Visualizing audio signals

Let's see how to visualize an audio signal. We will learn how to read an audio signal from a file and work with it. This will help us understand how an audio signal is structured. When audio files are recorded using a microphone, they are sampling the actual audio signals and storing the digitized versions. The real audio signals are continuous valued waves, which means we cannot store them as they are. We need to sample the signal at a certain frequency and convert it into discrete numerical form.

Most commonly, speech signals are sampled at 44,100 Hz. This means that each second of the speech signal is broken down into 44,100 parts and the values at each of these timestamps is stored in an output file. We save the value of the audio signal every 1/44,100 seconds. In this case, we say that the sampling frequency of the audio signal is 44,100 Hz. By choosing a high sampling frequency, it will appear that the audio signal is continuous when humans listen to it. Let's go ahead and visualize an audio signal.

**Create a new Python file and import the following packages:**

import numpy as np

import matplotlib.pyplot as plt

from scipy.io import wavfile

Read the input audio file using the wavefile.read method. It returns two values – sampling frequency and the audio signal:

**# Read the audio file**

sampling\_freq, signal = wavfile.read('random\_sound.wav')

Print the shape of the signal, the datatype, and the duration of the audio signal:

***# Display the params***

print('\nSignal shape:', signal.shape)

print('Datatype:', signal.dtype)

print('Signal duration:', round(signal.shape[0] / float(sampling\_freq), 2), 'seconds')

Normalize the signal:

***# Normalize the signal***

signal = signal / np.power(2, 15)

Extract the first 50 values from the numpy array for plotting:

***# Extract the first 50 values***

signal = signal[:50]

Construct the time axis in seconds:

**# Construct the time axis in milliseconds**

time\_axis = 1000 \* np.arange(0, len(signal), 1) / float(sampling\_freq)

Plot the audio signal:

# Plot the audio signal

plt.plot(time\_axis, signal, color='black')

plt.xlabel('Time (milliseconds)')

plt.ylabel('Amplitude')

plt.title('Input audio signal')

plt.show()

The full code is given in the file audio\_plotter.py. If you run the code, you will see the following screenshot:

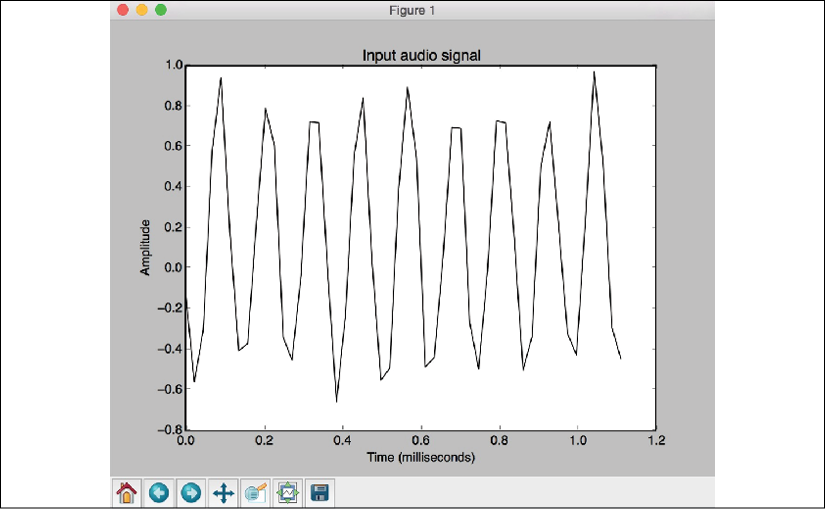


Figure 1: Visualization of input audio signal

The preceding screenshot shows the first 50 samples of the input audio signal. You will see the following output:

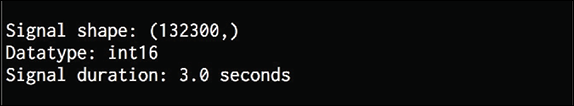


Figure 2: Input audio signal output

The output printed in the preceding figure shows the information that we extracted from the signal.