Application Exploration

Interactive Learning of Signal Processing Through Music

Swadhin Dash, Harsh Shah, Harshraj Chaudhari

October 26, 2022

Contents

0.1 Overview

- This report summarizes how music processing can bridge the gap between learning signal processing concepts in class and application of the knowledge to real life problems
- Introduction to application of Fourier Analysis to analyse properties of music
- We discuss how various software tools can make music processing much more intuitive and fun for students
- We then discuss how to use the Python package *librosa* to analyse audio files and applying signal processing concepts to enable broader experimentation

0.2 Motive

Music has always been an integral part of human civilization as a form of recreation. With advent of digital age, we can enjoy music anytime and from anywhere thanks to digital representation of audio signals. Signal processing plays a major role in music processing and the vast amount of math involved can scare newcomers. So intuitive introduction to music processing using software tools can help students apply concepts taught in class and attain a better understanding of the scale at which signal processing operates in the real world. Instructors can incorporate music based examples into an existing course on signal processing, motivating them to pursue further research and experimentation in the domain.

An introductory course on signal processing covers a range of topics including Types of Systems, Fourier Transform, Convolution, Laplace Transform, z-Transform, Filtering, Sampling and so on. All these concepts can be made easier to understand if hands-on application is allowed for. Here, we focus on audio representation of acoustic waves generated by instrument or human voice which are transmitted as pressure variations through air. We will stick to music(preferably) as human speech has a lot of **noise** that might be hard to analyse in the beginning.

0.3 Basic concepts in Music Processing

Pitch : The frequency at which our vocal chords or the strings of an instrument vibrate is referred to as *Pitch* in the language of music. Essentially frequency is the property of the sound which is inferred by the human brain as *pitch*.
Note : The term *note* refers to the pitch and duration of a musical sound i.e. basically a set of sinusoids with a fundamental frequency that lasts a finite time duration with intensity decreasing with time.

Timbre : The combination of qualities of a sound that distinguishes it from other sounds of the same pitch and volume such as the distribution of energy across the harmonics.
Tempo : It refers to the rate of the pulse or the number of beats per minute(BPM). It is the reciprocal of Beat Period (time duration between two sub-sequent beats).
Onset and Offset Times : It refers to the beginning and ending of basically any musical event i.e. a sudden change in signal's properties. It might be a sharp amplitude increase, a beginning of a new score or so on. One technique is to compute a kind of distance between adjacent column vectors of the spectrogram image. This results in a novelty function, the **spectral flux** that shows changes in signal's frequency distribution. The peaks of such a novelty function indicates onset of a musical event.

0.4 Fourier Analysis in Music

Fourier analysis can be used to find the fundamental frequency of a played note at short time scales, whereas it can give information about the *timbre, rhythm and melody* of the sound at longer time scales. For example let us consider a note played on the keyboard and analyse it in the frequency domain.

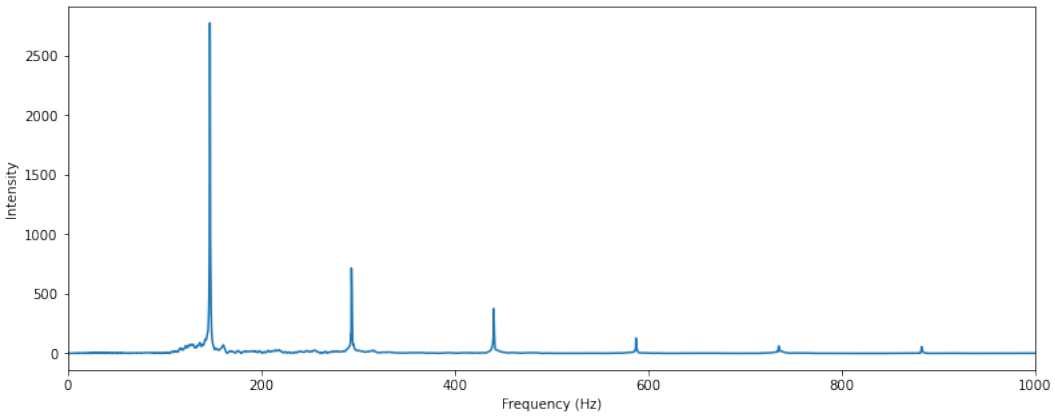

Figure 1: Fourier Transform

As we can see from the frequency domain plot that the intensity peaks occur at harmonics of fundamental frequency of 128 Hz with decreasing intensity as frequency increases. This tells us that much of the signal's energy is concentrated around lower frequencies which is distinctive of the note's timbre.
The Fourier transform has its limitations when it comes to short time scales as it gives information over the entire time duration. This led to advent of another technique in signal processing called **Short Time Fourier Transform (STFT)**. We use a window function that is non-zero for a short duration of time and mutiply the original signal with it to give a windowed signal. The STFT generates a 2D representation of the original signal where the horizontal axis is time and vertical axis is frequency, the

Spectogram. The STFT introduces new parameters i.e. window length and frame rate(number of samples taken between two successive frames), setting it apart from the classical Fourier Transform.
Below is the spectogram image of the same key shown above.

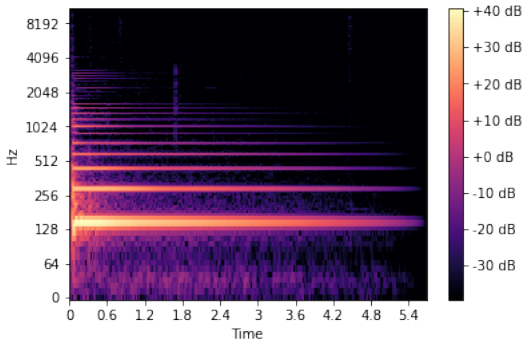

Figure 2: Spectogram Image

In the Spectogram image we can see that peak intensities occur at the multiples of fundamental frequency 128 Hz and the intensity decreases as for higher frequencies and with passage of time as the signal dies down. This reiterates the fact that majority of energy of the signal is concentrated in its lower frequencies in this example.

0.5 Software Tools

Over the past decade as Music Processing developed as a research field, the MIR community came up with many toolboxes that makes analysing music signals much more computationally accessible. Examples of such toolboxes include **MIRToolbox, madmom, Marsyas**. Prominent ones include the FMP notebooks that use the Jupyter notebook framework and the Python package *librosa* that have become a standard in the MIR industry and provide easy entry point for beginners. Here we explain how we can use Jupyter and librosa to analyse a song in the frequency domain, make inferences from the spectogram image, and gain insights into beats through novelty function.

```
import librosa
import matplotlib.pyplot as plt
import librosa.display
import seaborn
import numpy, scipy, IPython.display as ipd
from scipy.fft import fft, ifft

import urllib
filename = 'drum.mp3'
x, sr = librosa.load(filename)
ipd.Audio(x, rate=sr)

X = fft(x)
```

```
X_mag = numpy.absolute(X)
f = numpy.linspace(0, sr, len(X_mag))

#Plot the spectrum:

plt.figure(figsize=(13, 5))
a = plt.plot(f, X_mag) # magnitude spectrum
plt.xlabel('Frequency (Hz)')
plt.ylabel('Intensity')
plt.xlim([0,1000])
```

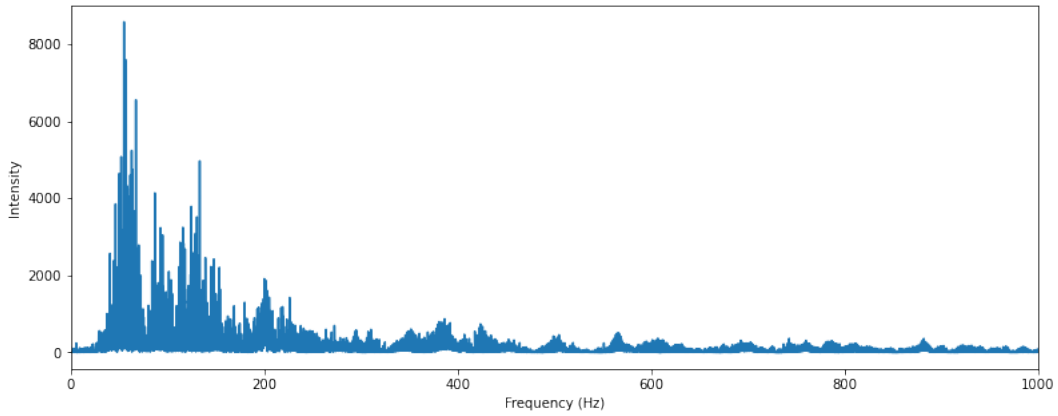

Figure 3: Fourier Transform

```
#librosa.stft computes a STFT.
    #We provide it a frame size, i.e. the size of the FFT, and a hop
   length,i.e. the frame increment:
hop_length = 512
n_fft = 2048
X = librosa.stft(x, n_fft=n_fft, hop_length=hop_length)

hop_length_sec = float(hop_length)/sr #hop length in sec
n_fft_sec = float(n_fft)/sr #frame size in seconds

print(hop_length_sec,n_fft_sec)
#output comes as - [0.023219954648526078 0.09287981859410431]

#Plotting Spectogram now
S = librosa.amplitude_to_db(abs(X))
#Let's plot a magnitude spectrogram where the colorbar is a linear
   function of the spectrogram values, i.e. just plot the raw values.
Xmag = abs(X)
librosa.display.specshow(Xmag, sr=sr, x_axis='time', y_axis='log')
plt.colorbar()
```

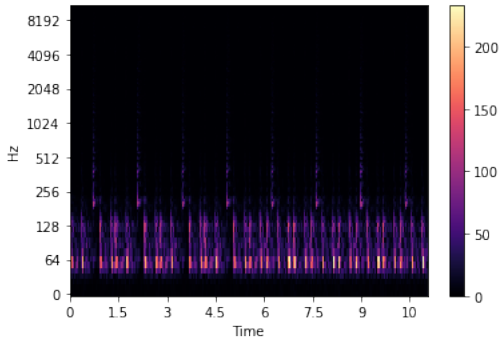

Figure 4: Spectogram

```
    #Now let's plot a magnitude spectrogram where the colorbar is a
    logarithmic function of the spectrogram values. We are mainly
    familiar with the log scale while calculating intensity.
Xdb = librosa.amplitude_to_db(Xmag)
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='log')
plt.colorbar(format='%+2.0f dB')
```

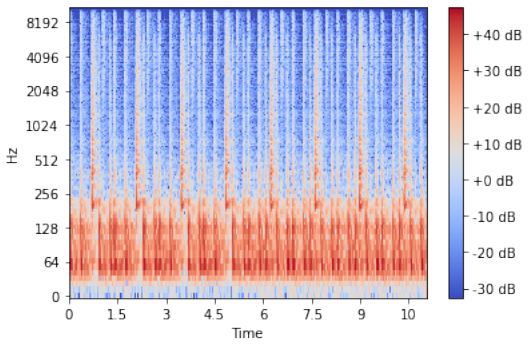

Figure 5: Spectogram

```
#Now we will be computing the novelty function and tempo
#computing the novelty function
onset_env = librosa.onset.onset_strength(x, sr=sr, hop_length=
    hop_length, n_fft=2048)
#plotting the onset envelope
frames = range(len(onset_env))
t = librosa.frames_to_time(frames, sr=sr, hop_length=hop_length)
plt.plot(t, onset_env)
plt.xlim(0, t.max())
plt.ylim(0)
plt.xlabel('Time (sec)')
plt.title('Novelty Function')
```

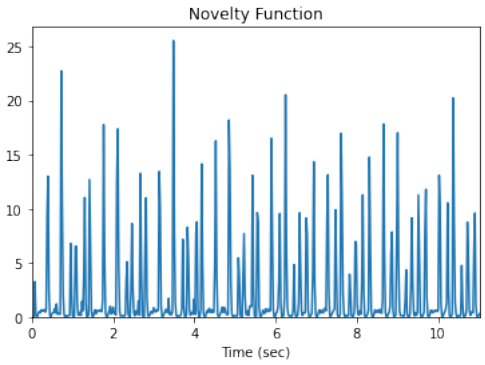

Figure 6: Novelty Function

```
tempo = librosa.beat.tempo(x, sr=sr)
print(tempo)

#Tempo comes as [172.265625]

T = len(x)/float(sr)
plt.figure(figsize=(20, 5))
seconds_per_beat = 60.0/tempo[0]
beat_times = numpy.arange(0, T, seconds_per_beat)
librosa.display.waveshow(x)
plt.vlines(beat_times, -1, 1, color='r')
```

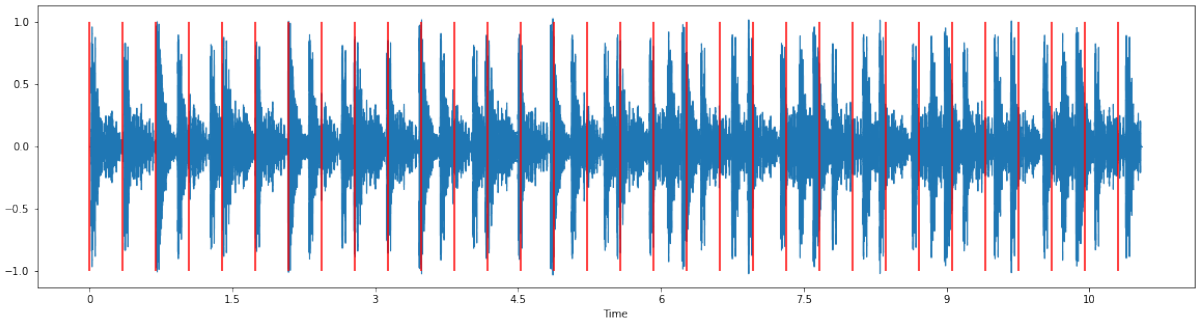

Figure 7: Tempo Estimation

0.6 References

1)[MMK21] This was the primary article which we thoroughly studied and created this summary
2)[Rat20] Used this Notebook to understand how to code using librosa to generate Fourier Transform and Spectogram.
3)[22] Used this webpage to understand how to code using librosa to generate Novelty Function and Tempogram

Bibliography

[Rat20] Ashwani Rathee. *Audio Signal Processing:Librosa*. 2020. URL: https://www.kaggle.com/code/ashkhagan/audio-signal-processing-librosa/notebook (visited on 07/14/2020).

[MMK21] Meinard Mueller, Brian McFee, and Katherine Kinnaird. "Interactive Learning of Signal Processing Through Music: Making Fourier Analysis Concrete for Students". In: *IEEE Signal Processing Magazine* 38.3 (2021), pp. 73–84. DOI: 10.1109/MSP.2021.3052181.

[22] *tempo estimation*. 2022. URL: https://musicinformationretrieval.com/tempo_estimation.html (visited on 10/27/2022).