Speech Classification

Matt Kubas, *Member, IEEE* and Ethan Morton

***Abstract*—A computer's ability to listen to the words you see and translate them to text you see on the screen, and can use to process requests, has become an integral part of the technological world we live in. We had to look into the characteristics of the audio card and microphones used to create a baseline. Next, we created a speech classifier that can, at a basic level, correlate if the word that was recorded was a word that is in our predefined dictionary. This was done using spectrograms. Furthermore, we looked into the use of Mel-Frequency Cepstral Coefficients (MFCC) to perform the same task by looking into a specific range of the cepstrogram.**

I. Background

TO understand the basic principle of audio processing, we had to begin from the basics, what an audio card does to process audio to send and receive it. Using MATLAB *Audio* and *DSP* Toolboxes, we were able to manipulate audio recording parameters. The sound card that was used was the Creative Labs Sound Blaster Play! 3, which has specifications of 24-bit audio recording and playback at a maximum of 96 kHz with a dynamic range of 93 dB. With the use of this sound card, we were able to simultaneously play and record audio, allowing for the measurement of impulse response, frequency response, and latency. Parameters such as data precision, sampling frequency, and frame size were altered when characterizing the sound card to classify it. These classifications were done with multiple microphones as well, including the Audio Technica ATR1200x, a microphone found in Urbauer 115, and a couple of sets of speakers found in Urbauer 115 and 320E, to play audio for the microphones.

Furthermore, we needed to create our own spectrogram. Spectrograms are graphs that look at the frequency response of frames of real-time signal displayed over time. This means that for a real-time signal, we need to take segments, or "windows" of the real time audio to generate a DFT of the most recent time data. Parameters, such as frame size, signal window, and overlap, can be altered to gauge the behavior and resolution of the spectrogram.

Next, we developed the cepstrogram, similar to the spectrogram, it was looking for the frequency response, but in this case, of the frequency response. The cepstrogram was created by taking the fast Fourier transform (FFT) of the real signal and then taking the inverse Fourier Transform of that signal. The cepstrogram looks to highlight the resonances and harmonic content in the recorded signal. The "frequency" associated with the cepstrogram is known as the quefrency, which represents the period time of the corresponding frequency.

Finally, we look into a specific range of the spectrogram, one that narrows down the resonances in human speech, known as the Mel-Frequencies, and then the DTCT is taken. This means the size of the words we use to correlate to our original speech can be compressed to a much smaller range, allowing for more words in our dictionary to be stored in the same amount of space. This method is known as Mel-Frequency Ceptrum Coefficients (MFCC), and is commonly used to pinpoint features in our speech for speech recognition.

II. Methods

SPEECH classification and recognition requires the ability to look into the behavior of our acquisition devices, the audio cards and the microphones. To begin we started by looking into the latency of the sound card itself, sending a unique signal (a drumbeat provided by MATLAB's audio toolbox) looped from the output to the input. Using the audio toolbox's synchronous player and the *ASIO for All* driver, we were able to play the audio through the loop-back cable (a 3.5mm audio cable to short the input to the output) and using the `xcorr` function, we can determine the delay in reception in the signal compared to the time it was sent. We repeated the process ten times for this measurement and took the average to get the best time possible. Next, the impulse response and frequency response were measured. For the impulse response, an impulse train was created that was about 5 units wide (0.1 ms), so that the physical hardware had a chance to react. Finally, we looked at the frequency response by sending a chirp, or a cosine wave with a linearly increasing frequency ranging from 20Hz to 20kHz, to observe the response for human voice.

Next, we characterized a couple of microphones (ATR1200x and some random mic found in Urbauer 115), we wanted to classify our webcam microphones, but the *ASIO* driver did not allow that. We classified the microphone and speaker setup (neither of which are standard equipment for this test or certified to have a "flat" audio response), as seen in fig. 1. Thus the classification of the system was entirely based on the characteristics of both the microphone and the speakers used and tested. This means that the speakers will have added significant uncertainty to the results of the microphone classification. Regardless, the same method for measuring latency as the sound card was used, but the audio was played through the speakers. The same was done for the impulse response but the pulse had to be extended to be a hundred units wide (2ms) so that the drivers in the speakers had enough time to react. Finally the chirp was played through the speakers to gather the frequency response of the speaker and microphone combination.

FURTHERMORE, a custom spectrogram script was made and designed to mimic the built-in function and gain an understanding of how they are made. The spectrogram is created using time frames of the time domain signal and transformed using a fast Fourier Transform (FFT) based on the discrete Fourier transform (DFT) eq. (1), with each subsequent

Fig. 1: Picture of the experimental setup to measure latency, impulse response, and frequency response of the speaker and microphone combination

frame plotted below the previous to create a continuously varying graph. We looked into how frame size, windowing function, and frame overlap all affect the visual representation of our data. With this background, we were able to begin creating a dictionary of many single-syllable words that we could correlate against. This was done for both the time-domain signal as well as the frequency-domain signal. For the frequency domain, we correlated each frequency bucket of the total time with the corresponding frequency bucket of the word, this was repeated for all frequency buckets and we looked for a high correlation in the buckets to detect if the word was spoken.

A. cepstrogram

THE cepstrogram is composed of time and quefrency as the two axes, while the color of each point was the intensity. To get each data point, the discrete Fourier transform (DFT) was taken fist (mathematically shown in eq. (1)) and the output of that through the discrete cosine transform (DCT) (mathematically shown in eq. (2). The roles performed by each have the same overarching goal of looking at the frequency response of the input, but the DFT can be complex while the DCT is entirely real. The approximate equality in eq. (2) is due to many texts having different representations of the DCT.

$$X_F(e^{j\omega}) = \sum_{n=-\inf}^{+\inf} x[n]e^{-2\pi jkn/N}, \quad k = 0, 1, 2, ..., N-1 \quad (1)$$

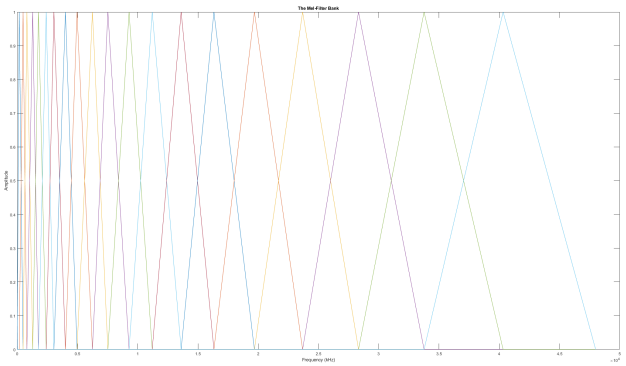

Fig. 2: The Triangle filter bank such that the peak of the triangle lands at each Mel frequency

$$X_C[k] \approx \sum_{n=0}^{N-1} x[n]cos\left(\frac{\pi k}{4N}(2n+1)\right), \quad k = 0, 1, 2, ..., N-1 \quad (2)$$

To complete the MFCC, we first down-sample the log of the spectrum. This is because the cepstrogram is good at extracting the fundamental frequency from an envelope. Thus to create this envelope we use a triangular filter bank applied to the log of the DFT to get the average power at each selected frequency, this leads to a down-sampling of the frequency spectrum and a smaller size of the array. The Mel scale eq. (3) [2] is then applied to this by centering each triangular pulse such that it is at each Mel-frequency as seen in fig. 2. Once this is applied to the spectrum, this creates what is known as the Mel-envelope, essentially picking out the average power of the signal and smoothing out the spectrogram. This is then transformed once again to get the Mel-Frequency Ceptral Coefficients (MFCCs) [1].

$$m = 2595\, log_{10}\left(1 + \frac{f}{700}\right) \quad (3)$$

To summarize, to take the MFCC of a time domain signal, we first take the DFT using a fast Fourier transform (FFT) to take the signal into the frequency domain. Then we apply the Mel filter-bank, a set of triangular pulses centered at the Mel frequencies, which are designed to pinpoint the frequencies at which the average energy of that frequencies is the most unique between words. Finally, the DCT is taken to create the cepstrogram, which can make resonances and the fundamental frequency more apparent to the eye and easier to compare. The Mel filter-bank is adjustable to the number of Mel frequencies we want to include, this means that we map equally spaced Mel frequencies from roughly 20Hz to 20kHz creating the triangular waveform, going from 1000 or more frequency buckets to roughly 60 or less, and this is adjustable.

B. Correlation

The final step in our audio classification and speech recognition, which

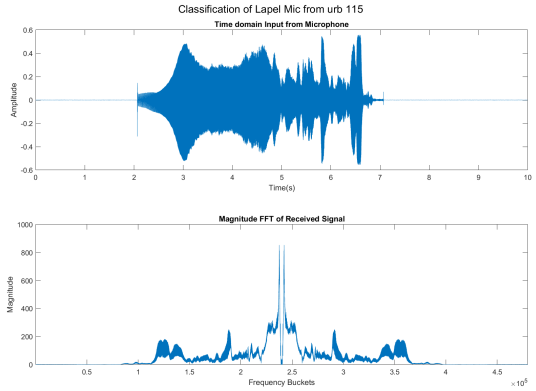

Fig. 3: Frequency Response of the ATR1200x and Z-1 speakers

III. Results

A. Device Classification

When classifying the equipment, the Sound Blaster Play! 3 had an impulse response that resembled a low pass filter, meaning that if any frequency was too high, the sound card was unable to play it. This is reasonable since physical devices require time to move electrons in a wire and thus cannot pass infinite frequency signals. The classification of the microphone was limited to the range of the frequencies, thus the classification on these systems was the system classification of the microphone and speaker combination. In fig. 3, the Audio Technica ATR1200x had audio played to it by a set of office speakers with model number Z-1 on them. The spike in the low frequencies was due to the speakers making a low-frequency ”pop” when turning on leading to another point of error.

B. Spectrogram, Cepstrogram, & MFCC

The spectrogram in its basic form presents the frequency content of the system vs time, with a heat map quantifying the strength of the content at that point. The spectrogram has three parameters that we can vary, frame size, windowing type, and overlap. This can be seen in fig. 4, where the window time/frame size is varied as well as the overlap. With a larger overlap, there is more smearing in the signal, due to so much of the signal being repeated. The window time/frame size determines if we have higher accuracy in the time or in the frequency. The smaller the frame size, the more accuracy in time that we have but the less accuracy in frequency that we have, and the opposite is true for larger frame sizes.

The Cepstrogram simply is the inverse Fourier transform of the spectrogram. An example of this for the spoken word ”nine” is shown in fig. 5. This figure shows the unique characterization of the word ”nine” in both the frequency domain and the quefrency domain. One can see that the harmonics of the frequency domain signal are captured by the cepstrogram allowing for a different type of audio correlation and recognition later on. One can also see that even in the cepstrogram, there is a bit of harmonics. This isn't a result of

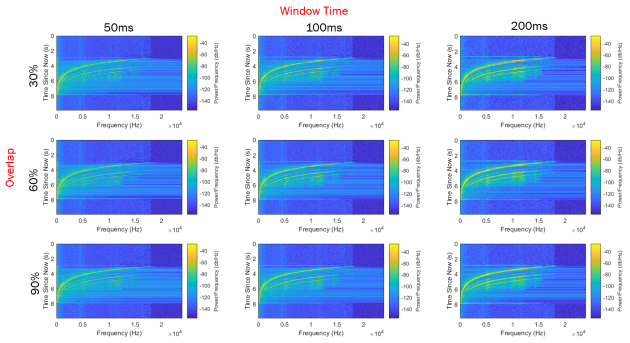

Fig. 4: Varied the parameters: window time and overlap, for the custom spectrogram function

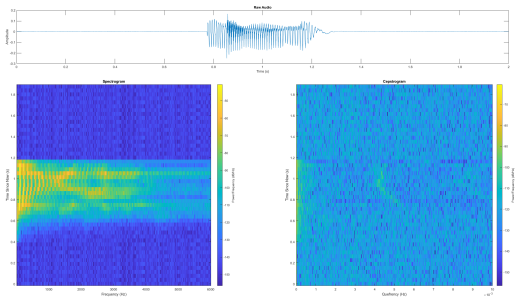

Fig. 5: Raw audio (time domain), spectrogram and cepstrogram for the spoken word ”nine”

the harmonics in the human voice. Rather, it is because we are taking a Fourier Transform of a finite signal. Just like how windowing a time-domain signal can introduce higher order harmonics, so too can windowing the spectrogram. Because our frequency spectrum is finite, we see the higher quefrency harmonics.

The Mel frequency ceptro coefficients (MFCC) for the spoken word ”nine” is shown in fig. 6. The figure also shows the spectrogram and Mel frequency spectrogram of the signal. The MFCC and Mel frequency spectrogram are roughly two orders of magnitude smaller in range compared to the spectrogram. We chose to use 60 Mel frequency coefficients because it made the graph look more presentable, but could be shrunk even lower while still allowing a computer to decipher and correlate the data, via more advanced algorithms such as Dynamic Time Warping (DTW) or ML techniques such as a Support Vector Classifier (SVC), neural net, or other model. These MFCC values correspond to a combination of features from the original frequency spectrum, essentially downsampling the original spectrum to a smaller but (hopefully) more easily interpretable feature set. In many machine learning audio processing models, getting the MFCCs and their ”delta” and ”delta-delta” (first and second order difference) is a preprocessing step to input into the network.

C. Correlation

Due to time constraints, we were only able to use the correlation function `xcorr` for the spectral data and found that

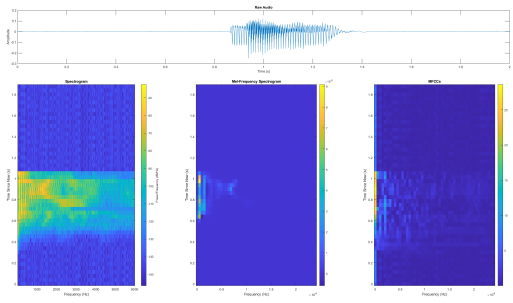

Fig. 6: Mel frequency ceptro coefficients, along with the spectrogram and Mel frequency spectrogram for the spoken word ”nine”

the MFCC was taking too long to justify using the correlation to compute if a word was said in real-time. The initial idea was to use 2-D correlation, but that proved inefficient and ineffective. Instead, we changed gears towads 1-D correlation and simply took each frequency bucket of the signal and correlated it against each frequency bucket of the dictionary word. This would need to be repeated for each time step, but showed a much more reliable set of values. Another technique that we did not get to implementing is doing a 2D windowing process for the spectrogram. In this process, we would take the spectrogram of each of our dictionary words, and ”slide” it across the real-time spectrogram. We then compute the euclidean distance between each overlapping point, and sum it up to produce an overall ”distance” metric, to represent the similarity between two slices (essentially taking a dot product).

The efficacy of the correlation of spectrogram-based signals to our dictionary and test signals is shown in the confusion matrix in fig. 7. Most of the spoken numbers have a high correctness rate while the spoken numbers ”two” and ”three” confused the correlation function and did not even get one of them correct. It appears that ”two” and ”three” are very similar to other spoken numbers on the spectrogram.

IV. Conclusion

OVERALL, we wish we could have gone further with this and could have created a more accurate real-time speech-to-text script, that with a good dictionary could detect many more works. The MFCC’s seem like a great way to classify audio but requires lots of code optimization to run efficiently enough to be able to run real-time classification. It would also be interesting to explore possibly using machine learning to complete the word correlation using our compressed MFCC dataset.

We wish we had more time to accomplish more but we did what we could. I hope that we can come back to this in the future to implement, at least mildly, the MFCC correlations, and determine if they are more accurate than the spectral correlation, as is likely the case (as well as being more performant with a from-scratch implementation, no tech debt from me [Ethan] furiously abstracting out functions).

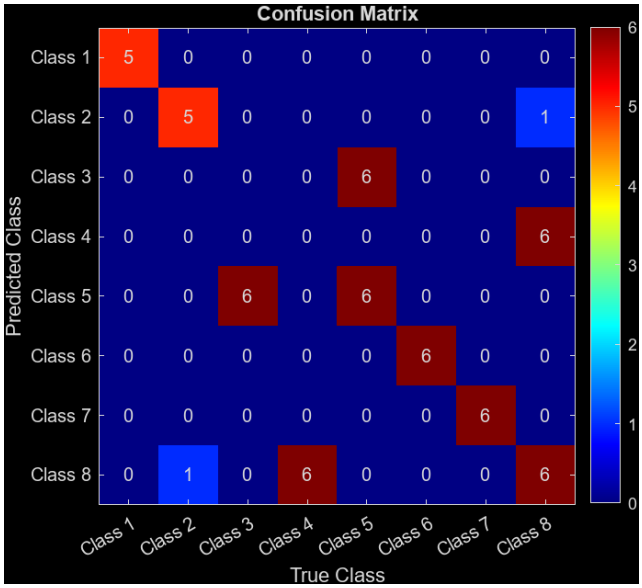

Fig. 7: Confusion Matrix

References

[1] Tom Bäckström et al. *Introduction to Speech Processing*. 2nd ed. 2022. DOI: 10.5281/zenodo.6821775. URL: https://speechprocessingbook.aalto.fi.

[2] *Mel scale - Wikipedia*. https://en.wikipedia.org/wiki/Mel_scale. Accessed: 2024-09-24. 2024.