Classification of Audio Signals

Blake Diamond
Electrical and Computer Engineering
UC Santa Barbara
Goleta, California, United States
bdiamond@engineering.ucsb.edu

Himangshu Chowdhury
Electrical and Computer Engineering
UC Santa Barbara
Goleta, California, United States
himangshuchowdhury@ucsb.edu

***Abstract*—This document details the process of creating an audio classifier that is capable of classifying an audio signal input with a given sample frequency into one of five categories. In order to achieve our goal, we have implemented significant temporal and spectral feature extraction before performing a k-nearest neighbors algorithm.**

I. Introduction

The task being performed in this challenge is to create an audio classifier that is able to classify different environmental audio signals into five different categories: rain, waves, fire, crickets, and birds. Our implementation calls for extracting temporal and spectral features from the input signal, which will be detailed in the next section, to define feature vectors that describe the characteristics of the input vector. Our approach then uses the k-nearest neighbors method to compare the feature vector generated by the input signal to the feature vectors that characterize 31 audio signals of each category that were part of the given dataset. Additionally, it is the aim of our algorithm to be invariant to volume and the variety of possible sounds that a particular category; for example, rain, can sound very different based on the intensity of the storm, average size of rain droplet, amount of raindrops per unit area, and the surface that absorbs the rain.

II. Methods

A. Pre-Processing

After reading in the audio file, the sampling frequency for the input is used to create a specific time interval which will serve as a single block in our overlapping block processing method; based off of empirical data from past experiments, the optimal time interval for a single block is 50 milliseconds. This was done in order to reduce the negative impact of lower sampling frequencies being input to our algorithm. Next, the block length and hop length, in terms of samples, is determined before entering the feature finding aspect of our algorithm, which is implemented in an overlapping block processing, which utilizes an overlap ratio of 0.8.

B. Time Domain Processing

In our time domain processing, we implemented an overlapping block processing technique in order to extract one temporal feature of the speech signal.

The sole temporal feature that was computed was the energy, which is described in equation 1. For each block, the average energy is computed and after block processing has concluded, the standard deviation of that average energy vector is taken in order to describe the temporal shape of the signal. While we are only using one time domain feature, we felt that it was important to have at least one characteristic that describes the time domain of the audio signal.

$$E[i] = \frac{1}{N}\sum_{n} x[n]^2 w_i[n] \tag{1}$$

C. Frequency Domain Processing

While temporal features are valuable, it is perhaps more important to compute the frequency content in order to extract features that uniquely describe the type of signals . In order to accomplish this task, we elected to compute 7 spectral features: cepstral coefficients, average spectral centroid, half-rectified spectral flux (equation 2a and 2b), pitch tracking, spectral crest, spectral flatness, and spectral entropy. For all frequency domain computations, a hann windowing function was implemented to compute the discrete Fourier transform, which emphasizes samples towards the center of the block. All of these features are eventually summarized into a single value, either standard deviation or mean based on the nature of the specific feature, and then this feature vector is used in the k-nearest neighbors method.

$$SF_R[i] = \frac{1}{N}\sum_{k} H(|X_i[k]| - |X_i[k-1]|) \tag{2a}$$

$$H(a) = (a + |a|)/2 \tag{2b}$$

The average logarithm of spectral flatness describes if the power of the spectrum of the signal is evenly distributed amongst the signal or concentrated in a relatively small number of bands. Spectral crest is computed and then averaged because it indicates the extremity of peaks in the waveform, since it essentially act as a ratio of the peak value to the average value of a particular block. Spectral entropy is utilized in order to estimate the amount of disorder within the frequency content of the signal.
The second through fifth cepstral coefficients are computed in order to describe the shape of the frequency spectrum that is calculated for a given block; this computation is described in equation 3. The average of each of the four coefficients is treated as an individual feature that is used in the final feature vector.

$$c_i[k] = DFT^{-1}\log(|STFT\{x[n]\}|) \tag{3}$$

D. *K-Nearest Neighbors*

To classify the input sound signal as one of the five environmental sounds, we used a machine learning approach, where we divided each of the sounds' files into 3 sets of files – training, validation and testing. We checked the k-nearest neighbors to each of the validation files and assigned a label to it based on a voting system; the labels being 0-4 based on which of the environmental sounds it corresponded to. To implement this algorithm, we took each validation file and found its distance from each of the training files that we had gathered using different methods discussed below:

a) Euclidian: Calculates the distance by evaluating the RMS difference between each element of two vectors.

$$d_{st} = \sqrt{(x_s - x_t)(x_s - x_t)'}$$

b) Squared Euclidian: Calculates the distance by squaring the RMS difference between each element of two vectors.

$$d_{st} = (x_s - x_t)(x_s - x_t)'$$

c) Manhattan: Calculates the distance by evaluating the sum of the difference between each element of two vectors.

$$d_{st} = \sum_{j=1}^{n} \left| x_{sj} - x_{tj} \right|$$

d) Minkowski: Calculates the distance by evaluating the p^{th} root of the (difference between each element of two vectors) raised to the power of p.

$$d_{st} = \sqrt[p]{\sum_{j=1}^{n} \left| x_{sj} - x_{tj} \right|^p}$$

e) Mahalanobis: Calculates the distance by evaluating the correlation between the difference between each element of two vectors and the correlation matrix.

$$d_{st} = \sqrt{(x_s - x_t)C^{-1}(x_s - x_t)'}$$

After finding the distances and storing them in an array along with their respective indices, we sorted them ascendingly. Depending on the value of k, we would then take the first k elements in the array and use their respective indices to determine the k-nearest neighbors. Using a majority voting system, we would then classify the validation file, assigning it a label. After measuring the accuracy of the algorithm by testing all the validation files and checking the predicted labels against their assigned labels, we implemented a looping algorithm that determined the best possible value of k that would yield the highest accuracy. Using this value of k for our test files, we then confirmed the validity of our algorithm.

III. RESULTS

In this section, one 5 second audio signal is characterized using a sample frequency of 44.1 kHz. Figures 2-7 show various temporal and spectral features that were calculated in order to uniquely characterize the audio signal into a unique feature vector.

In figure 2, the cepstral coefficients can be seen; the mean of each cepstral coefficient was computed in order to best summarize the data without using uniqueness that differentiates this signal from those in other categories. After thoroughly researching what the meaning of each of the features was, we came to similar conclusions for all of the other spectral features: taking the mean of each feature over time was the best way to summarize the data into a single integer. In our calculations, we used 8 features, but could have also included the standard deviation as a tool of summarization; the main reason that we did not do this was because of time constraints. It can be seen that most of the features describe the signal in some unique way, but in figure 4, the Spectral Centroid is shown. While this feature was summarized and used in our k-nearest neighbors method, it was not unique for different signals and had very similar shape and value for various signals across multiple sound categories.

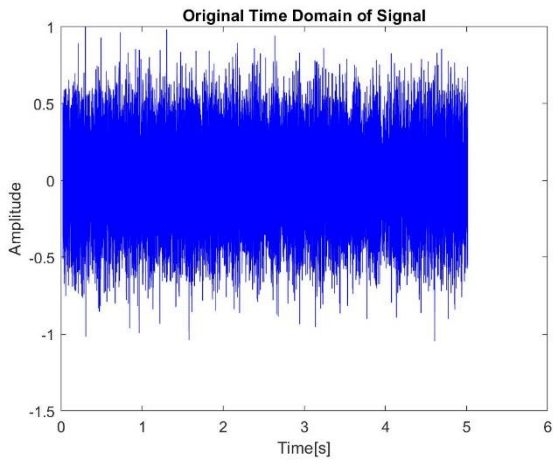

Figure 1: Time Domain of Original Signal

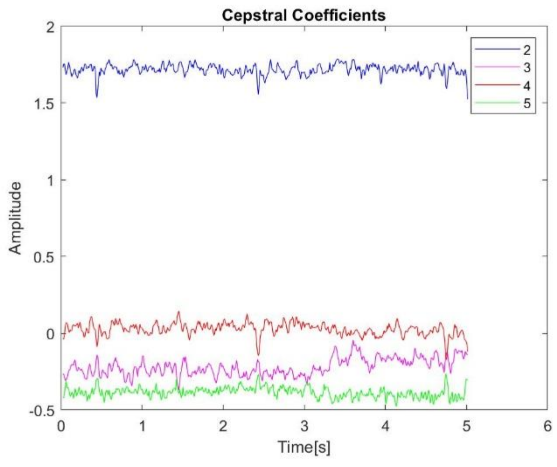

Figure 2: Cepstral Coefficients vs. Time

Figure 3 shows the temporal energy of the signal. This feature was somewhat unique because it is the only time-based feature that is used in our prediction algorithm. Rather than taking the mean in order to summarize this data, we opted to utilize the standard deviation to better describe the unique shape that an audio signal of a particular category takes on.

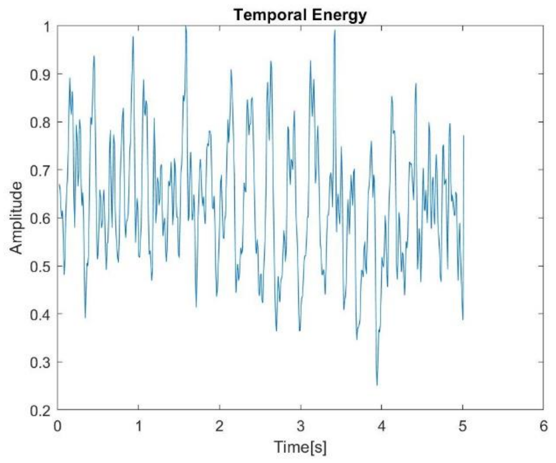

Figure 3: Temporal Energy of Original Time Domain Signal

The KNN algorithm helped us determine which features were important in classifying an audio file correctly. We also found some features that did not affect the performance of the classification, namely spectral centroid, spectral crest, spectral flatness and entropy. Contrary to intuition, these features, which we originally thought could have been important, turned out to be inadequate for our purposes. Out of all the distance calculating algorithms within KNN, Euclidian seemed to be the clear winner while Manhattan and Mahalanobis were close runners up. Over a series of tests, and analyzing their results, it was determined that the optimal value for k with the feature space we implemented was 9.

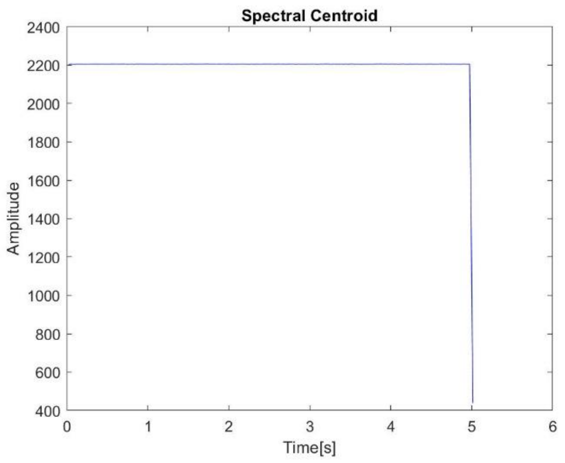

Figure 4: Spectral Centroid vs. Time

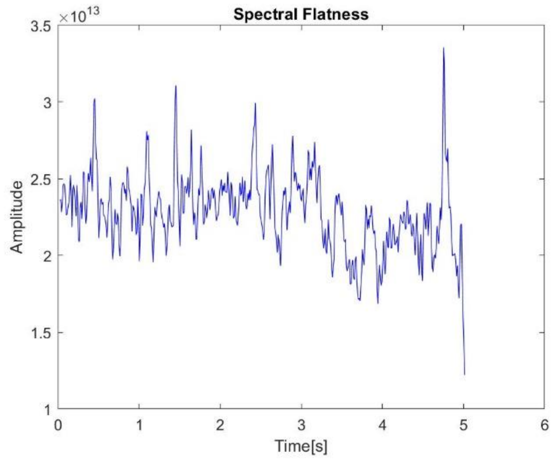

Figure 5: Spectral Flatness vs. Time

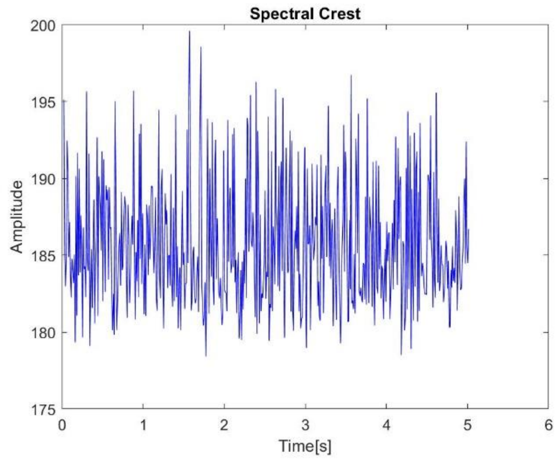

Figure 6: Spectral Crest vs. Time

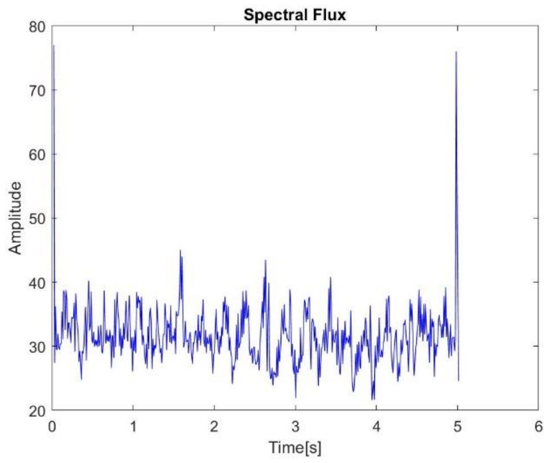

Figure 7: Spectral Flux vs. Time

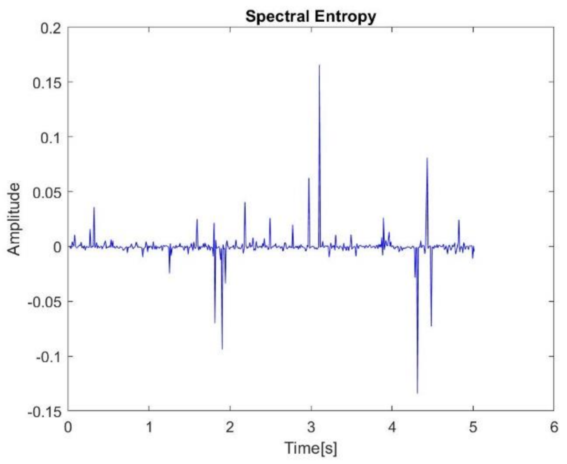

Figure 8: Spectral Entropy vs. Time

IV. DISCUSSION AND CONCLUSIONS

In completing this assignment, we learned quite a few things including feature extraction, feature summarization, and designing our own k-nearest neighbors method. Similarly, to previous assignments, it was not setting up the skeleton of the problem that took much of the time. Instead, choosing the best features and summarization methods was highly considered when we were designing our algorithm. Additionally, fine-tuning the k-nearest neighbors algorithm by adjusting the number of neighbors and distance method was a process as well. Automating the process of fine tuning the parameters to find the best possible accuracy was a great tool in saving time on this assignment.

In hindsight, we were able to achieve a much higher accuracy using MATLAB's built-in k-nearest neighbors functions but learned much more about feature extraction and classification by implementing our own algorithm. Using our own classification algorithm, we were able to achieve a test accuracy of 60%. Since we were able to achieve higher accuracy with built-in functions, this suggests that our k-nearest neighbors implementation could be optimized significantly but was not due to time constraints. Another option would have been to use a different type of classifier, but we did not opt to go with this strategy because of our prior familiarity with the k-nearest neighbors method. In addition, choosing a distance-weighted voting system could also have yielded better results.

Perhaps it would have been more beneficial to us to be more selective about the features that we included in our summarized feature vector. For better or for worse, we utilized a shotgun method: many features that try to extract a wide spectrum of properties from the signal. If we were to use more carefully selected features and only, let's say, 2 features, it would have certainly affected the speed at which our algorithm runs and could have improved test accuracy. Another potential way to improve our feature extraction would have been to employ more advanced statistical computations that standard deviation and mean.

V. AUTHOR CONTRIBUTIONS

The work for this project was very evenly distributed between Blake and Himangshu. We met on several different evenings to work together to optimize the k-nearest neighbors method and feature extraction. During the time that we were simultaneously working on the project, one partner would be adding features and the other would be working on the k-nearest neighbors method. Himangshu mostly focused on implementing the classification task, while Blake focused on feature extraction and summarization. After the skeleton of the project had been erected, we worked together to streamline the voting process of the algorithm and dial in signal processing levers that helped us to achieve a higher accuracy.