosa_display.waveshow(augmented_sound, sr=sr, color='b', alpha=0.25)
write_wav("jay-AddBackgroundNoise.wav", sr, augmented_sound)
plt.tight_layout()
plt.show()
```

Listing 16: Code section for AddBackgroundNoise

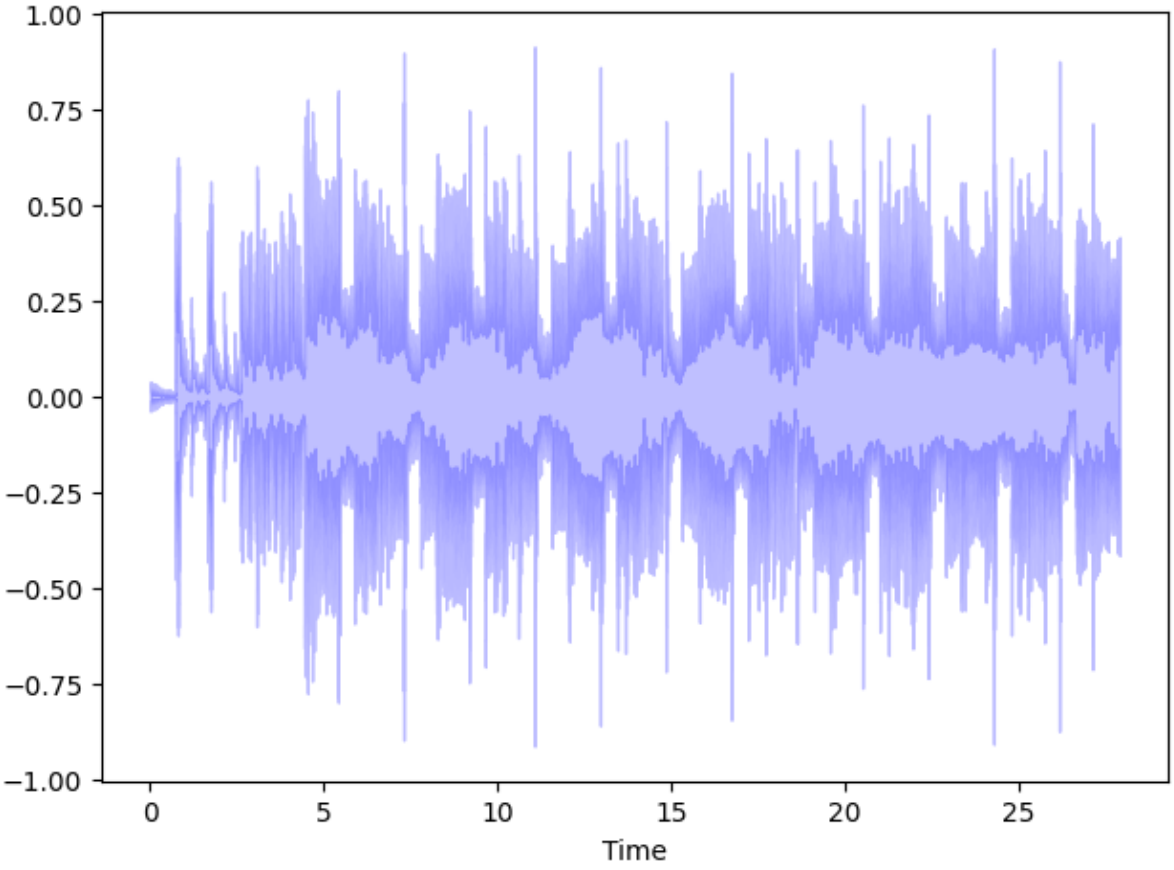

Figure 19: AddBackgroundNoise