Advanced DSP Final Report: General Audio Feature Extraction

B10502113
TAN XIAO XI 陳曉熙
b10502113@ntu.edu.tw
June 3, 2024

Abstract

This report delves into the process of general audio feature extraction, a crucial step in various audio analysis applications. It begins with preliminary steps for feature extraction including analog to digital conversion [1] and framing [2], setting the stage for further analysis. The report further explores into the extraction of features from both time and frequency domains [3]. In the time domain, it introduces features like amplitude envelope, RMS energy, and zero crossing rate. For frequency domain analysis, it discusses the use of Discrete Fourier Transform (DFT) and Short Time Fourier Transform (STFT), which help in obtaining spectrum and spectrograms. The application of the Hann window in STFT is also explained in detail. The report continues to introduce the extraction of frequency domain features such as band energy ratio and spectral centroid. Lastly, attention is given to the Mel scale for its perceptual relevance in frequency representation, leading to the creation of Mel spectrograms and the extraction of Mel Frequency Cepstral Coefficients (MFCCs) [4][5].

1 Preliminary work for feature extraction

1.1 Analog to digital conversion

Sound is a mechanical wave that is analog in nature. Since analog signal has continuous values for both time and amplitude, storing it exactly as it is would require infinite memory, which is not possible. To use digital technology to work with sound, we first need to convert these analog signals into digital form. This process is called analog to digital conversion. It involves two main steps: sampling and quantization.

Sampling is the process of recording data points across sound waves at specific time intervals. A common sampling rate F_s used is 44,100 Hz. At a frequency above the Nyquist frequency $f_N = \frac{F_s}{2}$, aliasing may occur. Therefore, by using a sampling rate of 44,100 Hz, we ensure that aliasing doesn't happen for frequencies below 22,050 Hz, which is approximately the highest frequency most humans can hear.

After sampling, the analog amplitude values of sample points will be mapped to a limited range of discrete values, which is the process of quantization. The quantized amplitude value is represented in binary form. The resolution or bit

depth is the number of bits assigned to each amplitude value. With a typical resolution of 16 bits, the number of quantization levels will be $2^{16} = 65536$. Quantization is a lossy process, meaning some of the original sound's detail is lost. This loss shows up as quantization noise, which is the difference between the original analog signal and the quantized digital signal. However, the higher the resolution, the closer our digital values can match the original analog amplitudes, resulting in less quantization noise and a higher dynamic range. The signal to quantization noise ratio (SQNR) is a measure that correlates with the dynamic range, where $\text{SQNR} \approx 6.02 \times \text{bit depth}$. So, a higher resolution means a higher SQNR and a better sound quality.

1.2 Framing

Audio signals change their characteristics frequently. In a music recording, there are sections that are high-pitched and loud, while others are quiet and subtle. If we compute features for the entire recording at once, the loudest and most intense parts could overshadow the rest. To avoid this, we divide the recording into brief segments before extracting features. This process is known as framing. Each frame is a short slice of the audio, and we analyse these slices one by one. By doing this, we can capture the varying properties of the sound throughout the recording. Overlapping of frames is commonly used to ensure that no significant part of the audio signal is missed and to provide a smoother transition between frames in the analysis.

In framing, the frame size M refers to the number of samples in each frame. The duration of a frame generally ranges from 10ms to 50ms. This range is chosen because it is important for the audio within a frame to be long enough to be perceptible by human ears, making the analysis meaningful. The shortest duration typically perceptible to the human ear is about 10ms. The hop size is the number of samples by which we shift the frame to create the next frame. This shifting determines the degree of overlap between consecutive frames. The total number of frames in an audio signal can then be calculated by

$$\text{Number of frames} = \frac{N-M}{hop\ size} + 1\ ,$$

where N = total number of samples and M = frame size.

2 Time domain feature extraction

After converting the audio from analog to digital and segmenting it into frames, we can begin extracting features. Common features extracted from the time domain include the amplitude envelope, root mean square (RMS) energy, zero crossing rate and so on.

2.1 Amplitude envelope

The Amplitude Envelope of a signal is created by taking the maximum amplitude values among all the samples within each frame. The Amplitude Envelope reflects the changes in the loudness of a sound over time. However, this method is sensitive to sudden spikes in loudness. If there's a loud spike in a single frame, the Amplitude Envelope will only capture that spike for that particular frame which may cause inaccuracy.

It is most commonly used for onset detection and speech analysis. For instance, when examining the Amplitude Envelope of someone reciting a Chinese poem as shown in Figure 1, we can examine the flow of the poem and also observe that the vowels typically have a higher amplitude than the consonants.

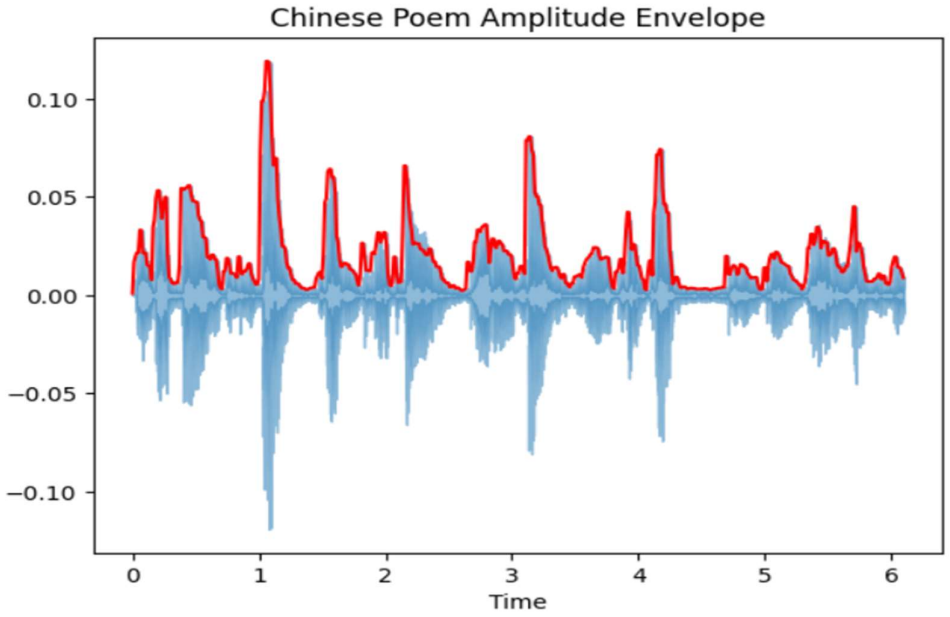

Figure 1. Chinese Poem Amplitude Envelope

2.2 RMS energy

The root mean square energy of all samples in a frame can be calculated by

$$RMS_t = \sqrt{\frac{1}{M} \cdot \sum_{k=t \cdot M}^{(t+1) \cdot M-1} |s(k)|^2} \,,$$

where M = frame size, $|s(k)|^2$ = energy of the k^{th} sample and t indicates the t^{th} frame we are calculating. Like the amplitude envelope, Root Mean Square energy serves as an indicator of the loudness of a sound. However, RMS energy is less affected by sudden loud spikes, or outliers, compared to the amplitude envelope. Figure 2 shows the difference between the amplitude envelope and RMS energy in

the audio of a Chinese poem. The amplitude envelope shows many spikes, indicating its sensitivity to outliers. In contrast, the RMS energy curve is smoother, reflecting a more reliable assessment of the sound's loudness over the duration of the audio.

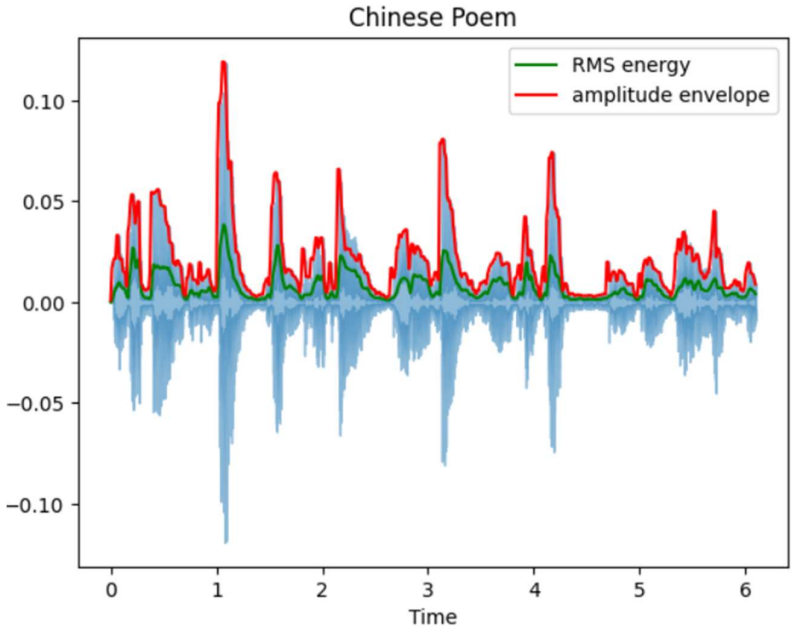

Figure 2. Difference between RMS energy and amplitude envelope

2.3 Zero crossing rate

The zero crossing rate is a measure of how frequently a signal crosses the horizontal axis, switching from positive to negative values and vice versa. The zero crossing rate for the t^{th} frame can be calculated by

$$ZCR_t = \frac{1}{2M} \cdot \sum_{k=t \cdot M}^{(t+1) \cdot M-1} |sgn\big(s(k)\big) - sgn(s(k+1))|,$$

where M = frame size and

$$sgn\big(s(k)\big) = \begin{cases} 1, & s(k) \geq 0, \\ -1, & s(k) < 0. \end{cases}$$

Zero crossing rate is particularly useful in voice activity detection as it helps indicate the noisiness of a signal. Generally, a noisier signal will have a higher zero crossing rate. Moreover, the zero crossing rate is useful for distinguishing between percussive and pitched sounds. For example, rock music typically exhibits a higher zero crossing rate compared to classical music due to its percussive nature. This feature is also used for basic pitch estimation in monophonic audio signals, where a higher zero crossing rate suggests a higher pitch.

3 Frequency domain feature extraction

Most of the key features we use to analyze audio signals are found in the frequency domain. In order to convert audio signal from time domain to frequency domain, we use tools like the Discrete Fourier Transform and the Short Time Fourier Transform. Some common features we look for in the frequency domain include band energy ratio, spectral centroid, bandwidth and MFCC.

3.1 Discrete Fourier Transform

By using Fourier Transform, an analog signal can be converted from time domain to frequency domain. However, once we've converted audio from analog to digital, we can't use the regular Fourier Transform anymore. For digital audio signals, we need to use Discrete Fourier Transform, a version of the continuous-time Fourier Transform adapted for digital signals.

Intuitively, to convert the continuous-time Fourier Transform

$$\hat{g}(f) = \int_{-\infty}^{\infty} g(t) \cdot e^{-i2\pi ft} dt,$$

to Discrete Fourier Transform, we can replace the continuous time variable (t), with the number of discrete samples (n), which then obtain:

$$\hat{x}(f) = \sum_{n=0}^{N-1} x(n) \cdot e^{-i2\pi fn},$$

where N = total number of samples.

However, we can only compute transform for a finite number of frequencies. To solve this problem, the number of frequencies for computation, or also known as number of frequency bins is chosen to be the same as the total number of discrete samples (N) in the audio signal. This approach is computationally efficient and also allows for invertible transformation, meaning we can switch back and forth between the time domain and the frequency domain easily. The final Discrete Fourier Transform will then become

$$\hat{x}(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-i2\pi \frac{k}{N}},$$

where k = [0, N-1]. The analog frequency for a given k value can be computed using the equation

$$f_k = \frac{kF_s}{N},$$

where F_s is the sampling rate.

In audio signal processing, we are particularly focusing on the magnitude of the Discrete Fourier Transform coefficients. By analyzing these magnitudes, we can construct what is known as a magnitude spectrum, which visually represents the different frequencies and how present they are in the audio. The magnitude spectrum has a central point of symmetry at the Nyquist frequency, which means that the right side of the spectrum is a mirrored image of the left side. Therefore, we usually only focus on the left side of the magnitude spectrum.

Moreover, to compute Discrete Fourier Transform, we usually use the Fast Fourier Transform algorithm. The standard Discrete Fourier Transform has a computational complexity of $O(n^2)$, on the other hand, the Fast Fourier Transform is much more efficient, with a complexity of $O(n \log_2 n)$.

3.2 Short-time Fourier Transform

By using Discrete Fourier Transform, we can create a visual representation of an audio signal called a magnitude spectrum. However, magnitude spectrum only provides information about the whole duration of the audio. In order to understand how the frequency components change over time, we need to perform Short Time Fourier Transform. The Short Time Fourier Transform is basically performing Discrete Fourier Transform on each frame after the audio signal undergoes framing. This way, we get a series of magnitude spectrums that tell us about the frequency components at different moments in the audio, giving us a detailed picture of how the frequencies vary with time.

A significant challenge when using the Short Time Fourier Transform is spectral leakage. This occurs when the endpoints of the framed signal are discontinuous, which will appear as high frequency components not present in the original signal after performing Fourier Transform. To prevent this, we apply a windowing function to each frame before performing the Discrete Fourier Transform by

$$x_w(k) = x(k) \cdot w(k),$$

where $w(k)$ is the windowing function. A popular choice is the Hann window,

$$w(k) = 0.5 \cdot \left(1 - \cos\left(\frac{2\pi k}{K-1}\right)\right), k = 1 \dots M$$

which is a bell-shaped curve that approaches zero at both ends. When we apply the Hann window to a frame, it effectively fades out the samples at the ends, creating a smoother, more periodic signal that reduces spectral leakage. The effect of applying Hann window can be seen in Figure 3 below. Since windowing can cause us to lose some signal information at the frame's ends, we use overlapping frames. This overlap ensures that any information lost at the end of one frame is captured

in the next, preserving the continuity of the audio signal throughout the Short Time Fourier Transform process.

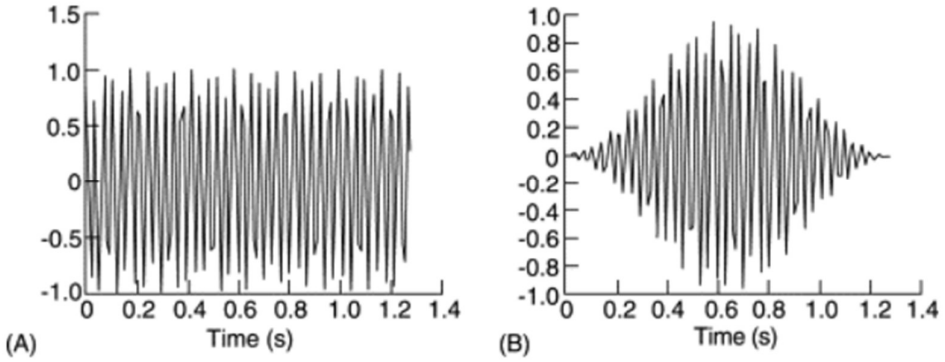

Figure 3. (A) Original signal (B) Signal after applying Hann window [6]

After the windowing process, we can then perform Short Time Fourier Transform. The equation of Short Time Fourier Transform will be

$$S(m,k) = \sum_{n=0}^{M-1} x(n+mH) \cdot w(n) \cdot e^{-i2\pi n\frac{k}{M}},$$

where M = frame size, H = hop size, $w(n)$ = windowing function and m indicates the m^{th} frame we are calculating. In section 3.1, it is stated that the number of frequency bins is chosen to be the same as the total number of discrete samples (N) in an audio signal for Discrete Fourier Transform. However, for Short Time Fourier Transform, the number of frequency bins for a frame will not equal to the frame size. Due to the central point of symmetry at the Nyquist frequency, the number of frequency bins in a frame can be reduced to

$$Number\ of\ frequency\ bins = \frac{M}{2} + 1,$$

where M = frame size.

When selecting the frame size for audio signal processing, it's important to consider the time-frequency trade-off. A larger frame size means more frequency bins and higher frequency resolution, which allows us to see more detail in the frequency domain. However, this also means we're looking at a larger segment of time, so our time resolution decreases, thus we see less detail in how the sound changes over time. On the other hand, a smaller frame size gives us better time resolution, as we're analyzing shorter segments of the audio. But this comes at the cost of decrease frequency resolution. The choice between time and frequency resolution depends on the specific needs of the application. For instance, in onset detection, where we're trying to identify the exact moments when sounds begin, time resolution is more critical.

After we apply the Short Time Fourier Transform to an audio signal, we get a matrix with complex numbers that has the size = (number of frequency bins, number of frames). By calculating the squared magnitude of each complex number in the matrix,

$$Y(m,k) = |S(m,k)|^2,$$

we obtain a new matrix of the same shape, but with all real numbers. This new matrix can be visualized as a spectrogram. Moreover, the decibel scale is used instead of amplitude since we perceive loudness logarithmically. Figure 4 presents a spectrogram of an audio recording where a child is singing the C major scale. It not only shows the frequency components, it also clearly shows the frequencies change as the child moves through the different notes of the scale over time, thus provides a time-frequency representation of the audio signal. The spectrogram is widely used in fields like AI music and music information retrieval.

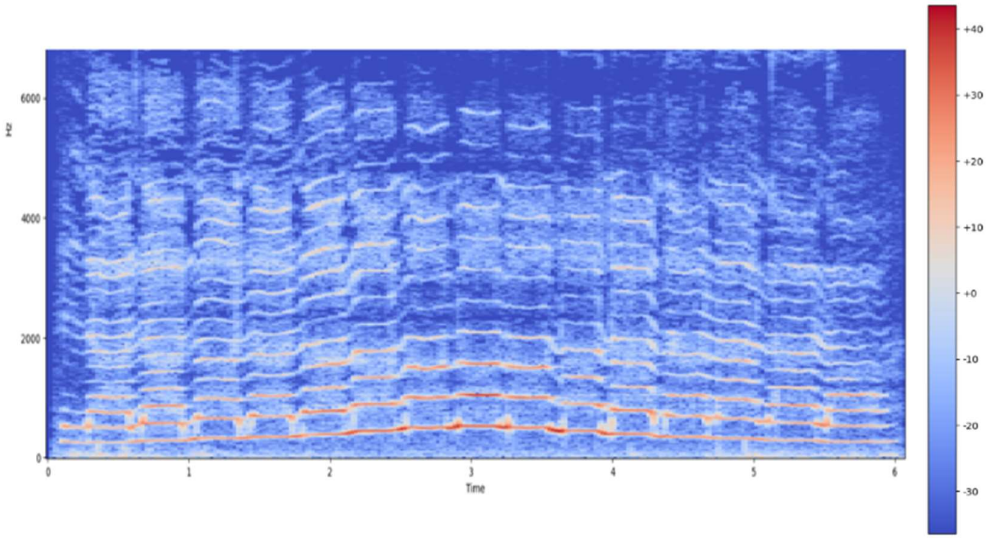

Figure 4. Spectrogram of C major scale

3.3 Band energy ratio

From the spectrogram we've obtained, we can calculate the band energy ratio. This ratio compares the energy present in the lower frequency bands to that in the higher frequency bands. The band energy ratio for the t^{th} frame can be calculated by

$$BER_t = \frac{\sum_{n=1}^{F-1} m_t(n)^2}{\sum_{n=F}^{K} m_t(n)^2},$$

where K = number of frequency bins, $m_t(n)^2$ = power at frequency bin n and

frame t and F = split frequency that separates the lower and higher frequencies. The split frequency is a threshold value and is often set at 2000Hz, although it can vary based on specific conditions or requirements. The denominator sums the power in the higher frequency bands, while the numerator sums the power in the lower frequency bands. By comparing these sums, the band energy ratio gives us an indication of whether low or high frequencies are more dominant in a particular frame of the audio signal.

The band energy ratio is a useful feature for classifying music genres, as different genres often have different energy distributions across frequency bands. It's also used to distinguish between music and speech, as these typically have different patterns of energy across the frequency spectrum.

3.4 Spectral centroid

From the spectrogram, we can also calculate the spectral centroid. The spectral centroid is like the center of gravity for a sound's spectrum. It shows us where most of the sound's energy is concentrated across different frequency bands. This measure is closely related to the "brightness" of the sound, which is a key aspect of its timbre. Sounds with higher spectral centroid values tend to be perceived as brighter. This makes it a useful feature for tasks like classifying music genres or identifying the mood of a piece of music.

It can be thought of as a weighted average of the frequencies present in the sound. Spectral centroid for the t^{th} frame can be calculated by

$$SC_t = \frac{\sum_{n=1}^{K} m_t(n) \cdot n}{\sum_{n=1}^{K} m_t(n)},$$

where K = number of frequency bins and $m_t(n)$ = weight for frequency bin n.

3.5 Mel spectrograms

Humans perceive frequency logarithmically. For example, while the frequency difference between 200 Hz and 300 Hz is the same as between 1200 Hz and 1300 Hz (100 Hz), the lower frequency pair sounds more further apart to our ears. The Mel scale is designed to match this logarithmic perception of pitch. On the Mel scale, a difference in Mel units represents a perceived change in pitch that is consistent across the frequency spectrum. For instance, a pitch difference perceived between 500 Mel and 510 Mel is similar to the difference between 1000 Mel and 1010 Mel. Therefore, we can create a Mel spectrogram, which is a spectrogram where the frequencies are converted to the Mel scale.

To convert frequencies to Mel scale, we need to construct Mel filter banks. First, we convert the lowest and highest frequencies of our range into Mel using the

equation

$$m = 2595 \cdot \log(1 + \frac{f}{700}) .$$

We then partition this Mel scale with equally spaced points, based on the number of Mel bands we want, typically 40. These points are then converted back to Hertz by

$$f = 700(10^{\frac{m}{2595}} - 1)$$

and we round them to the nearest frequency bin which will be the center frequencies for our filters. Each filter in the Mel filter bank is triangular, peaking at the center frequency with a value of 1 and decrease linearly to 0 at the center frequencies of the adjacent filters. This creates a series of overlapping triangular filters shown in Figure 5, which is the Mel filter banks.

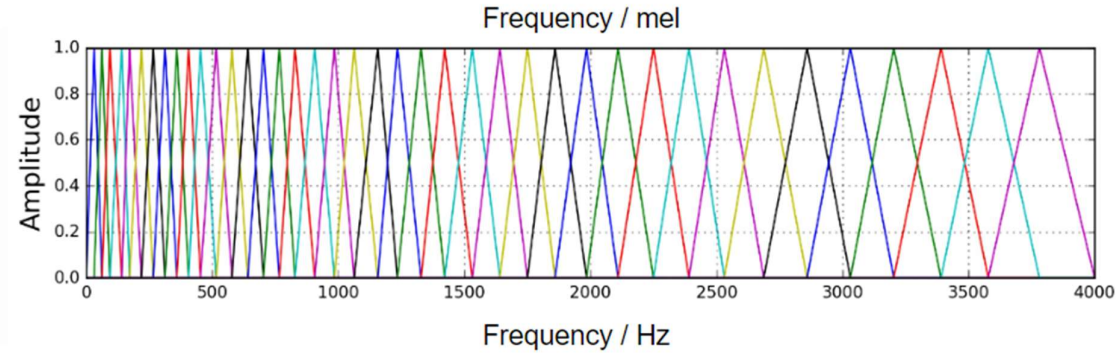

Figure 5. Mel filter banks with 40 Mel bands [7]

Once we have the Mel filter banks, we apply them to the spectrogram to produce Mel spectrograms. Note that the amplitude of the spectrogram here is already converted to decibels. The Mel spectrogram is obtained by multiplying the matrix of the Mel filter banks (M), with the matrix of the spectrogram (Y). The result is the Mel spectrogram matrix. Figure 6 displays the Mel spectrogram of an audio recording where a child is singing the C major scale. This Mel spectrogram is derived from the original spectrogram shown in Figure 4.

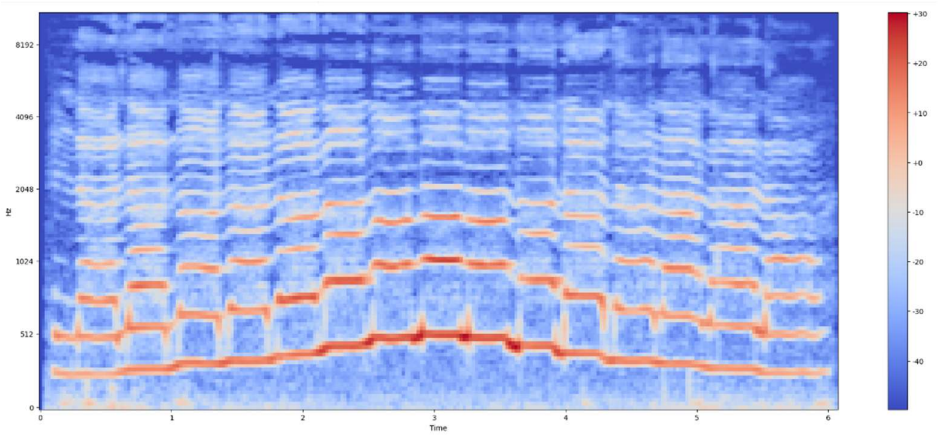

Figure 6. Mel spectrogram of C major scale

3.6 Mel Frequency Cepstral Coefficients

To extract the Mel Frequency Cepstral Coefficients (MFCCs), we perform Discrete Cosine Transform on the Mel-scaled Log-amplitude spectrum to obtain the Mel cepstrum. This step transforms the signal from the frequency domain into the cepstral (quefrency) domain. We use Discrete Cosine Transform because it produces real-valued coefficients and helps to decorrelate the energy in different Mel bands. The Discrete Cosine Transform also helps to reduce the number of dimensions needed to represent the spectrum.

We then select a subset of the DCT coefficients to be the MFCCs. Typically, we focus on the first few coefficients because the lower quefrency values provide information about the spectral envelope, which is important for describing the timbre of the sound. The higher coefficients, which give the detailed spectral information, are often less necessary for our purposes.

MFCCs are widely used in applications such as speech recognition, speaker identification, and music genre classification because they efficiently capture the timbre of the audio signal. However, with the rise of deep learning in audio processing, there's been a shift towards using raw data. Instead of relying on MFCCs, many modern systems prefer to work with Mel spectrograms. These provide a more direct representation of the audio signal, allowing deep learning models to extract relevant features on its own, potentially leading to better performance in tasks like speech processing.

Conclusion

In conclusion, the extraction of features from audio signals is a process that plays a crucial role in the analysis and understanding of sound. Through the initial steps of analog to digital conversion, framing, and the subsequent extraction of time and frequency domain features, we gain valuable insights into the characteristics of

audio signals. The use of the Mel scale and the creation of Mel spectrograms further enhance our ability to represent sound in a way that aligns with human auditory perception. While features like MFCCs were previously widely used, the development of deep learning has shifted focus towards using raw data for feature extraction. This evolution reflects the advancements in technology and the continuous search for more effective methods in audio analysis. Despite these changes, the foundational principles and techniques discussed in this report remain critical for anyone looking to understand the field of audio signal processing.

References

[1] Murthy, A. (2020, September 9). *5. quantization - digital audio fundamentals*. YouTube. https://youtu.be/1KBLguIXL30?si=k32fKl_4rYpCEiIx

[2] Velardo, V. (2020, July 6). *How to Extract Audio Features*. YouTube. https://youtu.be/8A-W1xk7qs8?si=SKSsPjwyuXYoXnj-

[3] Giannakopoulos, T., & Pikrakis, A. (2014). Audio features. *Introduction to Audio Analysis*, 59–103. https://doi.org/10.1016/b978-0-08-099388-1.00004-2

[4] Vaj, T. (2023, August 26). *MFCC*. Medium. https://vtiya.medium.com/mfcc-801a9fa53617

[5] Deruty, E. (2022, December 15). *Intuitive understanding of mfccs*. Medium. https://medium.com/@derutycsl/intuitive-understanding-of-mfccs-836d36a1f779

[6] Braun, S. (2001). Windows. *Encyclopedia of Vibration*, 1587–1595. https://doi.org/10.1006/rwvb.2001.0052

[7] Fayek, H. (2016, April 21). *Speech Processing for Machine Learning: Filter Banks, Mel-frequency cepstral coefficients (mfccs) and what's in-between*. Haytham Fayek. https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html