ECEN 4532 - Lab 1

Andrew Teta

January 25, 2019

Introduction

In this discussion, we will be exploring some techniques recently developed in the audio industry to organize, search, and classify large music collections. We are developing some statistical methods, along with frequency analysis to automatically characterize songs and genres.

Format

Fundamentally, we begin with a '.wav' file. This is one variety of many signal encodings, generated by sampling a physical sound wave using a microphone. The microphone detects pressure differences within a medium (sound) and converts them into electrical voltages. As the voltages are converted into a digital representation, they are quantized at a sampling frequency (fs). Furthermore, the Nyquist sampling theorem dictates that for a signal to be represented accurately in quantized form, it must be sampled at twice the rate of the highest frequency component. For example, a dolphin can only hear sound in the range $7kHz - 120kHz$, so if we want to record the sound that a dolphin would be able to hear, we have to sample at a rate $fs = 240kHz$ or higher.

In this lab, we consider a small collection of music, organized by genre, in '.wav' format. Each song is sampled at $fs = 11,025Hz$. Narrowing our data set even further, we will extract a 24 second sample from the middle of each song. We will then implement a short-time Fourier transform (STFT), which divides the sample into $N = 512$ samples, or $46ms$ and call each 46 ms interval a 'frame'. The STFT is a good way to obtain frequency spectrum data, while maintaining a level of time-domain relevance. For the purposes of this lab, the frames will be non-overlapping.

1 Time-domain Analysis

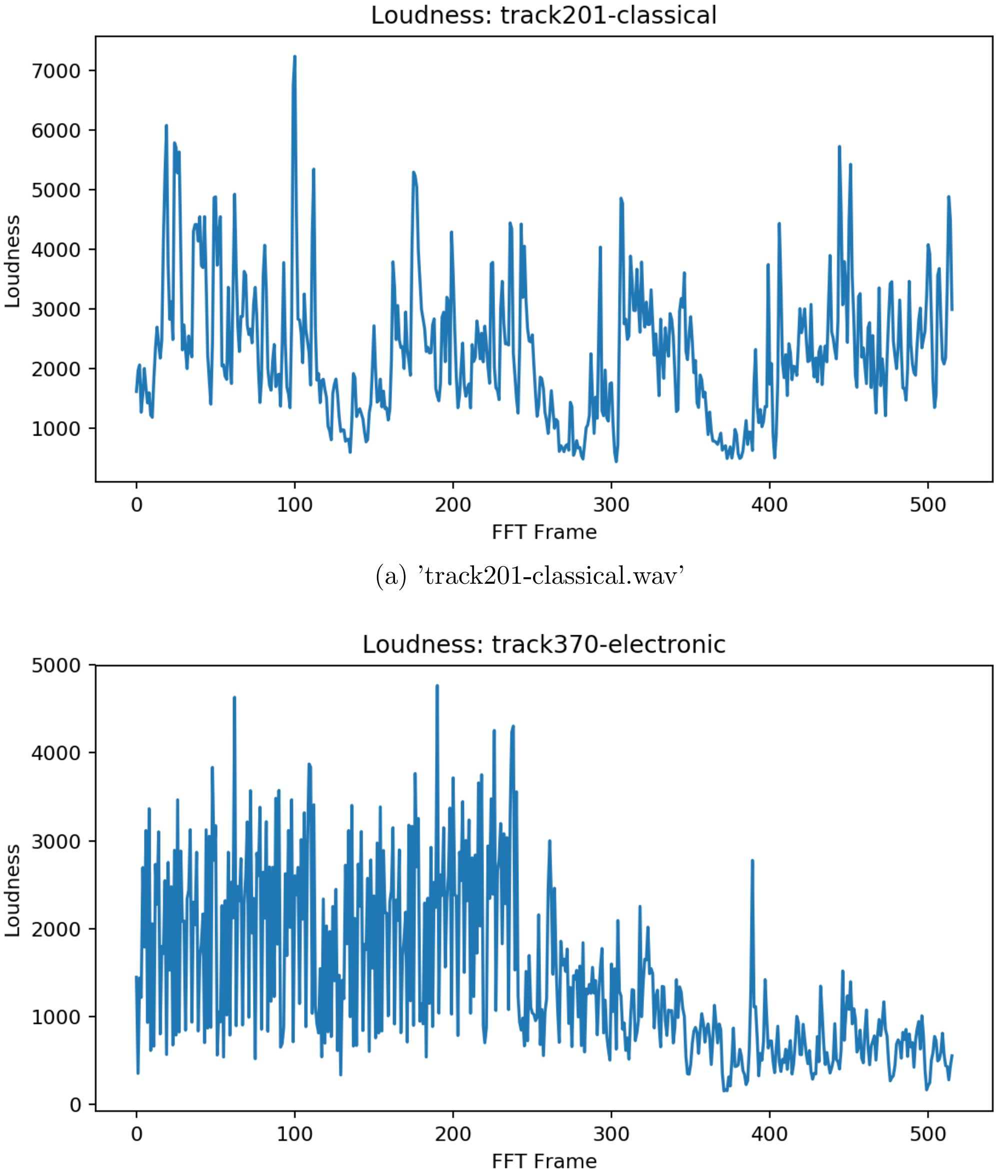
We begin by extracting a $24s$ sample from the middle of each song using the python function *scipy.io.wavfile.read*

1.1 Loudness

To get a sense of 'loudness', we will compute the standard deviation over a 'frame' of size $N = 512$ defined as:

$$\sigma(n) = \sqrt{\frac{1}{N}\sum_{m=0}^{N-1}[x(nN+m) - E[x_n]]^2} \qquad \text{with} \qquad E[x_n] = \frac{1}{N}\sum_{m=0}^{N-1} x(nN+m) \tag{1}$$

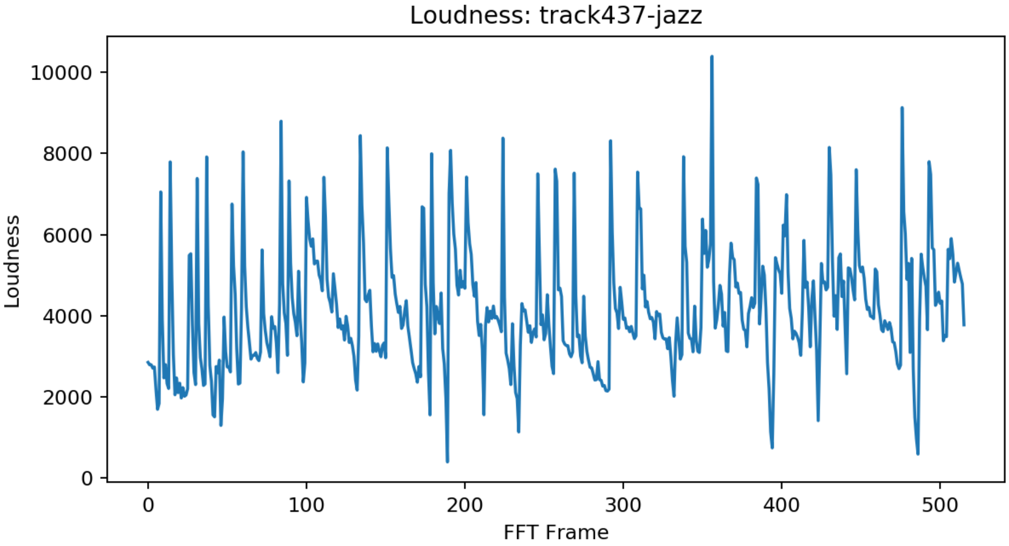
Results The output of the loudness calculation for one song of each genre is shown in the figures below.



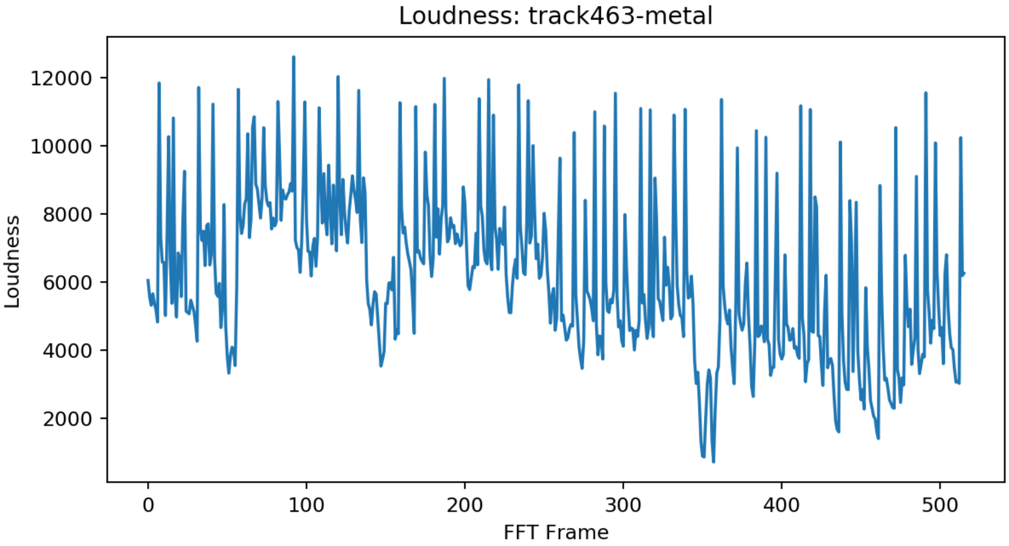
(a) 'track201-classical.wav'

(b) 'track370-electronic.wav'

Figure 1: Loudness vs. frame

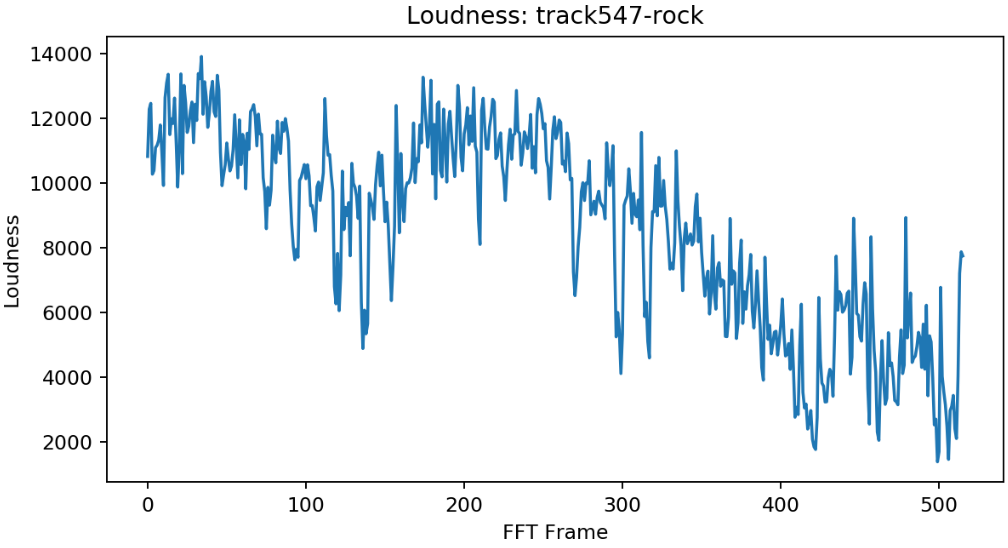


(c) 'track437-jazz.wav'

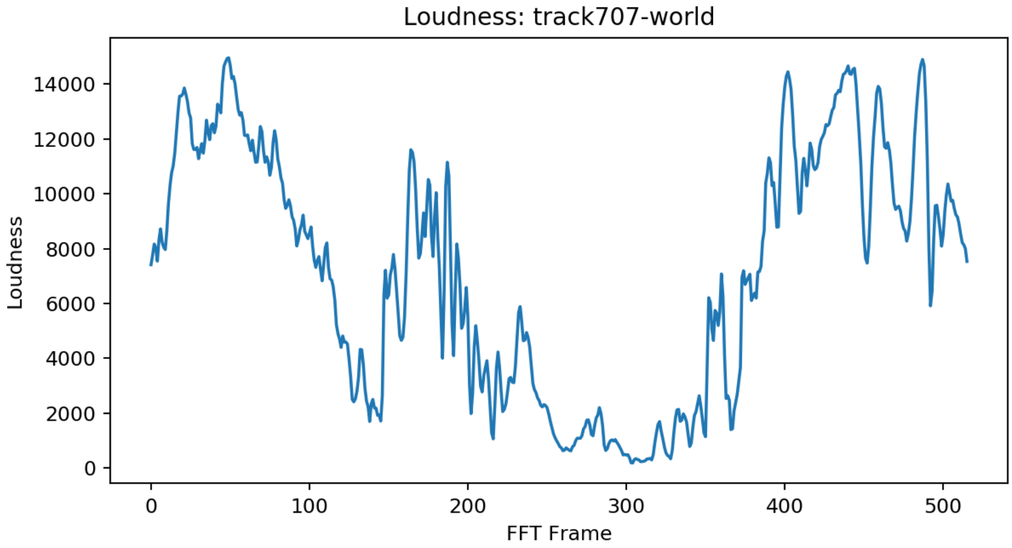


(d) 'track463-metal.wav'

Figure 1: Loudness vs. frame



(e) 'track547-rock.wav'



(f) 'track707-world.wav'

Figure 1: Loudness vs. frame

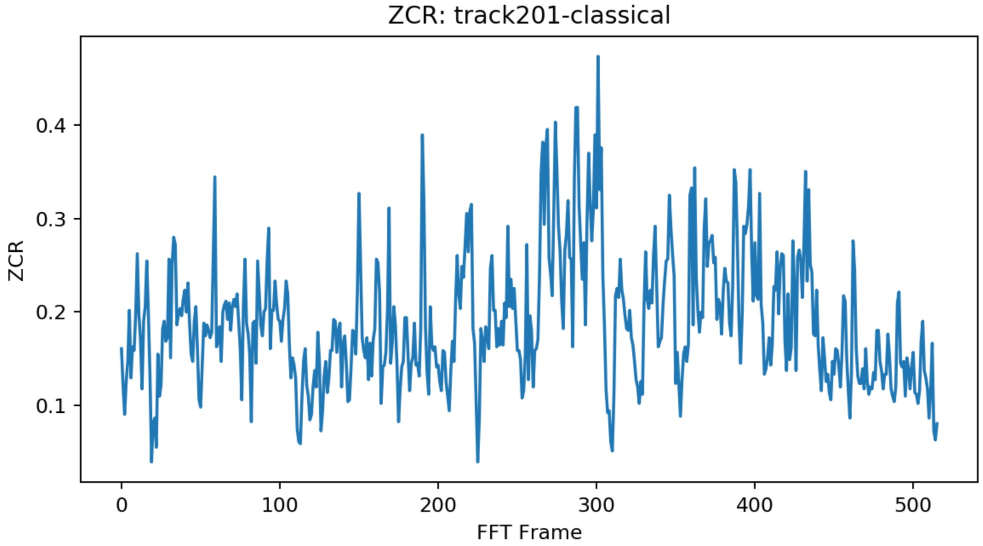
Comments While each of the plots in figure 1 has its own characteristics, loudness is not the best tool for characterizing genres. Figure 1b shows the uniformity of the track over time and contrasts with figure 1f as well as figure 1a, however it would be hard to distinguish between figure 1b and figure 1d. However, these plots do give a good sense of the progression of each song during the sample considered.

1.2 Zero Crossing Rate

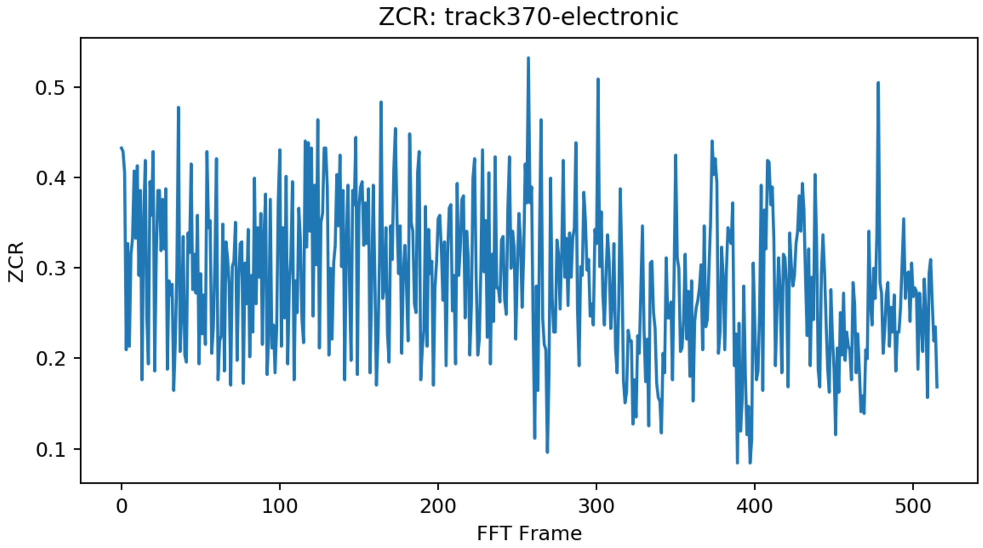
The zero crossing rate (ZCR) is the average number of times the audio signal crosses the zero amplitude line per unit time. It is related to pitch height and correlated to the noise in the signal. For this lab, ZCR is defined as:

$$ZCR(n) = \frac{1}{N-1}\sum_{m=1}^{N-1}\frac{1}{2}|sgn(x(nN+m)) - sgn(x(nN+m-1))|. \tag{2}$$

Results Again, we display the output of computed ZCR for one track of each genre in the figures below.

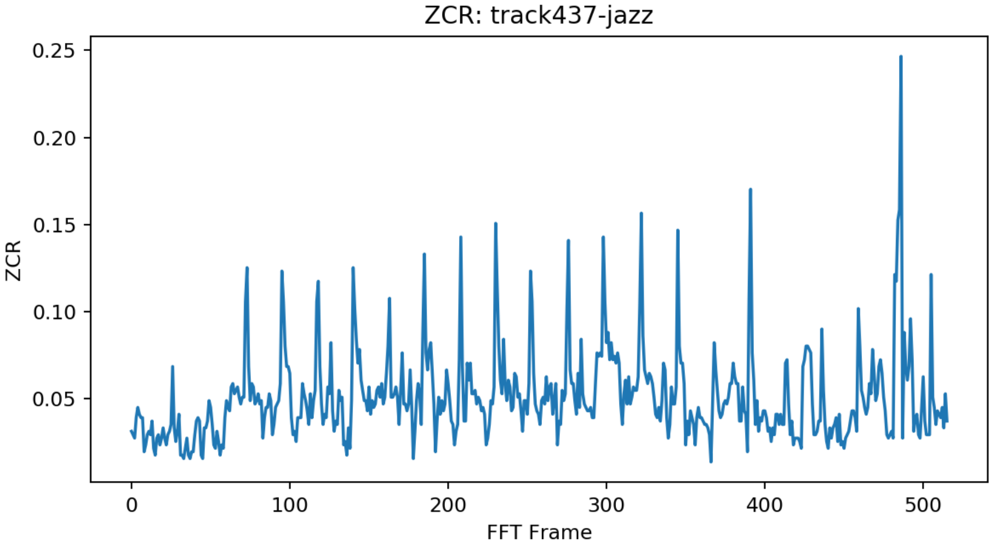


(a) 'track201-classical.wav'

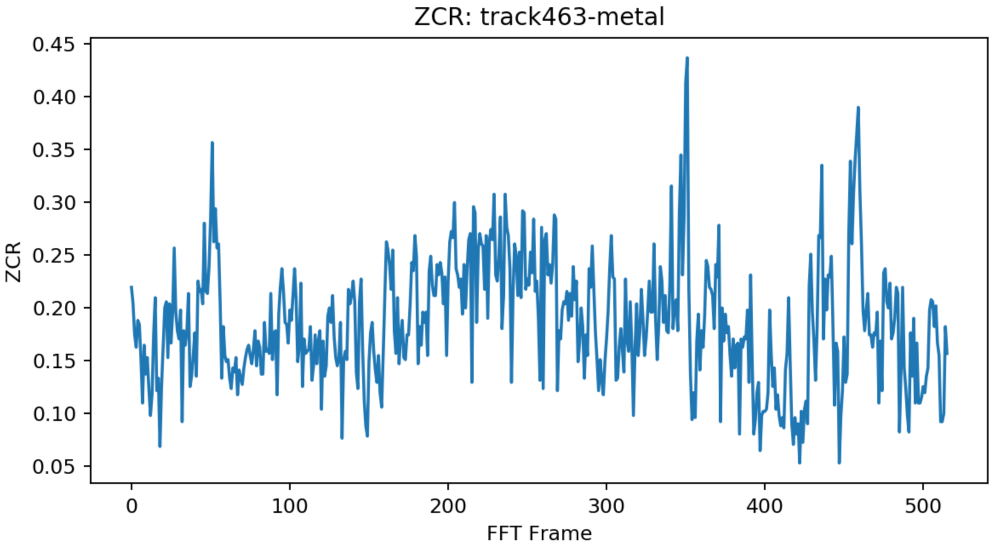


(b) 'track370-electronic.wav'

Figure 2: Loudness vs. frame

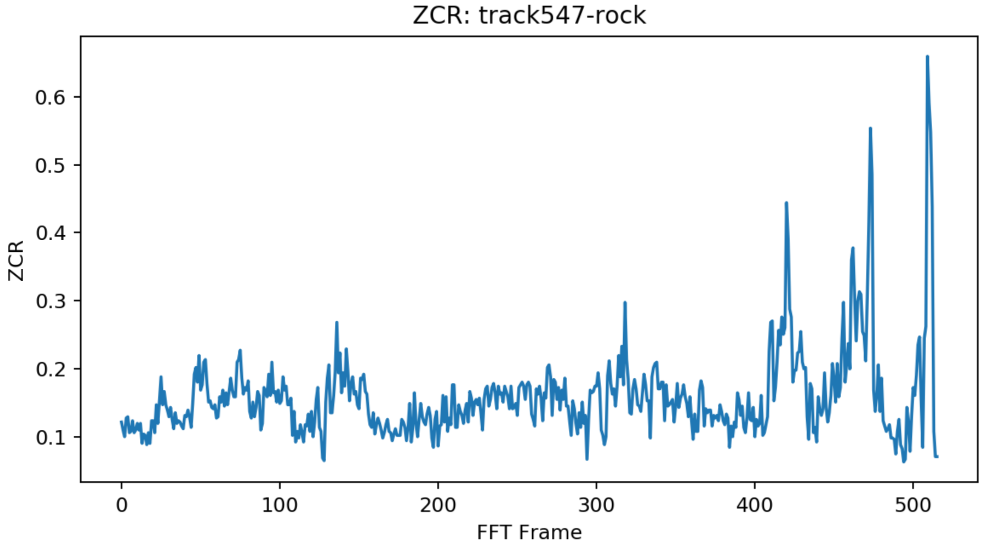


(c) ’track437-jazz.wav’

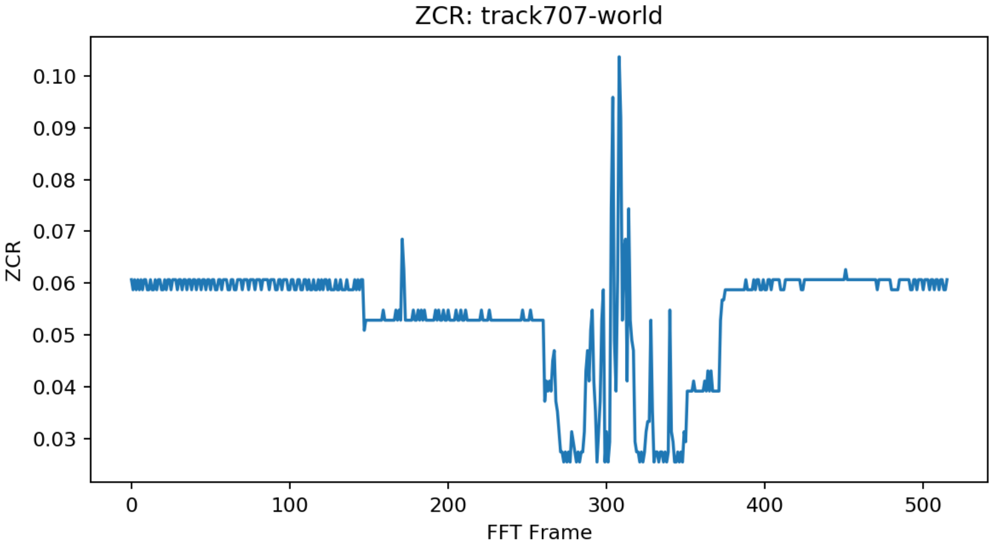


(d) ’track463-metal.wav’

Figure 2: Loudness vs. frame



(e) 'track547-rock.wav'



(f) 'track707-world.wav'

Figure 2: Loudness vs. frame

Comments From the plots in figure 2 we can definitely see some variability between genres. For example, the world track in 2f is easily distinguishable, however most of the other tracks are similar enough that this is also not likely a good measurement for classifying genre. This can be supported by the similarity between jazz (2c) and rock (2e).

2 Spectral Analysis

Sound, by nature is made up of many different frequencies or notes of different pitch. Thus it is natural to analyze audio in the spectral (frequency) domain.

Method As discussed in the introduction, we will accomplish our spectral analysis using a STFT, which decomposes short-time frames of a signal into discrete frequency bins, maintaining some time-domain relevance within the spectral domain. Then, we will perform some statistical analysis on the signal in an attempt to characterize some songs.

Rather than directly splitting each of our $N = 512$ frames, we will convolve (a fancy term for multiplication in the frequency domain) a taper window, w, with the signal and take the Fourier transform (FT). Since the signal is real, and we are concerned only with the magnitude, we will compute the FT of only half of the frequency spectrum to arrive at $N/2 = 256$ frequency bins for each of our $N = 512$ frames. Finally, for the purposes of this analysis, we are interested in the power spectrum of the original song, so we take the magnitude squared (i.e. $|X_n(k)|^2$, as a function of the frame index n and frequency index k).

To give some motivation for the taper window, w, we will derive the theoretical discrete-time Fourier transform (DTFT) for a given signal $x[n]$ where,

$$\begin{aligned} x[n] &= 1, && -N/2 \le n \le N/2 \\ x[n] &= 0, && \text{else} \end{aligned}$$

Recall the definition of the DTFT.

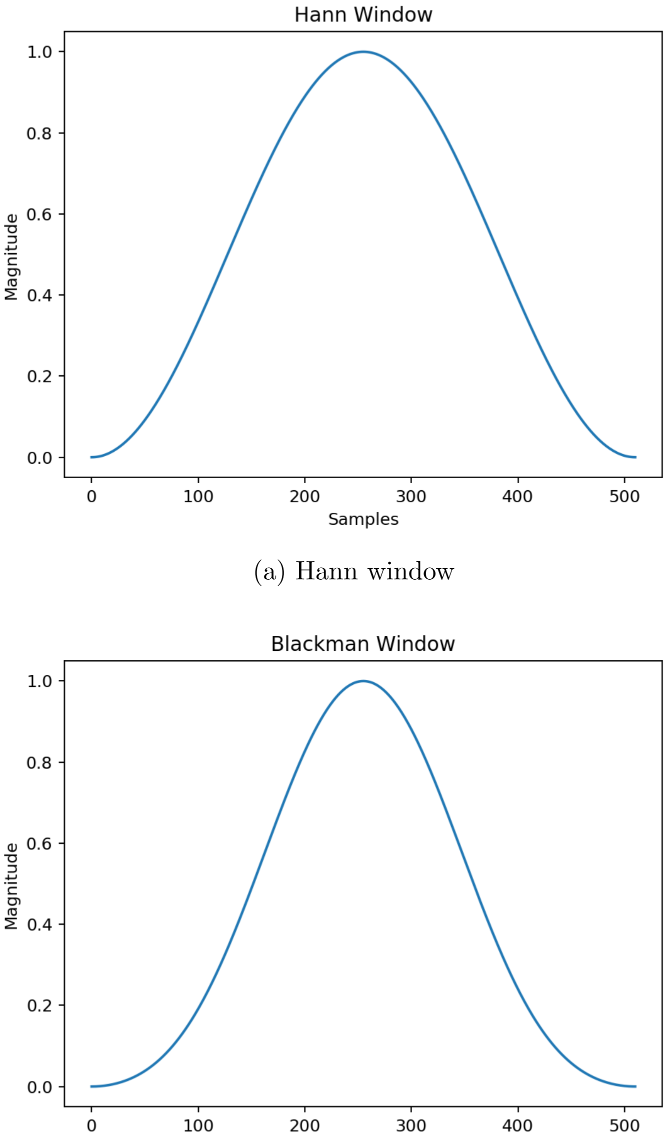
$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}. \tag{3}$$

Since $x[n] \neq 0$ over only a subset of values in the DTFT range, equation (3) becomes,

$$\begin{aligned} X(\omega) &= \sum_{n=-N/2}^{N/2} x[n]e^{-j\omega n} \\ &= \sum_{n=-N/2}^{N/2} e^{-j\omega n} \end{aligned}$$

2.1 Spectrogram

As promised, we will be implementing a STFT. In Python, we will use the *scipy.signal.spectrogram* function to automate this. However, we first need a window. For this, we will use the library function, *scipy.signal.get_window* which allows us to quickly generate taper windows of the same size as our STFT frames. There are many choices of windows, but we will look at the 'Hann' and 'Blackman' windows. See figure 3.

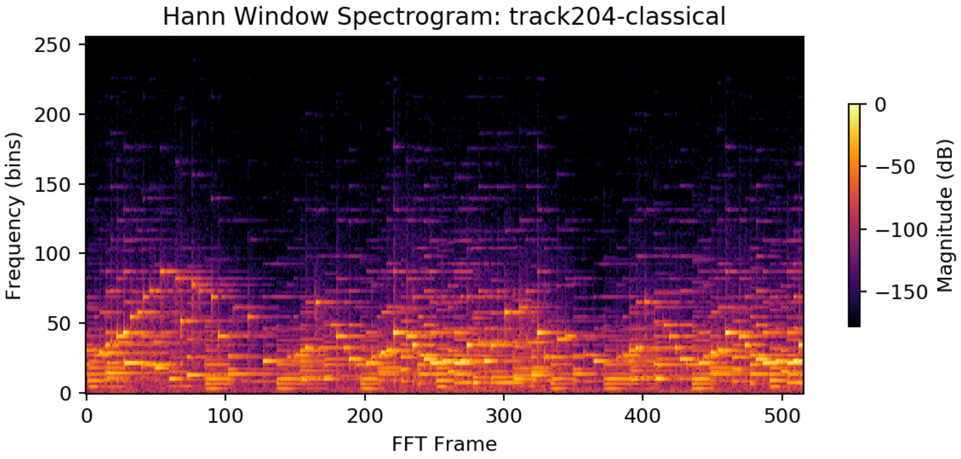


(a) Hann window

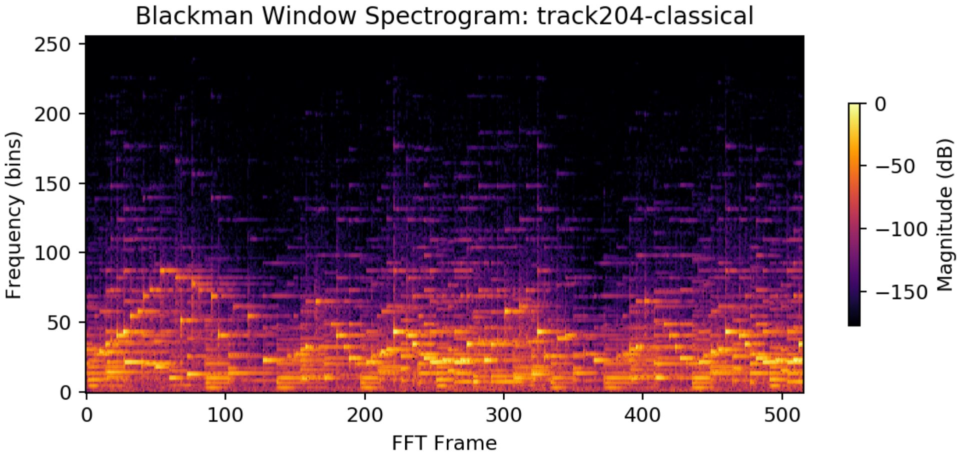
(b) Blackman window

Figure 3: Windows

As you can see, the Hann window (3a) tapers a little less steeply than the Blackman window (3b).

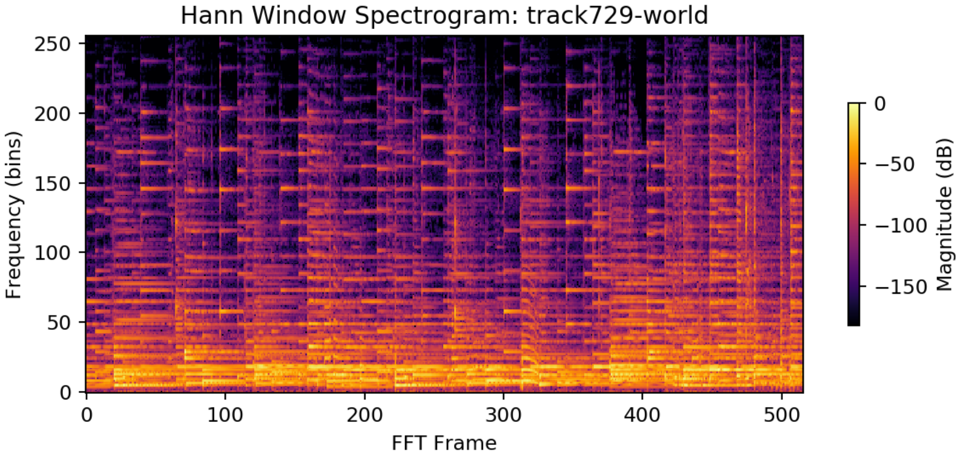


(a) Spectrogram of 'track204-classical' using a Hann window

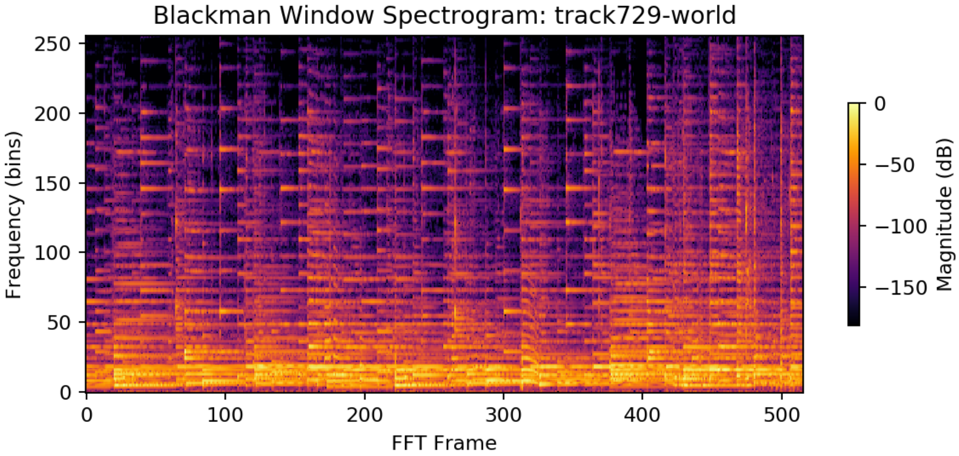


(b) Spectrogram of 'track204-classical' using a Blackman window

Figure 4: Hann vs. Blackman windows for a classical music track



(a) Spectrogram of 'track729-world' using a Hann window



(b) Spectrogram of 'track729-world' using a Blackman window

Figure 5: Hann vs. Blackman windows for a classical music track

Comments As far as I can tell, the two spectrograms produced by these two windows in figures 4 and 5 are identical. We will proceed with analysis using only the Blackman window.

2.2 Spectral Centroid and Spread

We wish to find the 'center of mass' of each frame in a spectrogram and will call this the spectral centroid. The centroid can be used to quantify sound sharpness or brightness. Additionally, we would like to find the spectral spread, or width of the spectrum around the centroid. Thus, we can then compare tone-like and noise-like sounds. For these concepts, we will treat the normalized magnitude of a spectral coefficient as if it were a 'probability' of that particular frequency. Then, for frame n, we have the 'probability' of frequency k

$$P_n(k) = \frac{|X_n(k)|}{\sum_{l=0}^{K} |X_n(l)|} \tag{4}$$

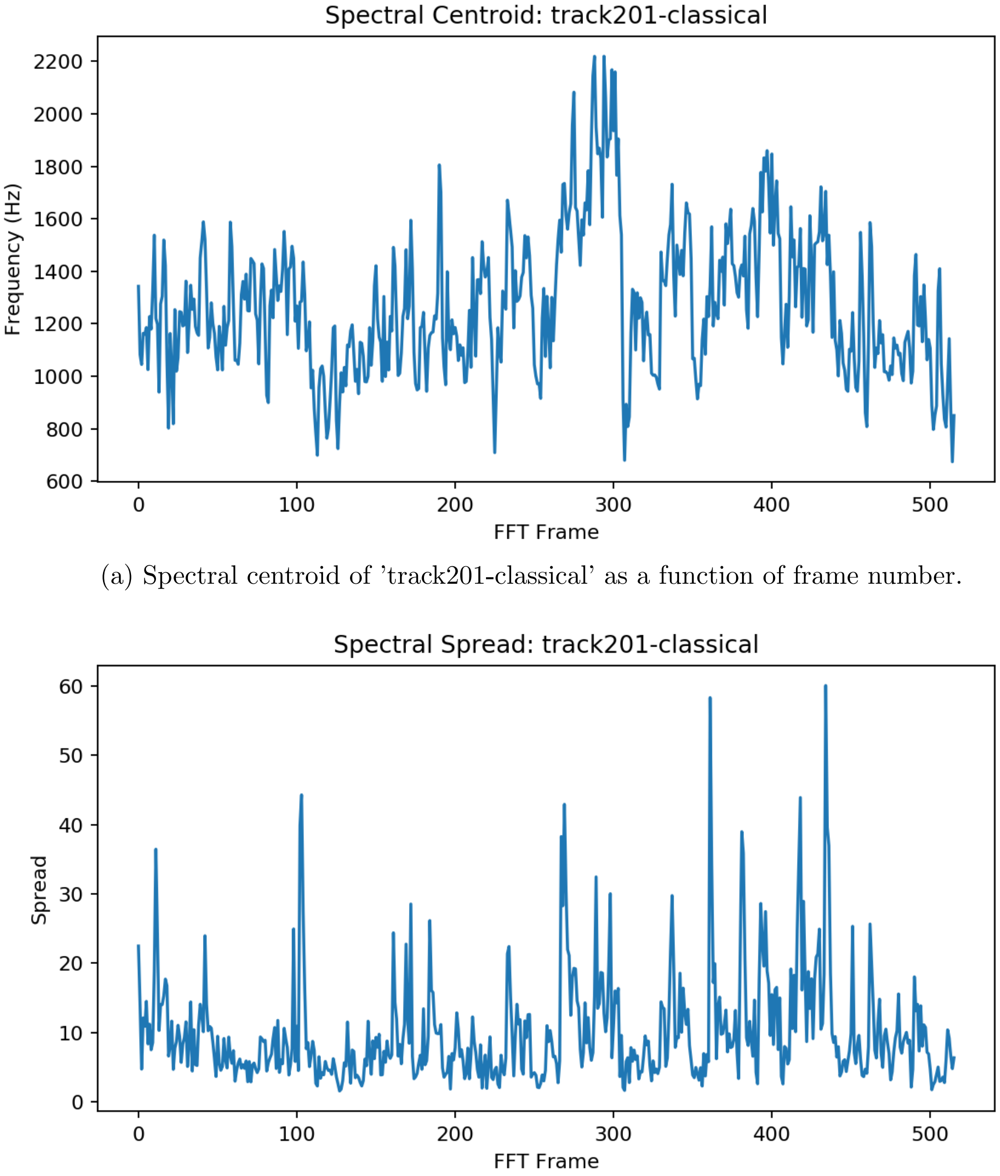
Which we can then use to define the spectral centroid as

$$\mu_n = \sum_{k=0}^{K} k P_n(k) \tag{5}$$

And the spectral spread for frame n is the standard deviation given by

$$\sigma_n = \sqrt{\sum_{k=0}^{K} [k - \mu_n]^2 P_n(k)} \tag{6}$$

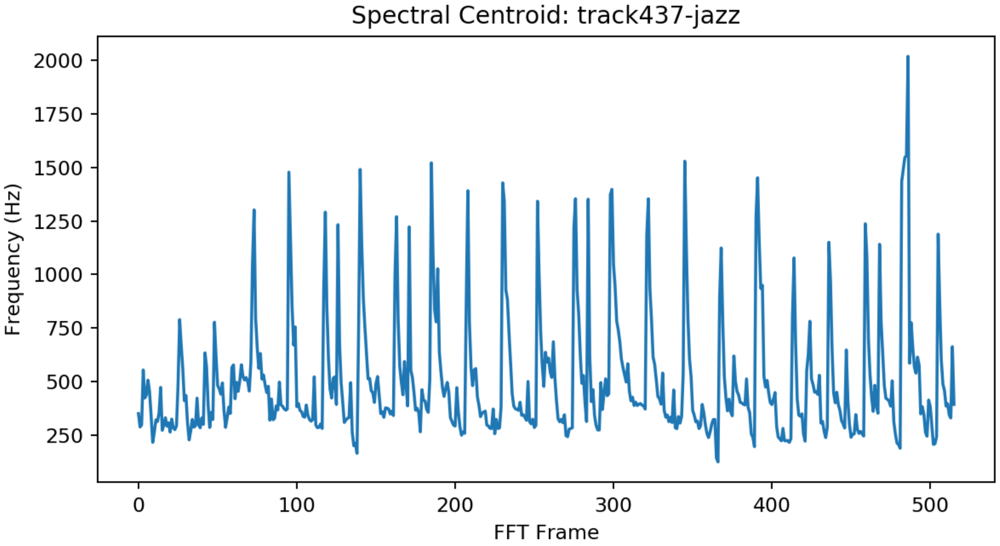
The spectral centroid and spread were calculated for a classical, jazz, and metal song and a time-domain plot is shown in figure



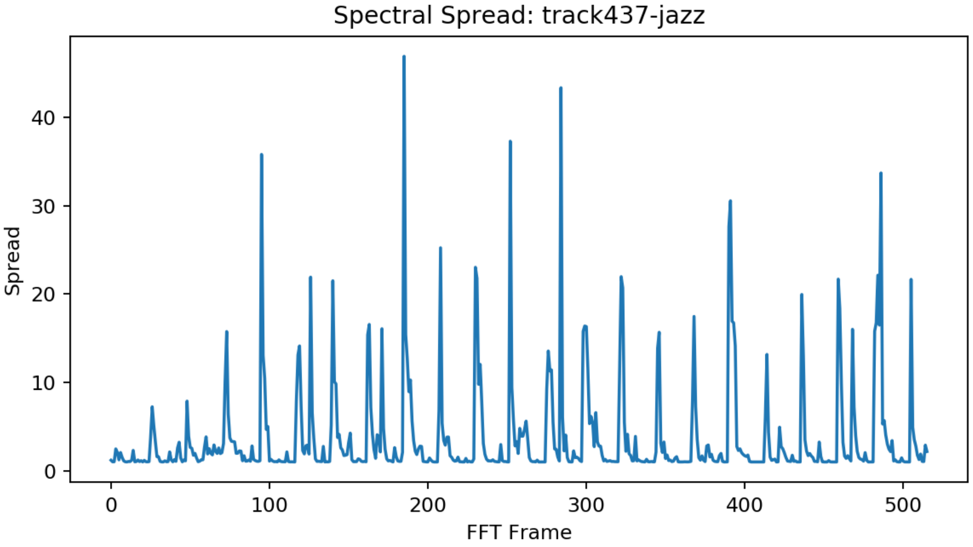
(a) Spectral centroid of 'track201-classical' as a function of frame number.

(b) Spectral spread of 'track201-classical' as a function of frame number.

Figure 6: Centroid and spread

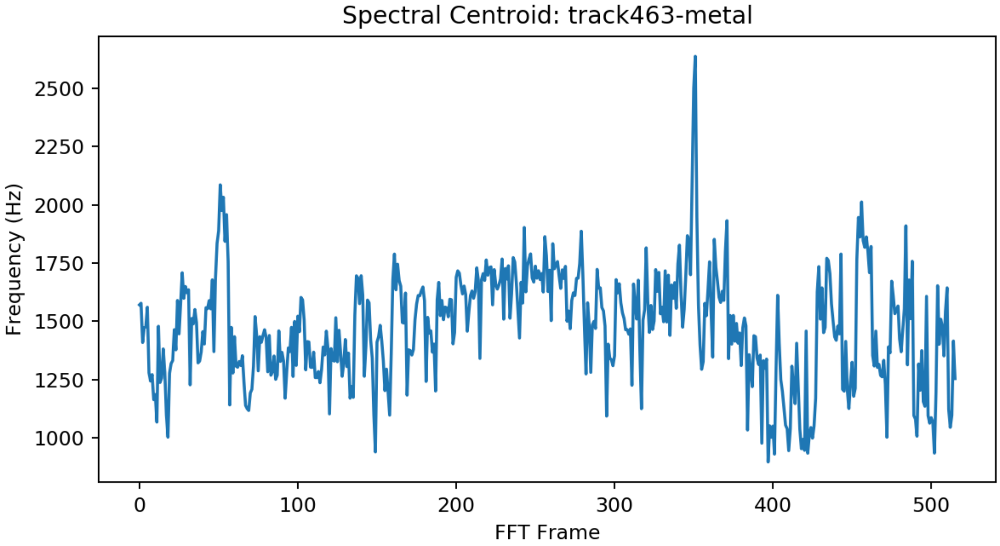


(c) Spectral centroid of 'track437-jazz' as a function of frame number.

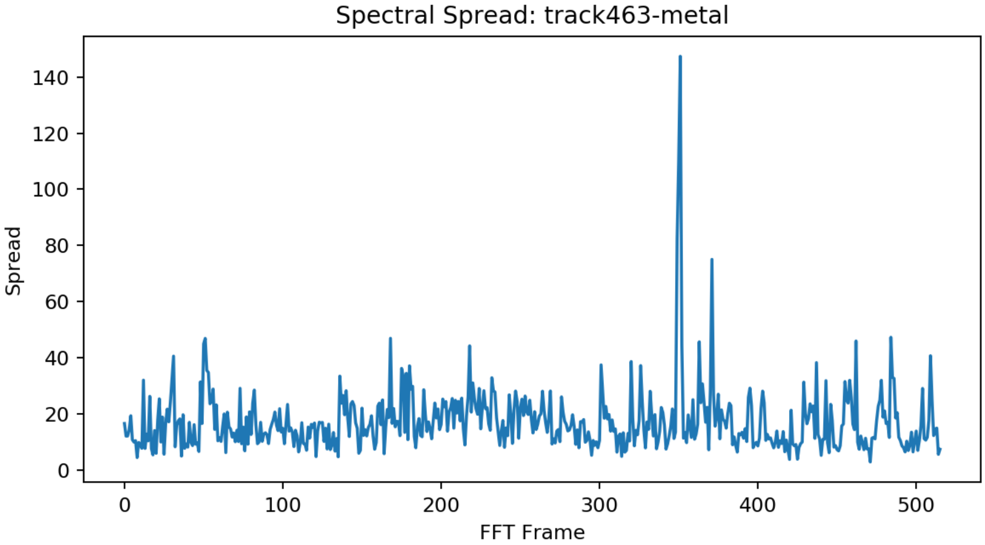


(d) Spectral spread of 'track437-jazz' as a function of frame number.

Figure 6: Centroid and spread



(e) Spectral centroid of 'track463-metal' as a function of frame number.



(f) Spectral spread of 'track463-metal' as a function of frame number.

Figure 6: Centroid and spread