Speech-Music Discrimination Using ANN and KNN Classifiers

Lawrence Godfrey, and Matthew Lock
Department of Electrical Engineering
University of Cape Town

Abstract

Classification of audio files into separate classes can be incredibly useful for many applications. This report aims to identify important audio features which can be used to distinguish an input audio signal as containing either speech or music. These features will then be used to train two classification models, and the accuracy and efficiency of these models will be evaluated and compared against one another to determine which is more appropriate for different potential applications. Highlighted in this report is the ability of these different training models to handle both higher and lower dimensional datasets.

I. Introduction

FOR the vast majority of us, interpreting and interacting with the mammoth amount of information presented by the everyday world is almost a completely trivial task. As humans we have gained an intuitive understanding for almost every situation that guides us through everyday life, without so much as second thought to consider the complexities of some of the tasks we are performing. One such example that this paper focuses on is the ability to distinguish between human speech and a musical piece. While the implication to you may be as simple as deciding whether to listen to some light humour presented by a radio presenter or some wake up music on your morning drive, there exist a vast array of applications to which capturing such an intuitive understanding would be greatly beneficial. These applications may include low bit-rate audio coding and content-based audio and video retrieval [1], or even automatic speech recognition algorithms that could be used to assist visually impaired individuals in interacting with their smartphone.

The difficulty in the challenge of automated discrimination between music and speech lies in the fact that this simple understanding and ability to distinguish between speech and music is inherently linked to our inexplicable human intuition of what music and speech are fundamentally. Unfortunately, capturing this understanding for use in an automated system is no simple task. Such systems require a deeply rooted mathematical understanding of the two in order to discriminate with the same level of accuracy as a human counterpart. Hence there have been many attempts at extracting both temporal and spectral features from such signals, in the hopes of identifying distinguishable patterns between the two that would allow for accurate discrimination.

These features are then used in combination with different classifier models in other to make the distinction. Two commonly used classifiers are the K-Nearest Neighbours (KNN) algorithm and an Artificial Neural Network (ANN). In this report we investigate the performance difference between these two classification algorithms in the context of speech-music discrimination, specifically focusing on which classifier is more appropriate in the context of different accuracy and timing requirements. While accuracy is obviously very important in a classifier, the speed of classification can also be a large factor in some cases, especially when taking into account the huge amount of data which would need to be classified in some applications. The performance metric used is thus a combination of both speed and accuracy of the final models.

II. Literature Review

This literature review seeks to outline the required theory behind the structure of speech and music signals, as well as features and classifiers which have been used to successfully discriminate between music and speech, and which will be the focus of our own investigation.

A. Analytical Features

Saunders [2] recognised that general speech patterns tend to contain a very regular structure and explains that continuous speech is composed of syllables, where each syllable is then further composed from consonant clusters followed by vowels, often then followed by trailing consonants. This pattern can be easy to discern visually when looking at the signal waveform, and can be seen in Figure 1 where dissipating amplitudes caused by consonants trailing a mixture of vowel-consonants clusters can be seen.Figure 2 on the other hand shows how this clear distinction cannot be made with music signals as tehy appear a lot more messy. Saunders further points out that vowels are usually presented with periodic components containing higher levels of energy due to glottal excitation and the nature of an unrestricted vocal tract, where most of the spectral energy can be found in the low frequencies. Consonants on the other hand are produced by frication, a constriction in the vocal tract causing a turbulent air flow, attenuation of acoustic energy and damped resonances. The resulting signal is a noise-like signal producing spectral energy distributed more towards the higher frequencies whereby speech is a succession of syllables composed of short periods of frication followed by longer periods of vowels or highly voiced speech.

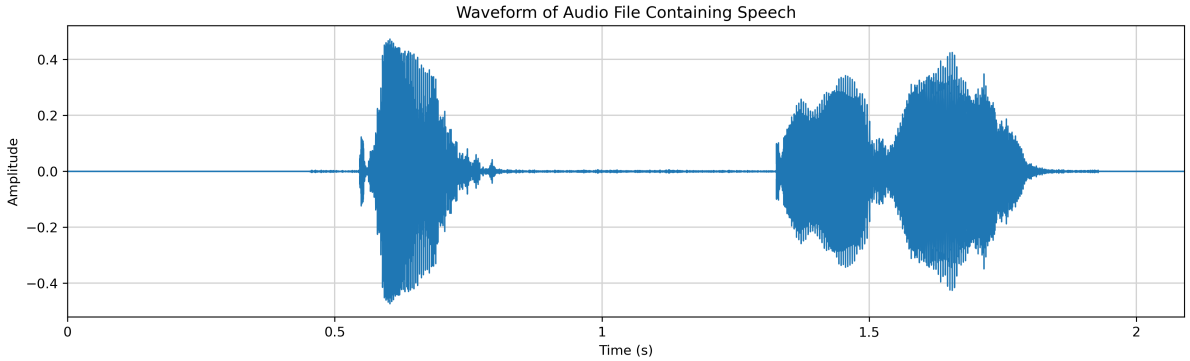

Fig. 1: Waveform of Speech Signal. In this signal we can visually discern the various components of speech. The particular recording is somebody saying the phrase ”Go, do you hear?” There is a pause between the two components in speech that can be seen visually. Furthermore we are able to see the see the that at around 0.6 s there is a sudden increase in amplitude associated with the consonant-cluster pair caused by frication with a much higher amplitude visible for the duration of the following vowel. This pattern can then be seen again after the pause two of these frications are clearly visible as two separate clusters within the signal.

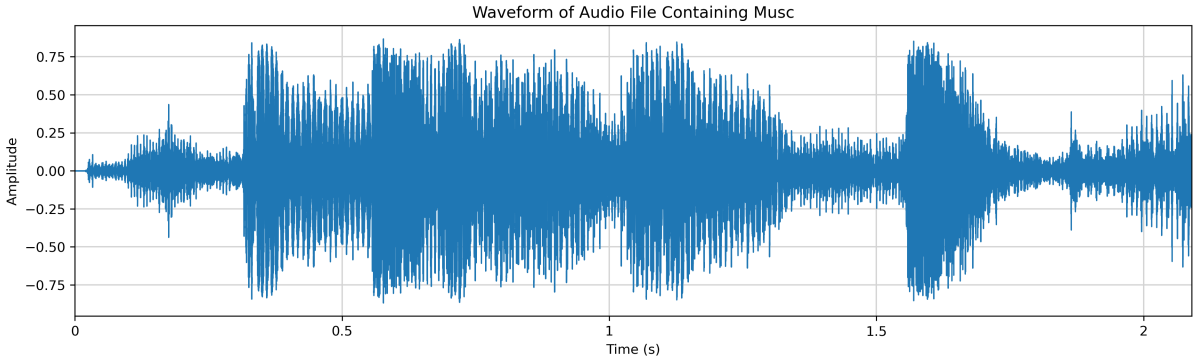

Fig. 2: Waveform of Music Signal. Unlike with speech, the signal waveform of music tends to be a lot more messy, and it is almost impossible to see discern what may be occurring within the signal. We are able to discern which periods of the waveform are dominated by loud and abrupt sounds, but we are not able to make accurate predictions as to what the structure and cause behind these abrupt spikes are like we can with speech.

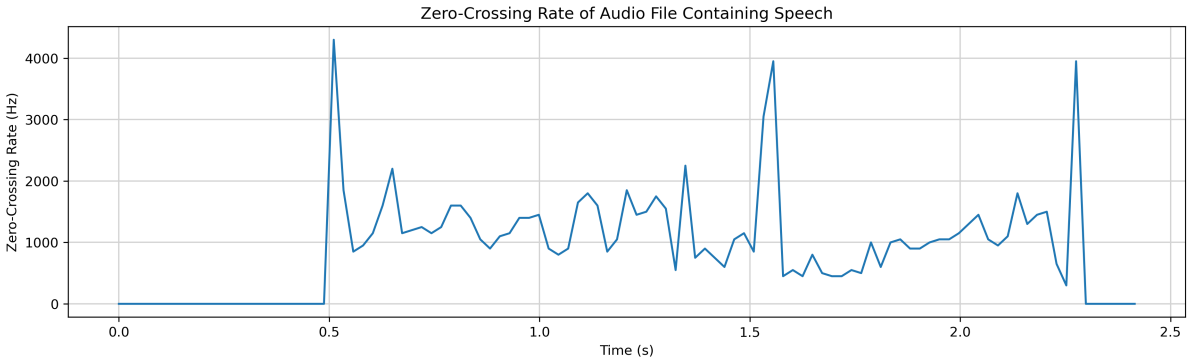

Fig. 3: Zero-Crossing Rate (ZCR) of Speech Signal. One can easily see the abrupt increases in the ZCR of the signal caused by frication during speech. While these changes occur within music , the are in line with the average distribution of the music. While with speech, the these abrupt changes tend to have a dramatic and measurable effect on the variance of ZCR the signal.

One of the temporal features that can be used to distinguish these patterns present in music and speech is the average variance of the zero-crossing rate (ZCR) of the signal waveform. The dominant frequency principle described by Kedem [3] states that when a certain frequency band carries more power than other bands, it attracts the normalized expected number of zero crossing per unit time. In another sense, when a frequency becomes dominant in the spectral signal it can be detected using the zero-crossing rate. This is useful in the discrimination of speech and music as while music may also contain speech, the frequency of speech will generally not be as dominant compared to pure speech signals or signals which emphasise the speech component. This can be seen in Figure 3 where there is an abrupt rise in the ZCR during periods of fricativity occurring at the start and end of words, resulting in a skewed distribution of the ZCR. Music signals on the other hand do not display these abrupt differences in the ZCR and rather show a more normal distribution of the ZCR with lower levels of variation. It must be noted though that due to the dominant frequency principle, music signals may show a ZCR distribution similar to that of a speech signal if the speech component is the dominant feature in the audio recording, causing the signal to be misclassified. An example of this may include solo vocal performances. This feature was used in a study by Scheirer at all [4] and while we not know the exact implementation used, the classification error of 18 ± 4.8 found shows that the ZCR will be valid feature for speech-music discrimination.

The next temporal feature that we considered for use in discriminating between speech and music signals was the percentage of “low-energy” frames. Panagiotakis et al [5] observed a difference in the root mean square (RMS) distribution of amplitude values between speech and music signals. The difference in distribution is demonstrated in Figure 4 where more samples of the speech file can be seen to be below the 50% mean RMS, this is explained in more detail below.

A simple means of extracting information regarding the distribution of the RMS amplitude is to determine the percentage of “low-energy” frames, which is described by Scheirer at all [4] as the proportion of frames with RMS power less than 50% of the mean RMS power within a one-second window, with proportions then being averaged across every second of the signal. In general the energy distribution for speech is more left-skewed than for music, meaning that there will be a larger proportion of frames below the 50% threshold (quiet frames). This is due to the nature of speech containing periodic pauses between the succession of words, where music can generally be seen to have a more continuous nature. Reasoning for proportions only being calculated over frames within a one second window include accounting for the possibility of elevated levels of excitation in the signals. Should there be short outbursts of ‘screaming’ or ‘elevated sound levels’, we do not want the average RMS value of each frame being affected by such outliers in the overall signal progression. The discrimination error achieved by Scheirer at all for the particular feature was $14 \pm 3.6\%$, proving to be an effective means for music-speech discrimination.

Panagiotakis et al further showed that the distribution of RMS values was independent of the distribution of the ZCR. Hence it is possible to exploit the difference in RMS distributions to discriminate between music and speech files, while simultaneously using information regarding the ZRC distribution without either of the features providing “redundant” information.

The third temporal feature we considered was the spectral “flux” [4] otherwise referred to as the Delta Spectrum Magnitude. As one can easily imagine, music generally has a higher rate of change compared to speech and goes through more drastic frame-to-frame changes. Speech on the other hand goes through less drastic frame-to-frame changes as there are alternating periods between periods of transitions involving consonant and vowel boundaries, and periods of relative stasis involving purely vowels. Hence we expect a more constant rate of change for speech. One way to measure this is to find the maximum amplitude value within each frame, and then to calculate the 2-norm of the frame-to-frame spectral amplitude difference vector defined

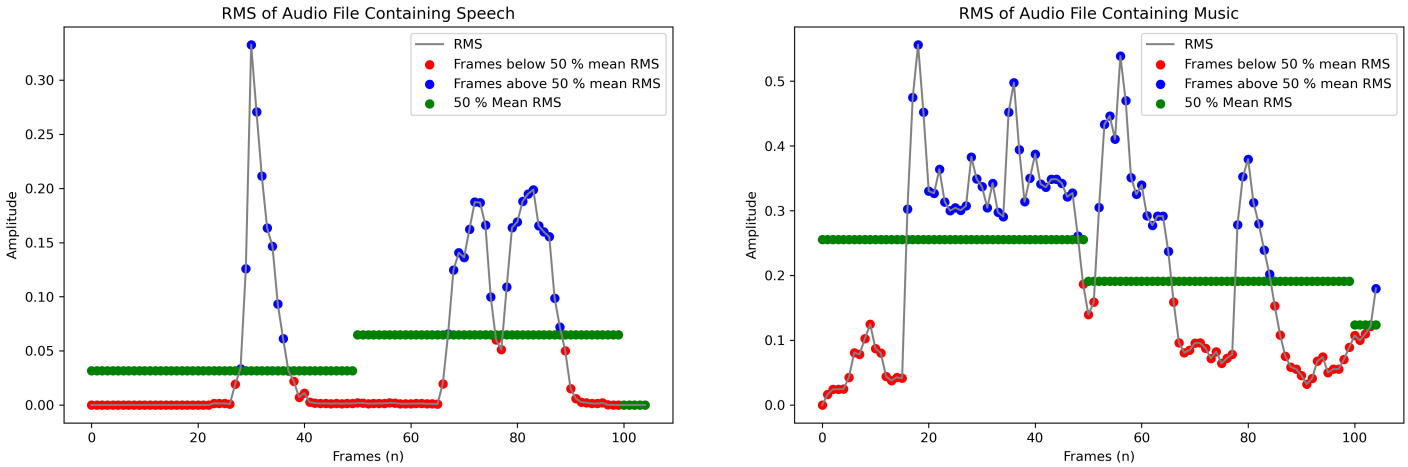

Fig. 4: Comparing the Percentage of Low Energy Frames between Speech and Music. Shown in the figure is mean RMS of each frame within the signal, as well as 50% of the mean RMS over each second of data. Points falling below or above this line are plotted in red and blue respectively. Quite evident is that the speech waveform has far less points above this threshold than music, and a hence great deal more points below 50 % of the average RMS. This can be attributed to the fact that there are varying levels of loudness within a speech signal within a one second period, with moments of relative quietness dominating the signal, while music tends to show a more evenly distributed level of loudness.

by Equation 1. The discrimination error achieved by Scheirer at all [4] for this particular feature was $5 \pm 1.\%$, proving to be the most effective means for music-speech discrimination in that particular study.

Bogert et al first introduced the notion of cepstral frequencies [6], which don't lie in the time domain nor the frequency domain. Bogart et al termed this domain the quefrency domain, and the frequencies in this domain the cepstral frequencies. Both these names are derived from the words frequency and spectral, with some of the letters rearranged.

The cepstral frequencies are found by first Fourier transforming the time domain signal, then taking the log of the magnitude of the Fourier transform, and finally applying a cosine transformation to that [7]. From this we can see that the final spectrum will not lie in the time domain or the frequency domain. A commonly used feature for speech analysis are the Mel-Frequency Cepstrum Coefficients (MFCCs). The Mel-Frequency Cepstrum is a scaled version of the Cepstral frequencies, such that the new frequencies more closely approximate how humans perceive frequencies (humans are more sensitive to changes in frequencies at the lower end of the audible range).

It turns out that the MFCCs of an audio signal are a useful way of extracting information about the timbral qualities of sound [8], making it a good feature for both music and speech classification tasks.

$$|||X_{i+1}| - |X_i|| \tag{1}$$

B. Classifier Models

A simple way of classifying data samples as music or speech will be a K-Nearest-Neighbours (KNN) classifier. As pointed out by Guo et al [9], the KNN classifier is a simple and effective non-parametric classification method. The classifier works by means of retrieving the nearest neighbours for a particular data record, forming a neighbourhood, and then deciding by majority voting as to what the classification of that data record should be. The distance metric used in determining the nearest neighbors is a euclidean distance metric, where the model can work with or without consideration of distance-based weighting in making predictions. Furthermore the number of neighbours (K) that the model will consider is a user parameter and will need to be set before performing classifications. Hence the results of a KNN classifier can be biased by the selected K value, and needs to be considered when making predictions. According to Guo et al this is one of the major drawbacks of a KNN classifier as results can be highly dependent on a "good" K value. Methods for determining an appropriate K value have been proposed by Wang [10], but such methods involve aggregating sets of nearest neighbours and generally have a poor efficiency with an expected performance of O(n^2).

Something else that should be considered when designing KNN algorithms is the distribution of the feature space. If some features are larger than others, they may naturally have a larger influence in the euclidean distance metric when determining nearest neighbours. Hence it is important to make sure that datases have somewhat been normalised to avoid these types of situations.

The final potential drawback is that KNN classifiers may be limited by the "curse of dimensionality" discussed by Kouiroukidis et al [11]. In essence the dimensionality curse states that in higher dimensional spaces distances, between nearest and farthest points from query points become almost equal. Chen [12] further explains that this issue is created by the fact the number of samples needed to estimate an arbitrary function with a given level of accuracy grows exponentially with respect to the number of input variables. This may pose an issue when determining the nearest neighbours for a higher dimensional data space as the KNN's accuracy may be limited by the number of data items available in our dataset. While it could be possible to generate an extremely large dataset, this would probably be highly expensive in terms of time needed to train and produce predictions.

While most statistical approaches to classification use models which suffer at high dimensions, the ANN does not. This is shown by Hamamoto et al in a paper which compares the effects of high-dimensional features on classifiers which are trained on relatively small datasets [13]. Hamamoto et al also discuss the fact that the generalization error decreases as the training set size increases. Therefore ANNs are best utilized when being trained on large amounts of data, using a high dimensional feature space.

The ReLU (Rectified Linear Unit) activation function is commonly used in modern ANNs. It maintains some of the desirable properties of a linear activation, while also being able to learn more advanced decision boundaries than a purely linear function. It also requires less computation than a Sigmoid or Tanh activation function, due to its simple shape [14].

The simplest ANN optimizer, batch gradient descent, has proven to be much too computationally expensive when training ANNs on large datasets. This is because when using batch gradient descent, every example in the training data needs to be taken into consideration when calculating the new set of parameters, and it might need to calculate these parameters hundreds of times before arriving at a global minimum. Mini-batch gradient descent is less computationally expensive since it only takes a small subset of the training data into consideration every time it takes a step, but this does result in a lower accuracy, since the parameters might never correspond to the actual global minimum, and will sometimes only get close to this minimum [15]. More advanced optimizers introduced the ideas of acceleration and momentum when calculating the ANNs parameters. This allowed larger steps to be taken depending on how "steep" the descent is in the error-space.

The Adam optimiser was introduced by Diederik P. Kingma and Jimmy Lei Ba in their paper "Adam: A Method For Stochastic Optimization" [16]. In this paper they show how the Adam optimizer uses individual adaptive learning rates for parameters, as opposed to a single, static weight for all parameters. They also show that the Adam optimizer is computationally efficient, and well suited for training on large datasets.

III. Methodology

The implementation and evaluation of the classifiers can be broken up into four main steps: data acquisition, feature extraction, model training and model evaluation. These steps will be discussed in detail in the subsections that follow.

A. Feature Extraction

A total of four features were extracted from the dataset for the purposes of training the KNN and ANN classifiers. Namely the average variance of the zero-crossing rate (ZCR), the percentage of "low-energy" frames, the spectral "flux", and the MFCCs of the signal. The reasons behind picking these particular features have been explored in previous sections.

1) Average Zero-Crossing Rate (ZCR): For the purpose of this investigation the ZCR over a frame has been defined as the number of time-domain zero crossings per unit time within frames [2], where a frame is defined to be a non-overlapping portion of the audio signal. This can be described more fluently by Equations 2 and 3 where $x(n)$ is the time domain audio signal, N is the number of samples in a frame, and t_0 is the period (s) of a frame.

$$\text{ZCR}[\sum_{n=1}^{N} |\text{sign}(x(n)) - \text{sign}(x(n-1))|\ ].\frac{1}{t_0}\ \text{[Hz]} \tag{2}$$

$$\text{sign}(a) = \begin{cases} 1, & a > 0 \\ 0, & a = 0 \\ -1, & a < 0 \end{cases} \tag{3}$$

This will produce a single ZCR value, and will be used to produce a ZCR value for each frame. The period of each frame has been set to 20 ms with N being directly related to the sample frequency of the signal. Since all the dataset items have been sampled at a frequency of 22.05 kHz, the nominal N value will be 441. The variance across each second of data will be calculated (i.e. over every 50 frames), and the average of these variance values will be determined to produce a final value for the average variance of the zero-crossing rate

2) Percentage of "Low-Energy" Frames: The aim of this feature is to determine the percentage of frames that fall below 50% of the average amplitude RMS value, within one-second windows. For this investigation, the RMS value per frame has been evaluated using Equation 4 where $x(n)$ is the time domain audio signal within a given frame, and N is the number of samples in a frame.

$$\text{RMS}\sqrt{\frac{\sum_{i=0}^{N} x(i)^2}{N}} \tag{4}$$

This will produce a single RMS value, and will be used to produce an RMS value for each frame. The period of each frame has been set to 20 ms with N being directly related to the sample frequency of the signal. Since all the dataset items have been sampled at a frequency of 22.05 kHz, the nominal N value will be 441. The average RMS across each second of data will be calculated (i.e. over every 50 frames), with the percentage of frames below a threshold of 50% of the RMS mean then being determined over each second. The average percentage of each of these extracted data points will then be taken as the percentage of "low-energy" frames within the signal.

3) Spectral "Flux": Spectral "flux" seeks to measure the rate of change of the signal on a frame-to-frame basis. For the purpose of this investigation it was decided that similarly to the way flux was calculated by Scheirer at all [4], we would measure the 2-norm of the frame-to-frame spectral amplitude difference vector previously defined by Equation 1 where i refers to the particular frame in reference, and X_i is defined by Equation 5 where $x(j)$ is the time domain audio signal within a particular frame, and N is the number of samples in a frame.

$$X_i = \max(x(j)) \; j \leq 0 < N, \; j \in N \tag{5}$$

This will in effect produce a vector which measures the difference in maximum amplitude from frame to frame, and produce a single measuring the spectral flux of the signal by calculating the second norm of this vector according to the definition of the 2-norm shown in Equation 6.

$$||x|| := \sqrt{x.x} \tag{6}$$

4) MFCCs: The MFCCs will be extracted and used to train the ANN. Since the MFCCs are naturally a high-dimensional feature, they are much better suited for use with an ANN, which does not suffer from the "Curse of Dimensionality".

After plotting the MFCCs with varying numbers of coefficients, we decided to use 80 coefficients as inputs to the ANN. Using fewer coefficients causes a loss in accuracy of the final model, and increasing the number of coefficients does not increase the accuracy of the final model significantly. The 80 MFCCs for a sample speech file and music file can be seen in figure 5 below.

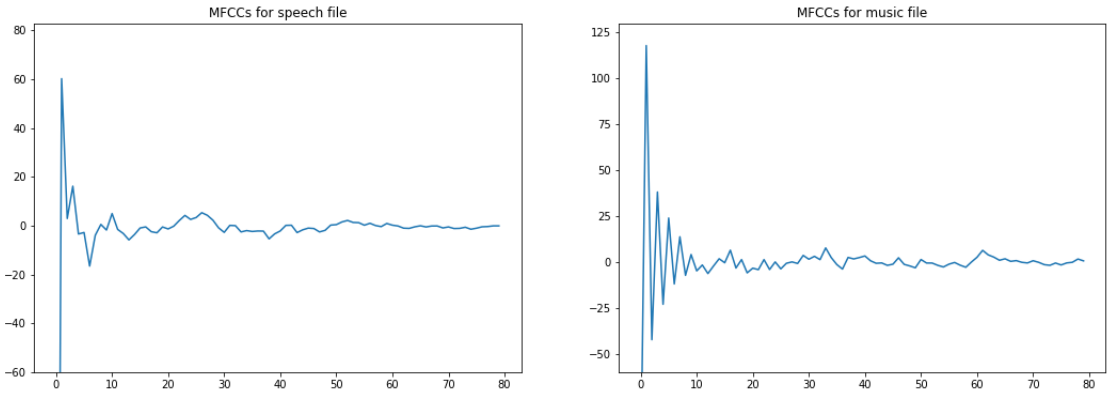

Fig. 5: Comparison of MFCCs for sample speech file (left) and music file (right)

The MFCCs can be found by applying the following steps to the original signal:

The signal $s(n)$ is first split into frames, each frame containing a certain number of samples. The Discrete Fourier Transform (DFT) is then applied to each frame.

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{-j2\pi kn/N} 1 \leq k \leq K$$

Where $s_i(n)$ is the signal at frame i, $h(n)$ is a window function, N is the number of samples in a particular frame i, And K is the length of the DFT.

The power spectrum of each frame can then be found as follows:

$$P_i(k) = \frac{1}{N}|S_i(k)|^2$$

A filterbank (an array of band-pass filters) is then applied to the power spectrum, separating the original spectrum into a number of components. In order for this to be a Mel-frequency Cepstrum, these filterbanks need to be spaced on the Mel-scale. Each frequency can be scaled as shown below:

$$M(f) = 1125ln(1 + \frac{f}{700})$$

In our case, since we are using 80 coefficients, we would need 80 components after this step.

The log of each of these components is taken, and then a Discrete Cosine Transform (DCT) is taken of the log components, leaving us with the MFCCs.

B. Data Acquisition

In order to ensure the equal evaluation and fair comparison across both classifiers, each will be trained on the same dataset. The dataset is a collection of smaller datasets and comprises a total of 8,000 labelled audio files, of which 4,000 contain speech files and the remaining 4,000 containing music files. The samples in the dataset are all of different lengths, varying from two seconds to one minute, in order to represent real life scenarios where we are not guaranteed to encounter an audio clip of a specific length. Furthermore, all these audio files have been sampled at 22.05 kHz, whether by means of upscaling or downscaling the original file, in order to ensure consistency and and more straightforward programming in creating the classifiers. While we acknowledge that this may cause information loss, especially in audio files which generally tend to be swamped at 44.1 kHz, our ultimate goal is to evaluate and compare the accuracy of the two classifiers, and are not considered about how the quality of the samples affects the accuracy.

In finding speech data, we made an effort at finding files which contained a variety of speakers, different accents and speech scenarios. Hence the final dataset of audio files contains audio clips evenly distributed between male and female speakers, people reading sections forms books, reading out numbers, along with various other scenarios. Furthermore, the hardware used to record these clips varies leading to both apparent low quality and high quality audio. The variation in the dataset allows the decision boundary learned by the classifiers to encapsulate the wide range of possible examples which could need to be classified in a real world test.

Similarly to the speech data, an effort was placed at finding a variety of music files in order to train the classifiers for a range of real world scenarios. Hence the music files contained within the final dataset represents an even distribution of music from many genres, varying rhythms and instruments, and importantly both with and without vocals.

C. Model Training

In order to ensure that the models can be trained and later validated, the dataset used for model training will be split into training and testing, and validation sets. Each set will be kept separate from one another to ensure there are no biases which may affect the results. Furthermore, several considerations were made in the training of the classifiers. These considerations amount to different algorithmic approaches that each classifier can take in order to improve overall efficiency, but often at a cost of mode performance. The considerations taken will be discussed in the following subsections, with results of the considerations being made available in subsequent sections.

1) KNN Classifier: First and foremost, we must consider one of the most limiting factors of the KNN classifier, namely the "Curse of Dimensionality" . As discussed in earlier sections, the KNN classifier may not work well with higher dimensional data and would not be well suited to using features such as the MFCCs due to their natural tendency to contain more than three dimensions. For the purposes of an apples to apples comparison, we have trained a KNN classifier using the MFCCs to showcase exactly how the KNN suffers at higher dimensions, as well as highlight the power of ANNs with regards to higher dimensions.

More importantly we have also trained the KNN exclusively using lower dimensional features for a more rigorous comparison of the capabilities of both KNN and ANN classifiers. These features being the average variance of the zero-crossing rate (ZCR), the percentage of "low-energy" frames and the spectral "flux", making for a three dimensional feature space.

Further considerations were made when training the KNN classifier such as the number of neighbors (the K value) and the weights function used in predictions. Since there is no predetermined optimal value for K, the range of values 1,5,11 and 25 were tested for both the higher and lower dimensional models to determine the highest possible level of accuracy. Two weight functions were also tested across both models in which all points in each neighborhood are weighted equally (uniform) and where weight points are determined by the inverse of their distance meaning closer neighbors of a query point will have a greater influence than neighbors which are further away (distance).

The final consideration made was the sensitivity of the KNN to the different magnitude of the features dominating the distance metric, as explained in earlier sections. Hence all features were normalised to be in the range of [0,1]. This normalisation took place independently between training and testing sets to prevent any interference and any relationship between training and testing that may bias the results.

Source code used to train the KNN classifier can be found in our Git repository [17].

2) ANN Classifier: The source code used to train the ANN classifier can be found in our Git repository [18].

The ANN classifier will have 80 input nodes for the 80 MFCCs which will be used to train it. There will then be three hidden layers, each containing 100 nodes. The choice of 100 nodes in these layers was somewhat arbitrarily chosen based on research which shows that a hidden layer size which is slightly bigger than the input layer size is a good rule of thumb. The choice of 3 hidden layers was chosen since it showed an increase in accuracy over 1 or 2 layers, while there wasn't a massive increase in training time, probably due to the use of the Adam optimizer. A single node will be used in the output layer, since this is a binary classification task.

The models need to be trained on the labeled examples. When Training the ANN, an Adam optimiser will be used to speed up the gradient descent process while still achieving a parameter configuration with a high accuracy. L2 regression will also be used to make sure none of the parameters grow too large, and cause overfitting. The ReLU activation function will be used on all hidden layer nodes, and a step function on the output node to force the output to be 0 or 1.

D. Model Evaluation

The performance of both models will be tested using K-folds cross-validation to measure accuracy. A K value of 10 was somewhat arbitrarily chosen. Most research shows that this is a good rule of thumb when running K-Folds on larger datasets, since the statistical significance of even one tenth of the data is still large. The final accuracy of the model will be calculated by taking an average of the K cross-validation accuracy predictions, and the variance of these accuracy predictions will also be calculated to give an idea of the associated error of the final accuracy.

The time taken to train the model, as well as the time taken to make a single prediction, will also be measured. Since both accuracy and speed can be large factors when deciding on a classifier, they both are important metrics when evaluating the models.

IV. RESULTS

A general trend was seen across all KNN results presented in Tables 4.1 - 4.4 whereby as the number of K values increased, there was a marginal increase in the amount of time taken to make a single prediction, while the total time spent to train the model stayed relatively constant. This is an expected result as naturally when the model needs to make predictions for higher K values, there are more calculations required to find the increased number of neighbors.

With regards to accuracy caused by weighting, Tables I and II show that applying different weighting functions to the KNN when working with lower dimensional data had a marginal impact on the accuracy of predictions. The best total accuracy achieved with uniform weighting was 7.412 ± 0.306 % with a K value of 25, while best total accuracy achieved with distance weighting was 97.400 ± 0.356 % with a K value of 11. These values are extremely close and can be considered negligible. The difference between the two seems to occur when considering the time taken to make single predictions and total training, with time taken to make a prediction using distance weighting increasing at a more rapid pace than that of the uniformly weighted counterpart. Once again this can be explained by the natural need to perform more calculations when increasing the value of K, and then further needing additional calculations for the inverse distance metrics.

Some surprisingly unexpected results are showcased in Tables III - IV where the results of the KNN classifier were not in line with the expected results. As discussed in earlier sections KNN classifiers tend to suffer from the "curse of dimensionality" and are not expected to perform well with high dimensional datasets. Tables III - IV in this study seem to show that not only does the KNN perform well with a high dimensional dataset, it is able to make even more accurate predictions that its lower dimensional counterpart, with the highest accuracy achieved being 97.850 ± 0.168 % with a K value of 1 and uniform weighting, while the lower dimensional dataset saw an accuracy of 97.412 ± 0.306 % with a K value of 25 and uniform weighting. Single prediction times significantly increases with the higher dimensional dataset, which is expected with an increased number of dimensions per datapoint, with prediction times also showing a large increase. Ultimately the higher dimensional dataset was able to produce an accuracy improvement of 0.438 % while increasing prediction times by 50 fold and increasing training times by 20 fold.

Several factors were explored to try and determine why the KNN classifier did not suffer from the pitfalls associated with higher dimensions. Several different training algorithms were tested from the sklearn library which was being used to build our KNN classifier, including the KD Tree algorithm which is known to not work well with higher dimensional datasets. None of this testing was able to result in poorly performing classifiers. Another reason explored was the size of the dataset being in line with the exponential size dataset that is required at higher datasets. This was also found to be inconsistent with expected

K	Speech Accuracy (%)	Music Accuracy (%)	Total Accuracy (%)	Single Prediction Time (ms)	Total Training Time (ms)
1	96.675 ± 3.214	96.175 ± 3.679	96.425 ± 0.379	0.0212 ± 0.000	4.69 ± 0.000
5	98.125 ± 1.840	96.600 ± 3.284	97.362 ± 0.158	0.0227 ± 0.000	4.79 ± 0.000
11	98.650 ± 1.332	95.875 ± 3.955	97.263 ± 0.198	0.0247 ± 0.000	4.99 ± 0.000
25	98.725 ± 1.259	96.100 ± 3.748	97.412 ± 0.306	0.0274 ± 0.000	4.99 ± 0.000

TABLE I: KNN Results of Lower Dimensional Dataset using Uniform Weighting

K	Speech Accuracy (%)	Music Accuracy (%)	Total Accuracy (%)	Single Prediction Time (ms)	Total Training Time (ms)
1	96.525 ± 3.354	95.775 ± 4.046	96.150 ± 0.902	0.00288 ± 0.000	4.68 ± 0.000
5	98.325 ± 1.647	96.050 ± 3.794	97.188 ± 0.295	0.00474 ± 0.000	4.58 ± 0.000
11	98.600 ± 1.380	96.200 ± 3.656	97.400 ± 0.356	0.00624 ± 0.000	5.0 ± 0.000
25	98.850 ± 1.137	95.775 ± 4.046	97.312 ± 0.276	0.0086 ± 0.000	4.88 ± 0.000

TABLE II: KNN Results of Lower Dimensional Dataset using Distance Weighting

K	Speech Accuracy (%)	Music Accuracy (%)	Total Accuracy (%)	Single Prediction Time (ms)	Total Training Time (ms)
1	99.850 ± 0.150	95.850 ± 3.978	97.850 ± 0.168	0.14 ± 0.000	91.8 ± 0.003
5	99.900 ± 0.100	94.750 ± 4.974	97.325 ± 0.413	0.188 ± 0.000	93.0 ± 0.008
11	99.850 ± 0.150	93.725 ± 5.881	96.787 ± 0.347	0.203 ± 0.000	93.0 ± 0.008
25	99.675 ± 0.324	92.475 ± 6.959	96.075 ± 0.272	0.228 ± 0.000	92.1 ± 0.007

TABLE III: KNN Results of Higher Dimensional Dataset using Uniform Weighting

K	Speech Accuracy (%)	Music Accuracy (%)	Total Accuracy (%)	Single Prediction Time (ms)	Total Training Time (ms)
1	99.875 ± 0.125	95.950 ± 3.886	97.912 ± 0.341	0.126 ± 0.000	92.0 ± 0.002
5	99.900 ± 0.100	94.650 ± 5.064	97.275 ± 0.424	0.168 ± 0.000	93.7 ± 0.034
11	99.825 ± 0.175	93.825 ± 5.794	96.825 ± 0.463	0.179 ± 0.000	92.1 ± 0.002
25	99.775 ± 0.224	92.625 ± 6.831	96.200 ± 0.738	0.214 ± 0.000	93.3 ± 0.003

TABLE IV: KNN Results of Higher Dimensional Dataset using Distance Weighting

Speech Accuracy (%)	Music Accuracy (%)	Total Accuracy (%)	Single Prediction Time (ms)	Total Training Time (s)
99.875 ± 0.125	99.725 ± 0.274	99.800 ± 0.038	0.00698 ± 0.000	3.77 ± 0.023

TABLE V: ANN Predictions and Timing Results

results as the 80 dimensional data space only had 8000 data samples from which to train, far fewer than the dataset that one expects is needed at such a high dimension.

From Table V it is clear that the ANN has a higher accuracy than any of the KNN variants, however, the difference in accuracy is surprisingly small, and in some cases would be considered negligible. The difference in training time is quite large, however, the prediction time is often faster than the KNNs prediction time.

A general trend that was seen across both ANN and KNN classifiers was the advanced levels of accuracy of classifying speech compared to comparing music. Some of the misclassified speech files were interrogated and there seemed to be several commonalities between these misclassified files, including the fact that almost all of these involved high levels of background noise from a crowd of people or background music, or a poor acoustics combined with poor recording quality resulting in a echoing of the speaker. These may have caused the misclassifications as they may show similar characteristics of music files. Similarly, some of the misclassified music files were interrogated, with the higher misclassification of music files definitely leading to a more interesting investigation. Unlike the misclassified speech files, there were no dominant similarities between the misclassified music files. Some of the misclassifications proved to be anomalies in the dataset whereby most of the audio sample consisted of speech, with small portions of music. Other music seemed to be a collection of arbitrary low frequency noises, while the vocals of some music samples were far more emphasised than the music created by instruments. Many of the misclassifications also occurred on music files void of any vocals, leading to question how these could ever be classified as speech. Some of the similarities between these music files included the use of only one or two instruments, as well as a slow and steady tempo. While the complete reasoning for music classification is not not understood, the result was expected as there is naturally more variation present in creation of a musical piece with a variety of different instruments as opposed to the natural variation associated only with the human voice.

V. Conclusion

Unsurprisingly, it was found that the ANN classifier achieved increased accuracy scores over it's KNN counterpart at the heavy cost of increased training and prediction times. However, this increased performance was only a marginal improvement over the KNN classifier, with the KNN classifier working extremely well on both lower and higher dimensional datasets. While in this specific instance the ANN classifier might have worked well on higher dimensional data, there is no evidence to back up a claim that this would be the case in future studies as all current literature points to this being unfounded.

It was ultimately found that the choice use of which classifier to use is closely related to the application for which the speech-music discriminator is to be applied. Should there be a need for a simple to implement classifier, where accuracy is not of utmost importance, a KNN classifier would be the ideal fit where a lower dimensional dataset using the features described in this report would produce excellent results. ANN classifiers on the other hand can be used in situations that require the highest levels of accuracy without too much concern for timing requirements.

References

[1] K. El-Maleh, M. Klein, G. Petrucci, and P. Kabal, "Speech/music discrimination for multimedia applications," in *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*, vol. 4, 2000, pp. 2445–2448 vol.4.

[2] J. Saunders, "Real-time discrimination of broadcast speech/music," in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 2, 1996, pp. 993–996 vol. 2.

[3] B. Kedem, "Spectral analysis and discrimination by zero-crossings," *Proceedings of the IEEE*, vol. 74, no. 11, pp. 1477–1493, 1986.

[4] E. Scheirer and M. Slaney, "Construction and evaluation of a robust multifeature speech/music discriminator," in *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, 1997, pp. 1331–1334 vol.2.

[5] C. Panagiotakis and G. Tziritas, "A speech/music discriminator based on rms and zero-crossings," *IEEE Transactions on Multimedia*, vol. 7, no. 1, pp. 155–166, 2005.

[6] M. H. B.P. Bogert and J. Tukey, "The quefrency alanysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking," in *Time Series Analysis*, 1998, ch. 15, pp. 209–243.

[7] G. K. Todor Ganchev, Nikos Fakotakis, "Comparative evaluation of various mfcc implementations on the speaker verification task," in *Proc. of the SPECOM-2005*, vol. 1, Patras, Greece., Oct. 2005.

[8] M. Muller, in *Informationg Retrieval for Music and Motion*. Springer, 2007, pp. 65–67.

[9] G. Guo, H. Wang, D. Bell, and Y. Bi, "Knn model-based approach in classification," 08 2004.

[10] H. Wang, I. Düntsch, G. Gediga, and G. Guo, "Nearest neighbours without k," Jan 1970. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-32370-8$_1$2

[11] N. Kouiroukidis and G. Evangelidis, "The effects of dimensionality curse in high dimensional knn search," in *2011 15th Panhellenic Conference on Informatics*, 2011, pp. 41–45.

[12] L. Chen, "Curse of dimensionality," Jan 1970. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9$_1$33

[13] S. U. Yoshihiko Hamamoto and S. Tomita, "On the behavior of artificial neural network classifiers in high-dimensional spaces," in *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 18, May 1996, pp. 571–573.

[14] A. A. Siddharth Sharma, Simone Sharma, "Activation functions in neural networks," in *International Journal of Engineering Applied Sciences and Technology*, vol. 4, Apr. 2020, pp. 310–316.

[15] S. Ruder, "An overview of gradient descent optimization algorithms."

[16] J. L. B. Diederik P. Kingma, "Adam: A method for stochastic optimization," in *ICLR*.

[17] M. W. Lock, "Classification-of-music-and-speach," Jun 2020. [Online]. Available: https://github.com/matthew-william-lock/Classification-of-Music-and-Speach/blob/master/training$_a$nd$_t$esting/knn.ipynb

[18] L. Godfrey, "Music-speech-classification," Jun 2020. [Online]. Available: https://github.com/Lawrence-Godfrey/Music-Speech-Classification