ASSESSING THE PERFORMANCE OF A RANGE OF ACOUSTIC FEATURES FOR SPEECH-BASED MENTAL HEALTH CLASSIFICATION USING A CONVOLUTIONAL NEURAL NETWORK

S. ALTINPINAR

3rd Year Project Final Report

Department of Electronic &
Electrical Engineering

UCL

Supervisor: Dr. Arsam Nasrollahy Shiraz

29th March 2022

I have read and understood UCL's and the Department's statements and guidelines concerning plagiarism.

I declare that all material described in this report is my own work except where explicitly and individually indicated in the text. This includes ideas described in the text, figures, and computer programs.

This report contains 29 pages (excluding this page, the contents, list of figures, and the appendices) and 9975 words.

Signed: Semih Altinpinar Date: 29/03/2022

(Student)

Contents

List of Figures:

List of Tables:

ASSESSING THE PERFORMANCE OF A RANGE OF ACOUSTIC FEATURES FOR SPEECH-BASED MENTAL HEALTH CLASSIFICATION USING A CONVOLUTIONAL NEURAL NETWORK

S. Altinpinar

Abstract

Mental health diagnosis is a tedious process which is not currently automated in any aspects, often requiring extensive psychiatric analysis to diagnose. This project aims to assess and compare acoustic features which can be extracted from speech recordings, so that the recordings can be classified using a convolutional neural network, autonomously distinguishing between subjects with mental pathologies and healthy controls. The best performing feature was the log Mel spectrogram, achieving an accuracy of 0.794, sensitivity of 0.946 and specificity of 0.642. Other features were also tested, listed in order of best performance to worst performance, Mel spectrograms, Mel frequency cepstral coefficients (MFCCs), spectral centroid, jitter, root mean square (RMS), shimmer, zero-crossing rate as well as the raw signal.

1 Introduction

1.1 Mental Health

Mental or psychological pathology involves, "the study of the causes, components, course, and consequences of psychological disorders [1]." Psychological disorders are "generally characterised by a combination of abnormal thoughts, perceptions, emotions, behaviour and relationships with others [2]" and affect over 10% of the worldwide population [3]. While most physical medical conditions can be diagnosed through methods such as physical examination, mental health disorders cannot be diagnosed through "objective clinical tests [4]." Rather, mental disorders require therapy and often extensive psychiatric analysis by a professional to reach a diagnosis, all of which can often take a considerable amount of time. In addition to this, other factors such as the stigma and discrimination around therapy as well as mental health treatment increase the time needed for a successful diagnosis. These factors contribute to findings by the Office of National Statistics showing that "36% of mental health disorders are undiagnosed [5]." Increasing the ease of assessment as well as general access to mental health assessment is key to reducing this number as well as allowing those with mental disorders to receive the treatment they need. Introducing automated aspects to this in the form of patient screening or a pre-diagnosis would revolutionise mental health care, allowing for more resources to be allocated towards treatment of those that are likely to suffer from mental health disorders.

1.2 Automated Diagnosis

Currently, no automated systems for the diagnosis or screening of patients with mental health disorders are in place within health services, all diagnosis and screening is completed via human interaction and psychiatric analysis. As this is the case, the place of automated speech-based mental health monitoring within the field of mental health care would be pre-diagnosis, as a method to screen out patients who are unlikely to show any signs of mental health disorders after more extensive analysis. Current research aims to reduce the dependence of

human interaction on the diagnosis and screening process via automated methods such as machine learning and statistical analysis. One area in which this can be achieved is through speech-based analysis. A large number of mental pathologies can be identified through various aspects, or features, of the patient's speech, with existing studies indicating that speech-based mental health classification is possible with accuracies exceeding 70 % [6] - [12]. More information regarding acoustic features as well basic machine learning theory is covered within the background theory section.

1.3 Literature Review

Classifying speech between patients with forms of mental disorders and those without has already been investigated to a significant extent regarding the feasibility of the concept as well as the accuracy of various methods. A recent review [6], analysed 127 different studies, utilising various automated methods, finding that the majority of automated systems (63%) use machine learning predictive models for classification as opposed to null-hypothesis testing; a simpler statistical method. [6] also details the use of a very large range of acoustic features for classification across the 127 different studies within the review, such as Mel frequency cepstral coefficients (MFCCs), Mel spectrograms, jitter, and shimmer. Machine learning predictive models which utilise many of the features mentioned in [6], are found throughout the related literature, providing an insight into the features that are most used for classification within the field. Furthermore, there is also basic information indicating which acoustic features correlate to given pathologies, however this is not expanded upon greatly. Despite containing detail regarding a large range of features, [6] is only a review of related literature and therefore lacks specific explanation of the methods used such as the architecture of the machine learning platforms used and the resulting accuracies using individual features.

Similarly, to [6], [7] utilises acoustic features such as Mel spectrograms and MFCCs combined with the use of a machine learning predictive model to classify those with mental disorders from healthy controls with an accuracy above 70%. This paper introduces the use of convolutional neural networks (CNNs), which are a form of machine learning algorithm and suggests that they are well suited for classification using these features. This paper explains in detail the architecture and design of the CNN which may be used as a base for this work, however, only uses two spectral features, in this case MFCCs and log Mel spectrograms. Therefore, unlike [6], this paper does not include an extensive range of features. [7] also lacks detailed comparison between the results of each feature in classification, simply stating, "models trained on log Mel spectrograms consistently outperform those of MFCCs [7]." While the comparison of these results is useful, there is no explanation of why each feature performs this way. Furthermore, the code used for feature extraction and to implement the CNN is not accessible which severely limits the reproducibility of this work.

Both MFCCs and spectrograms are also utilised in [8], achieving a maximum of 77% classification accuracy on the test set of data, using a CNN much like [7]. The repeated use of a CNN in both examples further illustrates that this form of machine learning algorithm can be used with these features, resulting in high classification accuracies. Information regarding feature extraction is provided as well as a large amount of detail regarding the exact configuration and design of the CNN classifier, indicating some appropriate parameters for the CNN design. This work also describes the use of the Saarbreucken Voice Database (SVD) [13] as a source of a high number of speech recordings from different pathologies

which will be used for this project as there are few publicly available databases containing such data. [8] trained and tested the CNN using 482 pathological samples and the same number of control samples, with each recording being formed of a sustained vowel sound, meaning there was a large amount of data for training and testing. Much like some of the previous literature, [8] determines the accuracy of the features in only one test and does not explore the accuracy of individual features. [8] also does not assess the properties of the acoustic features which may correlates with their high performance in classifying mental pathology, leaving room for development. Furthermore, there is no provided code for neither the feature extraction nor the CNN classifier which, once again, limits the reproducibility and development of this work.

The work performed in [9] compares the use of two different machine learning algorithms for classifying healthy versus pathological subjects finding that CNNs have a higher classification accuracy of 87.11% . MFCCs are used alongside other features such as zero-crossing rate, spectral centroid, energy, and spectral flux are used in combination with a single test. The CNN was tested against another form of machine learning algorithm, a recurrent neural network (RNN) which resulted in a classification accuracy of 86.55% in this case. [9] provides the architecture and design of both the CNN and RNN with high levels of detail as well as moderately detailed discussions on why the accuracy of the classifier is difficult to improve, describing the neural networks as, “black boxes [9].” Despite this analysis, there is a distinct lack of explanation and analysis of the individual features used, with only a basic explanation of MFCCs and pitch being given as well as no comparison between features due to their combined use leaving room for further development and analysis. Like [7] and [8], the code used to implement the feature extraction as well as the classifiers is not provided, once again limiting the reproducibility and future development of the work.

Focusing on classifying a single mental disorder, in this case, schizophrenia with a high classification accuracy of 79.4%, [10] utilises simpler statistical methods like those mentioned in [6]. Simpler features such as pitch, energy, and total length of pauses are combined while achieving a high accuracy compared to the other literature [7], [8]. This allows for development as much of the previous literature focuses on more complex spectral features such as MFCCs and Mel spectrograms to achieve such a good classification accuracy, meaning that exploring the use of simpler temporal features will provide value and insight into the relationship between speech and pathology. [10] also includes detailed explanation of how the features are computed and calculated, giving some very brief explanations of how they are related to the human voice and some pathologies. Despite these positive aspects, the sample size is comparatively small with 57 total speech recordings, with 39 pathological and 18 healthy controls of which were obtained purely for the purposes of this study and are not publicly available. Furthermore, the code used for the statistical analysis as well as feature extraction is not given, making the results severely unreproducible.

Much like [10], a statistical approach is utilised in [11] rather than a machine learning to analyse patients with a given condition, in this case, tremor. [11] utilises shimmer and jitter, two features which are not very common within the related literature despite their known relationship with speech quality, as found in many studies investigating vocal pathology such as [14]. Despite the use of these features, the process of feature extraction was not explained in detail. This study also does not explicitly perform any means of classification, but rather

aims to find direct relationships between the acoustic features and pathology, of which, “no significant differences were found for any parameter [11].” Despite the apparent conclusive nature of these findings, a very small sample size of 49 recordings was used in this study which is less than much of the literature and may be indicative of why no relationships were found.

Following [7] - [9], the work performed in [12] utilises MFCCs as the main feature for classification along with a CNN classifier, achieving an accuracy of 77.5%. The high accuracy of this implementation further reinforces the use of CNNs with MFCCs and similar spectral features for classification. A large number of samples are also used within this study, 259 healthy and 259 pathological, once again from the Saarbreucken Voice Database as in [8]. The CNN architecture is explained well with detailed information on the design along with brief explanations of the relationships between vocal symptoms and mental pathology, which is often beyond the scope of the previously mentioned literature. Despite these positive aspects, [12] uses no other features for classification and has no explanation of what MFCCs represent or why they are used for the purpose of speech-based classification, which does somewhat hinder the understanding of the results and conclusions. Furthermore, there is no code provided for the implementation of the CNN within the methods as well as features extraction, signifying that the results are not reproducible as with many of the studies in this field.

In summary, the literature clearly shows that machine learning predictive models are the most common method in speech-based mental pathology classification, achieving accuracies of above 80% in some cases [9]. The same process for feature extraction is followed by the majority of the literature with spectral features such as MFCCs and spectrograms passed to a classifier. It is also apparent that the most used form of machine learning algorithm is the convolutional neural network, as found in [7], [8], [9], [12], due to its design which will be discussed further in the background theory of this report. The aforementioned literature often fails to include multiple acoustic features within classification, while sometimes combining multiple features into one set of data to be passed to the classifier which may increase the accuracy yet fails to allow one to understand how each feature performs separately and the reasons for this. Therefore, future research and work, should aim to assess the use of a range of acoustic features individually, feeding them into a machine learning classifier as well as dissecting each feature to understand what the features represent, with regards to speech as well as sound in general.

Specifically, this project aims to investigate a range of acoustic features in their ability to classify speech samples pathological and healthy, using a CNN classifier and assessing the reasons behind the performance of each feature. Obtaining an intuitive understanding of what each feature represents is also of large importance to assess each feature’s performance. The process will be documented fully to allow reproducibility and development in future work, aiding automation to become a major addition to the field of mental health diagnosis.

2 Goals, Objectives, and Tasks

The main goal of this project is to assess the performance of a range of acoustic features when used to classify pathological and healthy speech recordings. In other words, how accurately a machine learning algorithm, in this case a convolutional neural network, can distinguish between subjects with mental pathology versus those without, depending on

which extracted acoustic features are being used to train and test the classifier. The focus is on assessing the accuracies of the classifier using each feature individually and not the classifier itself; the subject of interest is why certain features outperform other features and not the design of the classifier. The machine learning algorithm used will have its parameters and design fixed as machine learning is not the focus of this project and will aid in producing comparable results between the features.

To achieve this final goal, the acoustic features which are to be extracted and used for testing must first be identified using the related literature. Once the features that are to be used have been identified, they can then be extracted and explored using a single speech recording as an example, allowing for an in depth understanding of what they represent and possible relationships to pathology. After which, a database of samples of speech recordings must be obtained and the each of the features must be extracted from all the recordings to provide the classifier with data in a format that it can use. The literature must also be reviewed in order to understand which forms of machine learning are used for this purpose and which forms provide the best accuracy when classifying using a given set of features. This machine learning algorithm must then be trained and tested using each feature individually in order to obtain a set of results indicating the classification accuracy for each feature. The resulting accuracies of the classifier using each feature can then be assessed and compared to each other in order to investigate which features achieve the best classification and the reasons behind this. All methods are to be documented in a comprehensive manner to allow for reproducibility and provide a standard that similar work should be carried out to.

3 Background Theory

In order to begin to classify speech samples of subjects with forms of mental pathology and without, one must first understand the acoustic features and how they may correlate with forms of mental pathology. These features can be extracted from speech recordings and be used for classification purposes. In this assessment, the focus is on Mel-frequency cepstral coefficients (MFCCs), Mel and log Mel spectrograms, spectral centroid, root mean square (RMS), zero crossing rate (ZCR), jitter and shimmer. These features will be individually dissected in order to understand how they are calculated and what they represent prior to being extracted from the speech samples. As part of the process, the data from the extracted features is passed to a machine learning classifier which is used for training and testing purposes, allowing each feature's accuracy to be determined. As machine learning will be used, it is necessary to obtain a basic understanding of machine learning and convolutional neural networks (CNNs), which are to be used for this work. The basic theory behind machine learning and CNNs will be sufficient to understand the results of this project as the focus of this work is to understand the relevance of each feature and its associated performance for classification. As a result of this, the CNN used will take large inspiration from the literature and other sources, with modifications being made to allow it to be used for this purpose.

3.1 Acoustic Features

Speech signals and some of acoustic features within it often strongly correlate with forms of pathology, both physical and mental. An acoustic feature is the name given to various properties of sound signals and each capture different aspects of the sound, coming under three main categories: time domain, frequency domain and time-frequency domain. The process of feature extraction is similar for all features, starting with loading the audio signal

as a sequence of samples given by the sample rate, sr and then segmenting the audio into discrete frames containing a given number of samples which is denoted as the frame size, K, essentially diving a large signal into smaller segments which can be analysed individually. After framing the signal, the data in each frame can be transformed and manipulated in various ways to give the respective features in their respective domain. To understand the results after the features have been passed to a machine learning classifier, it is important to first understand what each feature represents as well as its domain so that one can determine exactly what information the classifier will be using.

3.1.1 Time Domain Features

Time domain features analyse signal waveforms of sound in its raw state as a function of time and amplitude where the amplitude represents the volume or loudness of the sound. Different methods of manipulating the amplitude over time provide different information about the signal with various uses. An example of a commonly used time domain feature in signal processing is the root mean square or RMS energy which is calculated as shown in (1) where $s(k)$ refers to the amplitude at the k^{th} sample, K is the frame size and t represents the frame number.

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

Much like the raw signal, RMS energy represents the loudness of the audio over time, but by utilising the root mean square, a form of average volume is represented rather than an exact depiction of the volume. This is more robust against outliers as well as sharp increases in volume which may be due to disturbances such as noise. Various volume levels can correlate with mental disorders where a weak or low volume "may represent shyness (normal), low self-esteem (depression), dysarthria [15]," and therefore may produce good results when used with a machine learning classifier.

Another time domain feature which can be used for as a feature is the zero-crossing rate (ZCR), found in the list of features from in [6]. The ZCR describes the "rate at which a signal's amplitude changes sign within each frame [16]." The definition, in the form of an equation is shown in (2) and utilises the signum function (sgn) which is used to extract the sign of a real number.

$$ZCR_t = \frac{1}{2} \cdot \sum_{k=t\cdot K}^{(t+1)\cdot(K-1)} \left|\mathrm{sgn}\big(s(k)\big) - \mathrm{sgn}\big(s(k+1)\big)\right| \tag{2}$$

Within the literature, this feature is sometimes passed to classifiers, both using machine learning and simpler statistical techniques and is used as "it is known to reflect, in a rather coarse manner, the spectral characteristics of a signal [17]," thus representing more information than time domain features typically do, having potential in its classification performance. ZCR also has other applications, such as being used before feature extraction to detect voiced and unvoiced sections of audio with voiced sections typically having a lower ZCR than unvoiced sections [18].

Shimmer is another time domain feature used within automated speech based mental health classification and is defined as "the average absolute difference between the amplitudes of consecutive periods, divided by the average amplitude [19]," thus representing the variation in volume across the length of the signal. Local shimmer is calculated as shown in (3) and is given as a percentage, where N represents the number of periods and A_i represents the amplitude of the signal at period i.

$$Shim(\%) = \frac{\frac{1}{N-1}\sum_{i=1}^{N-1}|A_i - A_{i-1}| * 100}{\frac{1}{N}\sum_{i=0}^{N-1} A_i} \tag{3}$$

As mental health disorders cause symptoms related to speech and volume control when speaking, shimmer is a valuable feature and can reveal a lot of information about the speaker. Furthermore, as mentioned in [14], shimmer is a key feature in grading speech quality and for these reasons, may show strong results during classification.

A similar feature to shimmer that is also in the time domain is jitter and is defined as "the average absolute difference between consecutive periods, divided by the average period, in percentage [19]," thus representing the difference between time periods rather than amplitudes. Local jitter is calculated as shown in (4), where *N* and i are the same as previously and T_i represents the length of a period at period i.

$$jitt\ (\%) = \frac{\frac{1}{N-1}\sum_{i=1}^{N-1}|T_i - T_{i-1}| * 100}{\frac{1}{N}\sum_{i=0}^{N-1} T_i} \tag{4}$$

Like shimmer, jitter has a direct link to the human voice as it is used as a measure of voice quality as well as having a direct links to glottal periods [20], with the glottis being defined as "the opening at the upper part of the larynx, between the vocal cords [21]," contributing to the formation of sounds, and hence, speech. This results in jitter also being a "good indicator of the presence of pathologies in the larynx such as vocal fold nodules… [22]"

3.1.2 Frequency Domain Features

Frequency domain features represent the frequency spectrum of a signal, displaying "how much of the signal is present among each given frequency band [23]." The most common way to transform a time domain signal into the frequency domain is through the Fourier Transform, providing a spectrum of the frequencies present in the signal. The calculation for this transform is given in (5) and will not be derived in this case, although the derivation is performed in [24] where $X(\omega)$ represents the signal as a function of frequency and $x(t)$ represents the signal as a function of time.

$$X(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}dt \tag{5}$$

While the frequency spectrum does have various applications, the Fourier Transform results in a spectrum providing only a single 'picture' of the frequency components within a signal as it considers the entire signal and thus cannot represents frequencies over time. This means that the Fourier spectrum of a signal has no time element within in which can limit its

usefulness when considering parts of the signal individually, thus highlighting the usefulness of time-frequency domain features, especially for the machine learning applications.

3.1.3 Time – Frequency Domain Features

Time-frequency domain features represent changes in frequencies and are presented in the form of spectrograms. Spectrograms "can be defined as an intensity plot (usually on a log scale, such as dB) of the Short-Time Fourier Transform (STFT) magnitude [25]." The STFT of a signal applies a similar concept to the regular Fourier Transform mentioned previously, except it performs the transform of each frame of a signal, showing the variation of frequencies over time. A detailed derivation as well as formulae are provided in [26] An example of a spectrogram is shown in Fig. 1. As a part of this process, when the signal has been segmented into frames, there are often discontinuities between the signals at the beginning and ends of the frames which is a result of the signal not being formed of an integer number of periods and thus can cause the "smearing of power across a frequency spectrum [27]," adding frequency components which are not present in the original spectrum; a concept known as spectral leakage. This can be mitigated using a windowing function such as the Hanning window as defined in (6). An example of the graphical representation of this is also shown in Fig. 2 Graph of Hanning windowing function generated using MATLAB

$$w(k) = 0.5 \cdot \left(1 - \cos\left(\frac{2\pi k}{K-1}\right)\right), \qquad k = 1 \dots K \tag{6}$$

The purpose of this function is to set the beginning and end values of the signal within a frame to zero and thus removing the discontinuities between frames. Thereafter, an overlap between frames must now be applies so that there are no parts of the signal that are set to zero unnecessarily. This is usually performed as one of the steps of the STFT to obtain a linearly-scaled spectrogram, with the $hop\ size$ denoting the number of samples within the overlap.

Spectrograms in this linearly scaled form are of limited use as the frequency components within human speech are not scaled linearly, but rather, logarithmically. Humans can perceive differences in frequencies better at low frequencies compared to higher frequencies [28] resulting in a different scale being used for frequency. Converting the spectrogram to a "scale of subjectively equal pitch distances [29]," in the form of the Mel spectrogram accommodates for this and better represents the frequency components in speech. Mel spectrograms are therefore an acoustic feature which provide more useful frequency information than that of a linearly scaled spectrogram and are a very common feature to be used for classification of audio. The Mel to Hertz frequency conversion being shown in (7), where m is the frequency in Mels and f is the frequency in Hertz. The power scale on these spectrograms is often also converted into dB resulting in a log Mel spectrogram, which is another useful feature to be used during classification.

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{7}$$

The penultimate feature to be explored, is the spectral centroid of a signal which is calculated by finding the average frequency present within a frame, weighted by amplitudes and dividing this by the sum of the amplitudes, the equation for which is shown in (8), where $f(n)$ represented the centre frequency of the n^{th} frequency bin and $x(n)$ represents the weighted frequency value of the n^{th} bin.

$$Centroid = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1}(x)n} \quad (8)$$

The spectral centroid is often described as a measure of the "centre of 'gravity' of the spectrum" as well as the brightness of the sounds [17]. This feature is useful as mental health disorders often exhibit themselves in the tonality of the patient's speech with lower and quieter tones being associated with conditions such as depression [15], indicating that the brightness of the sound being produced throughout the speech signal is of great importance and a valuable feature when classifying patients with mental pathologies, despite its lack of use in the literature.

The most common and final feature explored within this project are Mel frequency cepstral coefficients or MFCCs, which are calculated using the same Mel scale used for Mel spectrograms, described in (7). MFCCs are calculated by first applying the STFT to the signal after framing, resulting in a power spectrum, which must be converted to a log-amplitude spectrum, then allowing a series of "triangular-shaped windows, which are centred linearly in the Mel scale [30]," to be applied, forming filter bank energies. A visual representation of Mel filter banks is shown in Fig. 3 Example of Mel filter banks utilising 20 Mel filters. Adapted from. To calculate the cepstral coefficients from this, another transform must be applied, being the discrete cosine transform, as shown in (9), where "C_m and E_m represent the m^{th} cepstral coefficient and the k^{th} log-energy, respectively. N is the number of filters in the filter banks [30]," with the cepstrum numbers being taken from 1 to M.

$$C_m = \sum_{k=1}^{N} E_k \cos\left[\frac{m(k-0.5)k\pi}{N}\right] \quad (9)$$

"Traditional MFCC systems use only 8-13 cepstral coefficients. The zeroth coefficient is often excluded since it represents the average log-energy of the input signal, which only carries little speaker-specific information [31]." MFCCs are often used for speech recognition as well as audio classification, particular with speech as the MFCCs give a good representation of the activity within the vocal folds and glottal periods of the speaker and therefore correlate well with certain patterns of speech displayed as symptoms of mental health disorders.

A summary of all the features, including their definitions and equations can be found in Tab. 1.

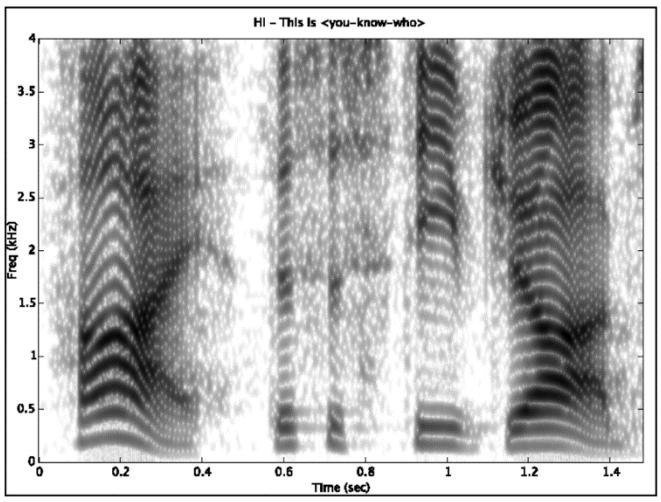

Fig. 1 Example of spectrogram with the most dominant frequencies at 0.2 and 1.3 seconds. Adapted from [25]

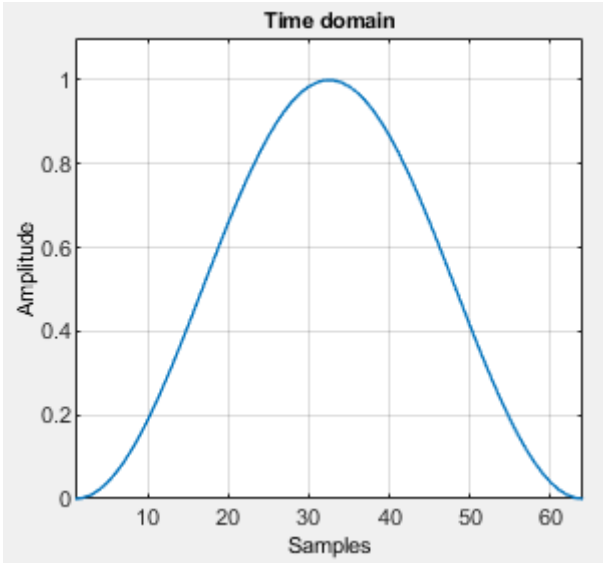

Fig. 2 Graph of Hanning windowing function generated using MATLAB

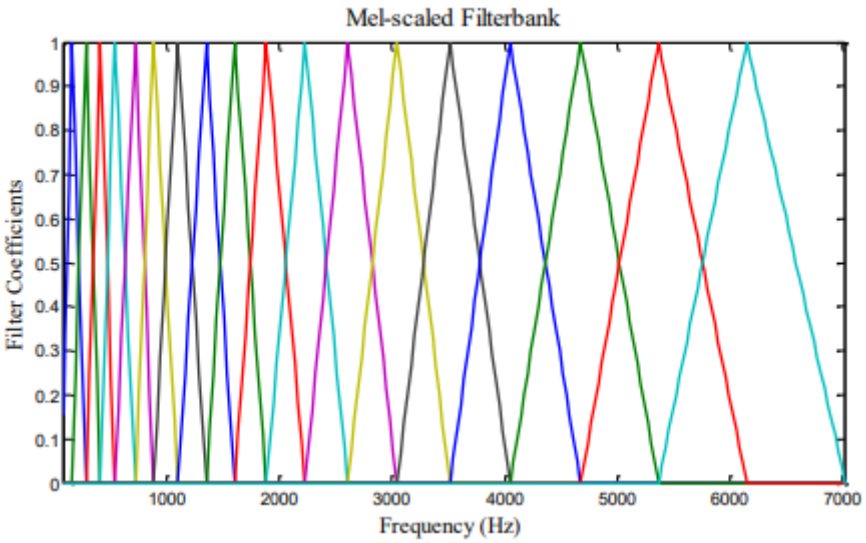

Fig. 3 Example of Mel filter banks utilising 20 Mel filters. Adapted from [30]

Acoustic Feature	Exact Definition	Simple Feature Description	How to Obtain/ Formulae
Mel-frequency Cepstral Coefficients (MFCCs)	Coefficients from Mel-spectrum of log magnitude of a segment of audio.	Essentially the short-term power spectrum of the sound using a Mel scale.	• Fourier transform of windowed segments of audio $$X(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}\ dt$$ • Convert to Mel Scale $$m\ =\ 2595\log_{10}(1\ +\frac{f}{700})$$ • Apply Mel filter-banks • Take discrete cosine transform of Mel log powers $$C_m = \sum_{k=1}^{N} E_k \cos\left[\frac{m(k-0.5)k\pi}{N}\right]$$ • MFCCs are the remaining amplitudes
Mel/ Log Mel Spectrogram	A visual representation of the spectrum of frequencies of a signal as it varies with time, converted to the Mel Scale.	Image showing the most dominant frequencies in a signal over time – essentially a 'power map'	Perform short time Fourier transform and convert frequency to Mel scale using: $m\ = 2595\log_{10}(1\ +\frac{f}{700})$
Spectral Centroid	Average frequency of a frame, weighted by amplitudes, divided by the sum of the amplitudes.	'Centre of gravity' of magnitude spectrum – frequency band where most of the energy within the signal is concentrated	$$Centroid = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1}(x)n}$$ $x(n)$ represents weighted frequency value of frequency bin, n and $f(n)$ represents the centre frequency of that bin.
Root Mean Square (RMS)	Square root of the mean value of the squared values of a quantity taken over an interval.	Definition is self-explanatory, more robust to noise than raw signal.	$$RMS_t = \sqrt{\frac{1}{K}\sum_{k=tK}^{(t+1)(K-1)} s(k)^2}$$ $s(k)$ represents the amplitude of the k^{th} sample, K is the frame size, t is the frame number.
Zero Crossing Rates (ZCR)	Rate which the signal changes from positive to negative or vice versa	Number of times the sign of the signal changes divided by the length of the segment or frame.	$$ZCR_t = \frac{1}{2}\sum_{k=tK}^{(t+1)(K-1)} \left\lvert sgn\big(s(k)\big) - sgn(s(k+1))\right\rvert$$
Jitter	Deviations in individual consecutive fundamental frequency period lengths	Average difference in length of consecutive periods of audio, divided by average period. Given in by a percentage	$$jitt\ (\%) = \frac{\frac{1}{N-1}\sum_{i=1}^{N-1}\lvert T_i - T_{i-1}\rvert * 100}{\frac{1}{N}\sum_{i=0}^{N-1} T_i}$$ T_i represents period length of period i, N is the number of periods
Shimmer	Difference in the peak amplitudes of consecutive fundamental frequency periods	Average difference in peak amplitudes of specific, consecutive time periods.	$$Shim(\%) = \frac{\frac{1}{N-1}\sum_{i=1}^{N-1}\lvert A_i - A_{i-1}\rvert * 100}{\frac{1}{N}\sum_{i=0}^{N-1} A_i}$$ A_i represents the amplitude of period, i

Tab. 1 Summary of features from background theory

3.2 Machine Learning Theory

This section will provide a brief overview of the theory of machine learning as well as CNNs. As machine learning is not the focus of the work performed in this project, the theory will only cover the knowledge required for a sufficient understanding of the machine learning used.

3.2.1 Machine Learning

Machine learning is the primary term used to describe a subject of artificial intelligence (AI) which "can be broadly defined as computational methods using experience to improve performance or make accurate predictions [32]." Machine learning has a wide range of practical applications regarding classification of items, such as computer vision which involves object recognition and identification, speech recognition and processing as well as document classification, to mention a few. There are also other uses for machine learning, besides classification of data such as regression calculations i.e., predicting real values for data such as, "predicting stock values or that of variations of economic variables [32]," as well as performing ranking tasks, requiring the system "learning to order items according to some criterion…," such as "returning web pages relevant to a search query [32]." Supervised learning is the most common learning scenario for machine learning algorithms which is the process of training the algorithm using a dataset with "values or categories assigned to each example [32]," known as labels. In this application, there are two classification groups and thus the labels two labels, representing recordings pathological and healthy subjects.

When using supervised learning, large datasets must be used and are typically split into three subsets, the training, validation, and test set, each with a different purpose within the learning process. These subsets are formed from randomly chosen samples, where the validation set is a subset of the training set and the size of each set is defined by the user. The training set being the largest, followed by the validation and test set.

The purpose of the training set is for supervised learning of the algorithm and is the only set in which the algorithm has access to the labels. The data and the associated labels are used by the algorithm to detect patterns within the data and learn the relationships between the features and the labels.

The validation set is used to "tune the parameters of a learning algorithm…," and to "select appropriate parameters for the learning algorithm [32]," meaning that the algorithm uses the information that it has learned from the training set to test itself on the validation set to assess how effective the learning has been. In an ideal scenario, the validation accuracy will be very high, close to 100% as the validation set is a subset of the training set, so the algorithm has already been exposed to that data.

Lastly, the testing set is formed of unseen data with the labels being hidden from the algorithm during testing. The algorithm will use the previously detected patterns and information learned to predict the labels from each sample within the test set, which is compared to the true label, providing an accuracy statistic, and is used to assess the effectiveness of the algorithm's learning. More information regarding the general theory of machine learning can be found in [32].

3.2.2 Convolutional Neural Networks

A CNN is form of neural network, a subset of machine learning, and is formed from a series of artificial neurons, defined as "a function that takes and input and produces and output [33]." These neurons are combined in different layers and ways to form various kinds of networks, the most common of which is known as a feed-forward network, meaning that "the neurons in each layer feed their output forward to the next layer until we get a final output [33]," an example of which is shown in Fig. 4. While regular neural networks take input data in the form of "simple numeric structure with no spatial structure [33]," they cannot compute multidimensional arrays, which includes data in the form of images. As spectrograms as well as MFCCs and other features are in the form of multidimensional arrays, neural networks, such as CNNs, which can accommodate these are often used for the process of speech processing and classification. Further information regarding neural networks as well as CNNs can be found in [33]. Tab. 2, displays some of the parameters which need to be considered when designing a CNN as well as their definitions. These can be modified to suit various data types to change the way the CNN learns and thus the accuracy of the classification it performs.

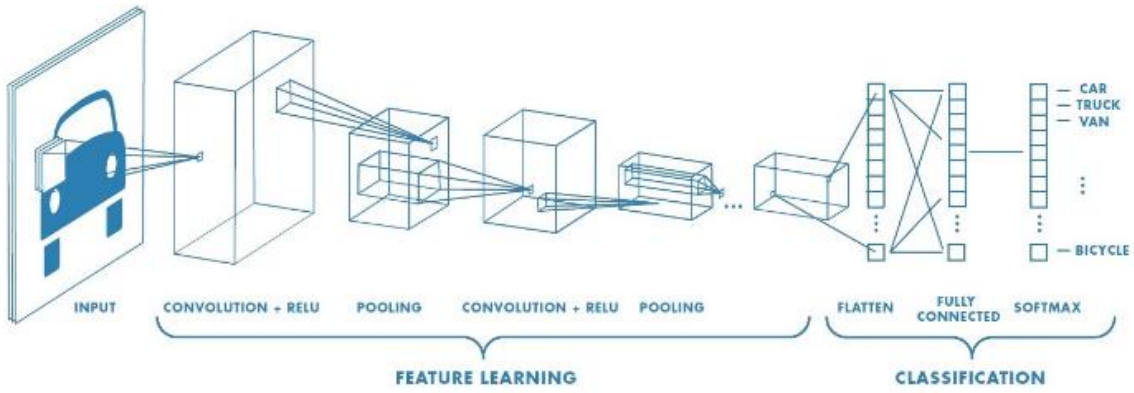

Fig. 4 Example of CNN architecture. Adapted from [34]

Parameters	Definition
Kernel Size	The size of the filter
Stride	Rate at which the kernel passes over the input
Hidden Layers	Layers between the input and output layers
Activation Functions	Allow the model to learn non-linear prediction boundaries
Learning Rate	Regulates the update of the weight at the end of each batch
Epochs	The number of iterations of the entire training dataset to the network during training
Batch Size	The number of patterns shown to the network before weights of the neurons are updated

Tab. 2 CNN parameters and definitions, redrawn from [35] with changes.

4 Methods

As shown within the literature review, existing works and studies were reviewed in order to identify which features are most commonly extracted from audio files in order to be used to classify patients with mental pathology against healthy control subjects. The features decided upon to be compared within this project are RMS, ZCR, shimmer, jitter, spectral centroid, Mel spectrograms, log Mel spectrograms and MFCCs, which are all defined and explained within the background theory section of this report. These features were chosen as they are often extracted in other work and cover both the time domain and time-frequency domain. These features also each convey different information regarding audio signals, and thus represent different information in relation to pathology and the human voice. Investigating the use of each feature individually as well as comparing them to each other will reveal important relationships between the features and mental pathology, allows further development of speech-based mental health monitoring systems in the future.

Prior to feature extraction, the speech recordings from which the features are to be extracted from must be obtained and the chosen source of this is the Saarbruecken Voice Database (SVD) [13]. This database was used within similar work such as [8] and [12] and contains over 2000 recordings from a large range of pathologies, providing large amounts of data for the CNN classifier to learn from. The recordings chosen for this work are those of a full sentence, in German, including the phrase, “Good morning, how are you?” with 629 pathological recordings and 252 healthy recordings being available. Other recordings containing various vowel sounds from the patients were also available, however, the recordings of full sentences were chosen instead as automated speech-based mental health classification in a real-world application would be more likely to utilise speech recordings from everyday life, such as recordings from interviews or everyday speech, rather than vowel sounds recorded for the specific purpose of monitoring. All code used for the following processes is provided in the appendix and was written using Microsoft Visual Studio Code, running on a Windows 10 computer with an AMD Ryzen 5 3600 processor, an Nvidia GTX 1070 GPU and 16GB RAM.

4.1.1 Initial Feature Extraction

Prior to batch feature extraction, some of the features were extracted from an individual recording in order to understand the format in which the features will be extracted in, as well as obtain a visual representation of the features. For all feature extraction as well as implementation of the CNN, the Python programming language and its related packages were utilised due to the extensive documentation regarding its use as well as its popular use within related literature and works. The following additional Python packages were installed: IPython, matplotlib, numpy, librosa, sklearn, praat-parselmouth and tensorflow. These packages deal with data processing, plot generation, feature extraction and building machine learning models. Librosa was used to extract various features from ‘4-phrase.wav,’ one of the pathological voice recordings, using a frame size of 1024, hop length of 256 as well as the default librosa sampling rate of 22050 Hz, using the respective commands within librosa. The raw audio waveform, ZCR and RMS are shown within the results. As a frame of reference and to aid in the understanding of the other features, the Fourier Transform was also applied to the signal in the form of the fast Fourier Transform, converting the time domain signal to the frequency domain, thus resulting in a frequency spectrum. Librosa was also used to perform the STFT of the audio signal to produce the linear spectrogram, which is part of the

time frequency domain, using the same frame size and hop size as previously. The Hanning window was also applied to this function by default in librosa as with the other time-frequency domain features, in order to reduce spectral leakage. The linear spectrogram was extracted in order to be compared visually with the Mel spectrogram and log Mel spectrogram, which were extracted in a similar way using librosa to convert to a Mel scale. Finally, 13 MFCCs were extracted in a similar way to the other features. All graphs from this feature extraction are shown within the results section of this report.

4.1.2 Batch Feature Extraction

To extract features from more than one file, Python code was written in order to process each file, load the audio, extract the feature and write the subsequent data to a JSON file so that it can be processed by the CNN for training. Before this can be achieved, the audio recordings were split up into 1 second sections using MATLAB in order to provide the CNN classifier with more files so that it has more data to learn from as well as ensure that all audio clips were of the same length as this makes the processing simpler. This resulted in there now being a total of 333 healthy recordings and 1,123 pathological recordings.

Using [36] and the previous feature extraction code as framework, Python code was written for each feature to extract them and write the data to a JSON file, as well as the label denoting whether the data was extracted from a healthy or pathological sample, so that it can be read by the CNN at a later stage. These features include, ZCR, RMS, shimmer, jitter, spectral centroid, Mel spectrogram, log Mel spectrogram and MFCCs as well as the raw signal as a control. These features were extracted using separate code files, which follow the same format with minor changes as shown in the appendix. All features were extracted using the same parameters with a frame size of 2048, hop size of 512 and Hanning window where applicable, as these were values which were seen within the literature and are within the range of standard parameters used. The full list of parameters used for each librosa command for feature extraction is shown in Tab. 3.

Feature	**Extraction Parameters**
MFCC	Sample Rate – 22050 Frame Size – 2048 Hop Length – 512 Number of MFCCs – 13
Mel Spectrogram Log Mel Spectrogram Spectral Centroid	Sample Rate – 22050 Frame Size – 2048 Hop Length – 512
Jitter Shimmer	Sample Rate - 22050 Period Floor – 0.00001 Period Ceiling – 0.02 Maximum Period Factor – 1.3
Root Mean Square (RMS) Zero Crossing Rate (ZCR)	Sample Rate - 22050 Hop Length - 512

Tab. 3 Feature parameters for Python commands

4.1.3 CNN Implementation and Testing

Following feature extraction, the CNN can be implemented so that it can be trained and tested using the extracted data. Using a design and inspiration from [37], Python code was written in order to work with the previously extracted features, as well as calculate the mean accuracy across 10 tests with the sensitivity and specificity of classification. This was implemented in Python utilising the sklearn and tensorflow packages. The architecture as well as parameters for the CNN are displayed in Fig. 5, with more information on the exact function of each parameter provided in Tab. 2 as well as [33]. The CNN consists of three convolution layers; each followed by a maximum pooling layer and utilises the 'Adam' optimiser with a learning rate of 0.0001, batch size of 100 and 30 epochs which provided the greatest classification accuracy after some very basic testing.

Within the code, the data file is loaded and inputted into numpy arrays, after which it is split into training, validation, and testing sets. Two separate tests were done for each feature where the first test had a split of 75-25% training to testing split with 20% of the training set also being used for the validation set, a setup used in [9]. As there was a larger number of pathological voice recordings than healthy recordings within the dataset, the second test was proportioned in a way that the test set was formed of an equal number of healthy and pathological samples so that the results would be easier to interpret and to highlight the misleading nature of the results with an unbalanced test set. This resulted in approximately a 90-10% train-test split with 20% of the training set being used for validation once again. Each feature was tested 10 times for each arrangement of the test set, recording the mean accuracy, sensitivity, and sensitivity each time which are documented in the results section.

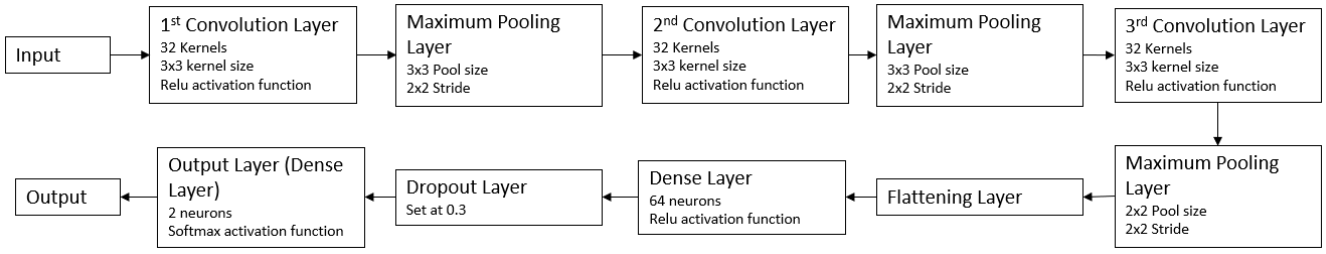

Fig. 5 Implemented CNN architecture

5 Results

5.1.1 Initial Feature Extraction Results

The resulting figures from the extraction of the raw audio signal, ZCR, RMS, frequency spectrum, linear spectrogram, Mel spectrogram, spectral centroid and MFCCs are shown in Fig. 6-11. The plots in Fig. 6 show the time domain features including the raw signal, the ZCR and RMS and it is evident that they show limited information about the signal. The ZCR plot shown in Fig. 6b shows little information as the waveform is relatively flat with only small fluctuations, and it is also clear that feature is susceptible to noise as there is a sharp peak later in the signal where there is little speech occurring, as seen from the raw waveform. The RMS plot in Fig. 6c follows the raw waveform rather closely as it is essentially just a representation of the average volume and does not provide much additional information than the raw plot from Fig. 6a.

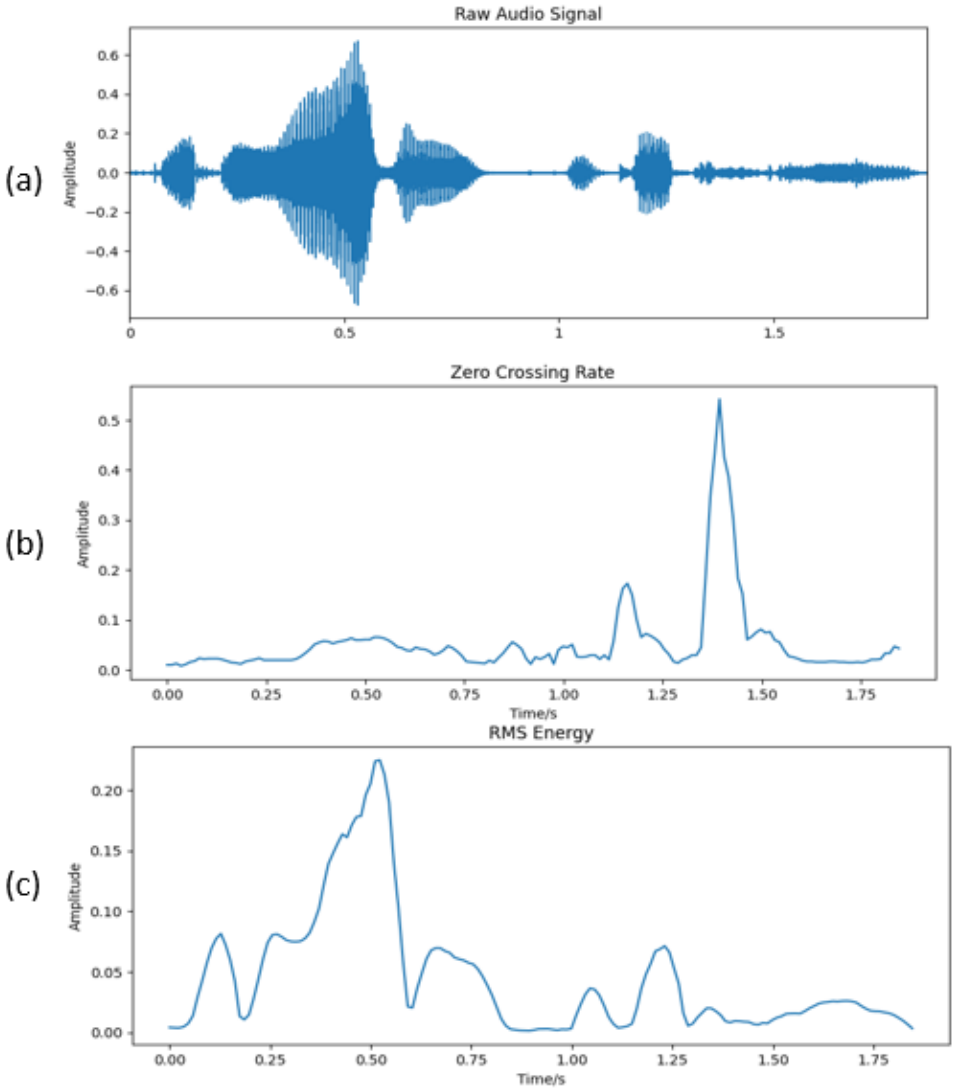

Fig. 6 Time domain features from '4-phrase.wav' (a) Raw signal waveform (b) Zero Crossing Rate (c) Root Mean Square

The frequency spectrum in Fig. 7 shows the dominant frequency components within the signal, however it is of limited use as there is no time component to the plot. The frequency spectrum considers the entire signal as one and cannot show variations in the frequencies present, making it susceptible to noise. It does however give one an idea as to which frequencies are present in the voice, with the majority in the 0 – 4000 Hz range.

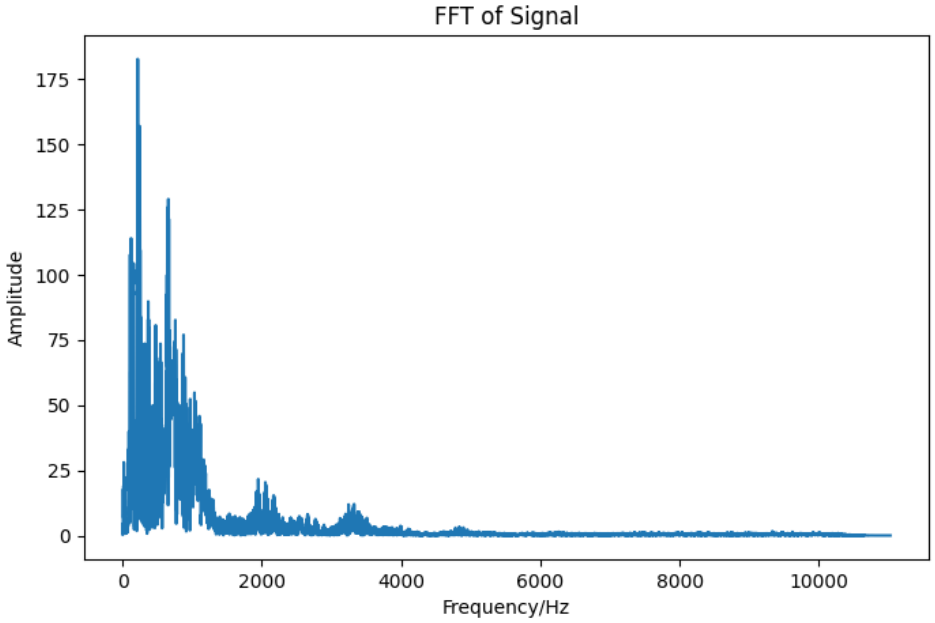

Fig. 7 Frequency Spectrum of '4-phrase.wav'

Fig. 8 shows the linear spectrogram of '4-phrase.wav', computed using the STFT to introduce time into the frequency spectrum. The spectrogram now shows the variation in frequencies over time, however, due to the linear scaling, only a small amount of information is captured as shown by the mostly black spectrogram. Comparing this with the Mel spectrogram in Fig. 9, it is clear that there is much more information present with dominant frequencies being shown within the dark red areas of the spectrogram, as well as a clear variation throughout the course of the signal. Fig. 10 is a graphical representation of the MFCCs from the '4-phrase.wav', which can be difficult to interpret visually, however upon inspection, is seen to loosely follow the Mel spectrogram from Fig. 9 with dark red areas in similar regions to the Mel spectrogram.

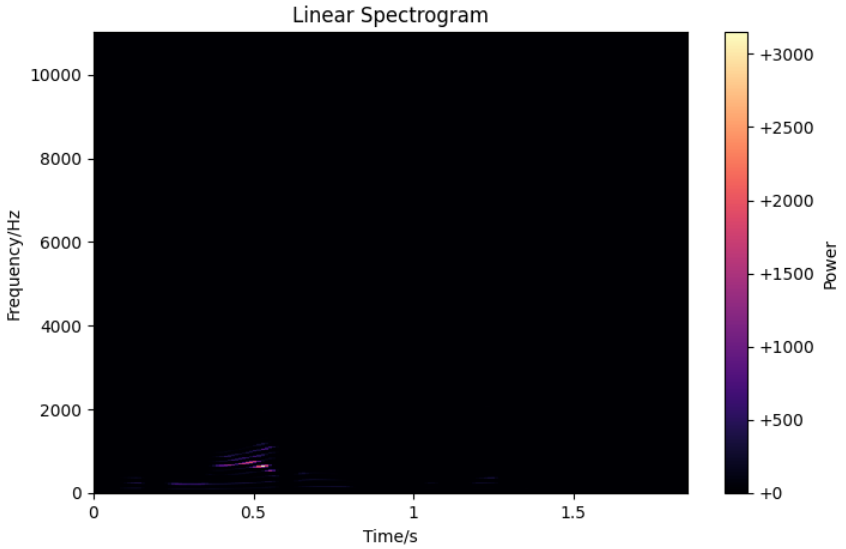

Fig. 8 Linear spectrogram from '4-phrase.wav'

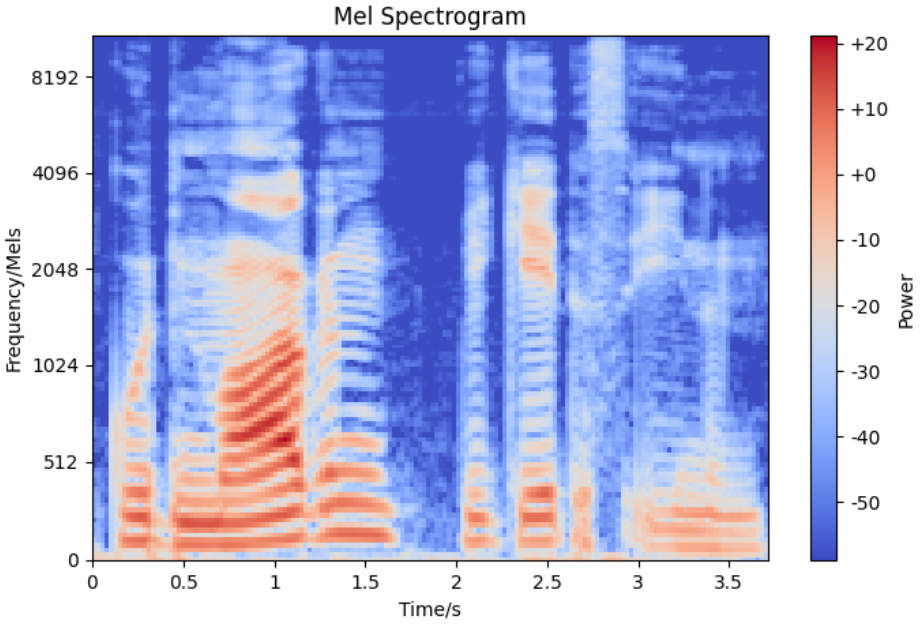

Fig. 9 Mel spectrogram from '4-phrase.wav'

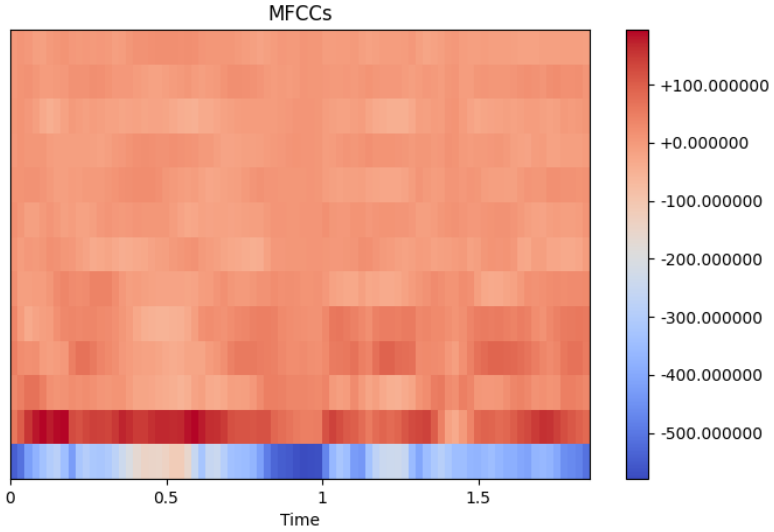

Fig. 10 MFCCs from '4-phrase.wav'

The spectral centroid is displayed in Fig. 11, with several peaks throughout the latter parts of the signal. The data within the plot contains a sharp peak near the 3 second mark, indicating that the signal is dominated by a 6000 Hz signal which is different to the dominant frequencies between 0.5 and 1.5 seconds in Fig. 9 and Fig. 10.

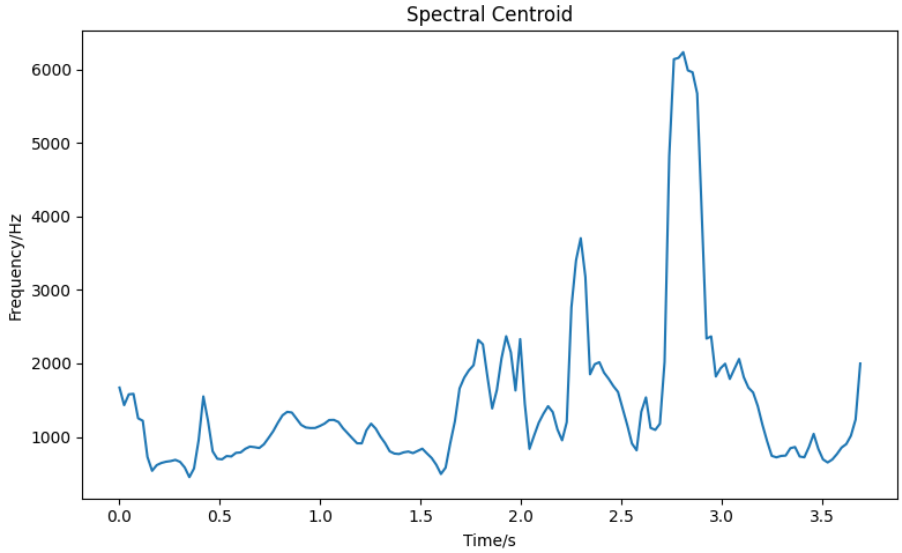

Fig. 11 Spectral centroid from '4-phrase.wav'

5.1.2 CNN Classification Results

After 30 epochs and a total of 10 tests for each feature, the mean accuracy, sensitivity, and specificity for the CNN classifier for all features are shown in Fig. 12 with the accompanying data table in Tab. 4, for the unbalanced test. The results for the balanced test are also shown in in graphical format in Fig. 13 and as a table in Tab. 5. The accuracy, sensitivity and specificity are calculated using (10), (11) and (12) where TP, TN, FN and FP represent the number of true positive, true negative, false negative and false positive results respectively. A true positive is defined as "an outcome where the model correctly predicts the positive class [38]," and a false positive is "an outcome where the model incorrectly predicts the negative class [38]." The definitions of true negative and true positive are vice versa.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{11}$$

$$Specificity = \frac{TN}{TN + FP} \tag{12}$$

The CNN produces greater classification accuracy when using an unbalanced test set, dominated by data from pathological speech recordings. Log Mel spectrograms performed the best with both test sets, having an accuracy of 0.871 using the unbalanced test and 0.794 with the balanced test set. The implications of an unbalanced versus balanced test set are explored in the discussion of results.

It is also clear that the time-frequency domain features used here, perform better with greater accuracies than the time-domain features using machine learning techniques, as implied by the literature [9], [12]. From these results, all features achieved a classification accuracy of greater than 0.600 further showing, along with the previously mentioned literature, that speech-based mental health classification is accurate to a reasonable extent and that many features do correlate with pathology, opposing the conclusions provided by [11]. With both test sets, the sensitivities for all features, besides shimmer with the balanced test set, are very

high reaching 0.946 with log Mel spectrograms in the balanced test. In contrast, the specificities are much lower, even reaching 0 with ZCR in the unbalanced test and the raw signal in the balanced test.

The time taken for both feature extraction and for the 10 tests for each feature are shown in Tab. 6, with shimmer and jitter taking the longest time to extract from all samples and the raw signal taking the most amount of time to be processed by the CNN classifier.

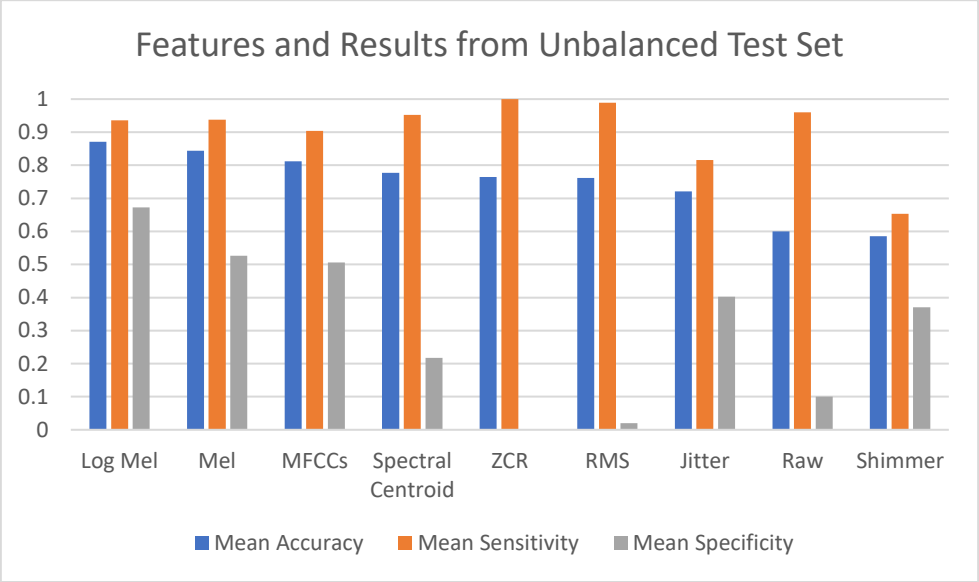

Fig. 12 Features and the mean results from 10 rounds of classification using the implemented CNN for the unbalanced test set

Feature	Mean Accuracy	Mean Sensitivity	Mean Specificity
Log Mel	0.871	0.936	0.673
Mel	0.843	0.938	0.527
MFCCs	0.812	0.904	0.506
Spectral Centroid	0.777	0.952	0.218
ZCR	0.764	1.000	0.000
RMS	0.762	0.989	0.020
Jitter	0.721	0.816	0.402
Raw	0.600	0.960	0.100
Shimmer	0.586	0.653	0.370

Tab. 4 Table of results from 10 rounds of classification using CNN for the unbalanced test set

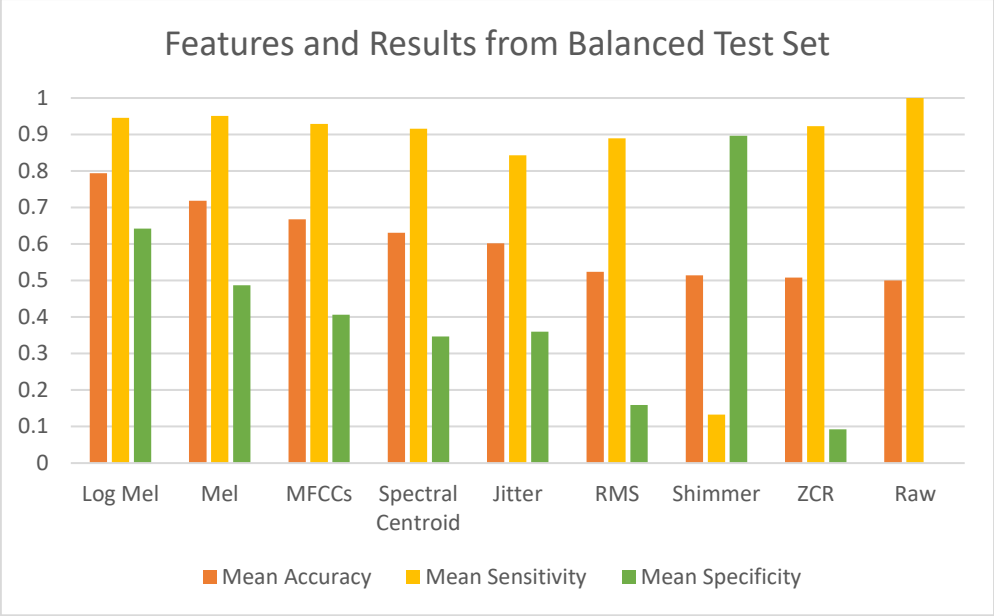

Fig. 13 Features and the mean results from 10 rounds of classification using the implemented CNN for the balanced test set

Feature	Mean Accuracy	Mean Sensitivity	Mean Specificity
Log Mel	0.794	0.946	0.642
Mel	0.719	0.951	0.487
MFCCs	0.667	0.929	0.406
Spectral Centroid	0.631	0.916	0.347
Jitter	0.602	0.843	0.360
RMS	0.524	0.889	0.159
Shimmer	0.514	0.133	0.896
ZCR	0.508	0.923	0.093
Raw	0.500	1.000	0.000

Tab. 5 Table of results from 10 rounds of classification using CNN for the balanced test set

Feature	Extraction Time	Classification and Testing Time
MFCC	55s	2m34s
Mel Spectrogram	1m09s	10m50s
Log Mel Spectrogram	1m10s	11m40s
Spectral Centroid	46s	2m10s
Jitter	22m37s	2m47s
Shimmer	23m14s	1m51s
RMS	44s	1m13s
ZCR	1m53	1m23s
Raw Signal	1m58s	39m10s

Tab. 6 Time taken for feature extraction and 10 tests of the CNN classifier

5.2 Discussion of Results

5.2.1 Results from Initial Feature Extraction

From initial feature extraction, the key results to discuss are the differences between the linear and Mel spectrogram in Fig. 8 and Fig. 9 as well as the spectral centroid in Fig. 11.

The Mel spectrogram shows much more information than the linear spectrogram because the non-linear Mel scale is used which correlates better with the frequencies that are present in human speech [28]. Through Fig. 7, one can see that most of the frequencies are in the lower range which can be distinguished better using the Mel scale [29], thus providing a more useful spectrogram with more information.

Despite Fig. 7 showing few dominant frequencies above 4000 Hz, the spectral centroid plot in Fig. 11, shows a peak at 6000 Hz indicating a high concentration of energy at this frequency. As the other plots do not show this, the peak is likely due to the way the spectral centroid is determined within Python or from some form of noise which was not accounted for with computation of the spectrograms or MFCCs.

5.2.2 Results from CNN Classifier

The most important results from this project are the accuracies and associated results from each feature when passed to the CNN classifier. The sensitivity and specificity, defined previously, indicate the proportion of the test set which were correctly classified as pathological and non-pathological respectively. A high sensitivity means that the classifier is good at correctly identifying recordings from pathological patients and a high specificity means that it is good at identifying recordings from healthy patients. It is important to understand this when assessing the two rounds of testing, using the unbalanced and balanced test set.

The results shown in Fig. 12 from the unbalanced test set show very high classification accuracies, exceeding the greatest accuracies achieved in [7], [8], [10] and [12] while matching [9]. However, as the test set is formed of a larger amount of data from pathological speech recordings, the results from the unbalanced test set are somewhat misleading regarding the actual performance of the features. The classifier tended to have very high sensitivities for nearly all of the tested features with lower specificities, so having a test set dominated by pathological samples will naturally result in a higher accuracy as the classifier is more likely to predict that a sample is from the pathological group rather than the healthy group. In turn, this generates misleading results as by looking at the accuracy without considering the impact of a skewed test set, it gives the illusion of a better performing classifier. This is evident when considering the classification accuracy achieved when inputting the raw audio signal into the CNN with the unbalanced test set, achieving an accuracy of 0.600. The raw audio signal was used here as a control feature, as it only represents the volume of the sound and little else, meaning that, in theory, the raw signal should not contain enough information to enable the CNN to use it effectively. Ideally the CNN should achieve an accuracy of approximately 0.500 as the classifier would have a 50/50 chance of classifying correctly: thus making it the worst performing feature. When using the unbalanced test set, a greater accuracy than this was achieved which suggests that the CNN can classify simply through the volume of the audio which should not be possible to this extent. To rectify these misleading results, a second set of tests were conducted, using a test set with an equal amount of data from both pathological and healthy control groups,

providing the baseline accuracy with the raw signal of 0.500. Henceforth, the discussed results will be that from the balanced test set from Fig. 13 and Tab. 5.

Log Mel spectrograms outperformed regular Mel spectrograms, however the reasons behind this are not explored within the any of the literature. [7] comes to a similar conclusion as discussed in the literature review, stating the log Mel spectrograms outperformed the other features, however there is no assessment of this and no comparison to regular Mel spectrograms. The log Mel spectrogram differs to the Mel spectrogram with regards to the power scale, being in decibels (dB) as opposed to a linear scale. It is likely that the decibel scale allows for better classification due to the fact that "a one-decibel difference in loudness between two sounds is the smallest difference detectable by human hearing [39]," and thus aligns better with human speech, resulting in better accuracy and explaining the superior classification using log Mel spectrograms.

MFCCs performed worse than the spectrograms as also found in [7]. This is because, "traditional MFCC systems use only 8-13 cepstral coefficients… [31]," which is a more compressible and simpler representation of the signal compared to a fully-fledged Mel spectrogram. Despite this expected performance, the classification using MFCCs with this CNN classifier was significantly lower than that of the literature which ranged from 70-87% as opposed to the 66.7% accuracy here. This may be due to reasons such as the CNN design or the parameters for MFCC extraction not being ideal but is difficult to determine since the much of the literature, [7], [8], [9] and [12], does not provide enough detail regarding either the extraction of MFCCs or the machine learning algorithm. Many machine learning algorithms also function in a way similar to 'black box' as found in [9], meaning that it is difficult identify exactly what the algorithm is doing and can therefore be difficult to improve

Spectral centroid was not a feature used within the literature yet resulted in an accuracy of 0.631, close to that of MFCCs. As the spectral centroid is known to represent the 'brightness' of the sound, with many mental pathologies exhibiting themselves in way which affect the tonality and 'brightness' of the speech produced [15], it would be expected that the spectral centroid would perform fairly well when classifying speech recordings from pathological subjects. Jitter falls under a similar category, performing marginally worse than the spectral centroid while also rarely being used as a feature for classification, despite its investigation for links with pathology in [6], [11]. Jitter was the only time-domain feature to achieve an accuracy above 0.600, unlike shimmer which also shares a known relationship with pathology as discussed in the background theory. The reasons for this may be due to the large range of pathologies used to obtain the data. Jitter is known to be a "good indicator of the presence of pathologies in the larynx such as vocal fold nodules… [22]," and the dataset may have been more heavily dominated by pathologies which exhibit themselves in this way, thus causing shimmer to result in lower accuracy tests. Whether this applies is difficult to establish as the dataset does not provide information about which pathologies are contained in each recording.

RMS as well as the raw signal both essentially represent only the volume of signal and therefore did not perform well within this setup. This is likely due to the large amount of variation within these features as shown in Fig. 6a and Fig. 6c which is caused by normal changes in volume as the subject speaks, with no distinguishable patterns being found by the CNN in the presence of pathology.

Finally, the course spectral characteristics described by ZCR [17], did not contain the necessary information and patterns needed to characterise pathology by the CNN, resulting in the poor performance of this feature. The representation is too basic for the CNN to discern a trend and thus could not be used to effectively classify the data.

The very high sensitivity and low specificity of the CNN with both test sets is likely due to the design and parameters of the CNN causing overfitting. Overfitting is "when a model learns the detail and the noise in training data to the extent that it negatively impacts the performance of the model on new data [40]," and can be a result of the algorithm not knowing how much detail to learn from the training set. This is common in less developed machine learning setups where the exact parameters are not finely tuned. Despite this effect, the performance of the features can still clearly be identified, with the literature [7] - [10], [12] demonstrating the greater potential classification accuracies of MFCCs, for example.

6 Conclusion

6.1.1 Summary of Findings

In summary, acoustic features have been extracted from audio recordings and analysed before being passed to CNN to classify pathological and healthy speech samples. The performance of each feature has been assessed, with logical reasoning behind why each feature performed in such a way. Sensitivities and specificities along with in depth information regarding feature extraction as well as the CNN classifier have been provided, unlike the majority of the literature allowing the work to be reproduced and developed in the future if necessary.

From the features tested, the best performing feature was the log Mel spectrogram, achieving an accuracy of 0.794, sensitivity of 0.946 and specificity of 0.642 with the accuracy matching that of the best performing features within the literature [7] - [10] and [12]. Many of the simpler, time domain features such as ZCR, RMS, shimmer as well as the raw signal did not perform well, with accuracies marginally above 0.500. Features that were not often used within the literature also achieved good results with the spectral centroid performing almost as well as MFCCs, a feature often used within the field, and jitter only performing marginally worse than the spectral centroid. As these features have not been individually tested in previous work, this provided valuable information and shows that there is some correlation between these features and mental pathology. With more development, the process of using acoustic features from speech recordings to detect mental pathology, could one day be implemented into the healthcare systems around the world, reducing the need for lengthy diagnosis from healthcare professionals as well as the stigma around mental health.

6.1.2 Future Work and Improvements

To improve the results obtained in this work and develop it further, there are several design aspects which could be modified to potentially increase the classifier's performance.

The design parameters of the CNN such as the number of layers, batch size and kernel size were not changed within the tests and thus the effect of changing these is unknown and may lead to better performance and may combat the overfitting found in these tests. The use of a CNN altogether could also be changed, instead substituting for other forms of machine learning algorithm such as the RNN used in [9], which may perform better with features not tested in [9], such as spectral centroid and jitter.

With regards to the features extracted, only a small number of features were used in these tests, with all features being tested separately. As some features are more susceptible to noise, methods to clean and filter the audio could be explored to prevent the classifier learning trends caused by noisy data. In addition, to maximise the performance of each feature, the parameters behind the extraction such as those in Tab. 3, could be tested in different configurations. If all features are extracted to their highest potential, combining the best performing features in different ways could also yield good results and boost the accuracy of the classifier.

If the accuracy, sensitivity, and specificity were consistently high enough using a configuration such as that used here, healthcare systems could adopt this and implement it into the diagnostic process, revolutionising the healthcare system and allowing more resources to be allocated towards treatment.

7 References

[1] M. Holmqvist, "Psychological Pathology," *Encyclopedia of Behavioral Medicine,* pp. 1559-1560, 2013.

[2] "Mental Disorders," Who.int, 2021. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/mental-disorders. [Accessed 19 March 2022].

[3] S. Dattani, "Mental Health," Our World in Data, [Online]. Available: https://ourworldindata.org/mental-health. [Accessed 19 March 2022].

[4] G. Bedi, F. Carrillo, G. A. Cecchi, D. F. Slezak, M. Sigman, N. B. Mota, S. Ribeiro, D. C. Javitt, M. Copelli and C. M. Corcoran, "Automated analysis of free speech predicts psychosis onset in high-risk youths," *npj Schizophrenia,* vol. 1, no. 1, 2015.

[5] "Britain: A nation of undiagnosed and untreated mental health conditions," [Online]. Available: https://www.openaccessgovernment.org/undiagnosed-and-untreated-mental-health/64510/. [Accessed 19 March 2022].

[6] D. M. Low, K. H. Bentley and S. S. Ghosh, "Automated assessment of psychiatric disorders using speech: A systematic review," *Laryngoscope Investigative Otolaryngology,* vol. 5, no. 1, pp. 96-116, 2020.

[7] B. Suhas, J. Mallela, A. Illa, B. Yamini, N. Atchayaram, R. Yadav, D. Gope and P. K. Ghosh, "Speech task based automatic classification of ALS and Parkinson's Disease and their severity using log Mel spectrograms," *2020 International Conference on Signal Processing and Communications (SPCOM),* pp. 1-5, 2020.

[8] H. Wu, J. Soraghan, A. Lowit and G. Di Caterina, "Convolutional Neural Networks for Pathological Voice Detection," *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC),* pp. 1-4, 2018.

[9] S. A. Syed, M. Rashid, S. Hussain and H. Zahid, "Comparative Analysis of CNN and RNN for Voice Pathology Detection," *BioMed Research International,* pp. 1-8, 2021.

[10] V. Rapcan, S. D'Arcy, S. Yeap, N. Afzal, J. Thakore and R. B. Reilly, "Acoustic and temporal analysis of speech: A potential biomarker for schizophrenia," *Medical Engineering & Physics,* vol. 32, no. 9, pp. 1074-1079, 2010.

[11] J. Shao, J. K. MacCallum, Y. Zhang, A. Sprecher and J. J. Jiang, "Acoustic analysis of the tremulous voice: Assessing the utility of the correlation dimension and perturbation parameters," *Journal of Communication Disorders,* vol. 43, no. 1, pp. 35-44, 2010.

[12] J.-Y. Lee, "Experimental Evaluation of Deep Learning Methods for an Intelligent Pathological Voice Detection System Using the Saarbruecken Voice Database," *Applied Sciences,* vol. 11, no. 15, p. 7149, 2021.

[13] B. Woldert-Jokisz, "Saarbruecken Voice Database," 2007.

[14] C. Peng, W. Chen, X. Zhu, B. Wan and D. Wei, "Pathological Voice Classification Based on a Single Vowel's Acoustic Features," *7th IEEE International Conference on Computer and Information Technology (CIT 2007),* 2007.

[15] "Assessing Speech," Medschool.co, [Online]. Available: https://medschool.co/exam/psych/assessing-speech. [Accessed 20 March 2022].

[16] mlearnere, "Learning from Audio: Time Domain Features," Towards Data Science, 2021. [Online]. Available: https://towardsdatascience.com/learning-from-audio-time-domain-features-4543f3bda34c. [Accessed 20 March 2022].

[17] T. Giannakopoulos and A. Pikrakis, Introduction to Audio Analysis, Elsevier Science, 2014.

[18] T. Bäckström, "Zero-crossing rate - Introduction to Speech Processing - Aalto University Wiki," [Online]. Available: https://wiki.aalto.fi/display/ITSP/Zero-crossing+rate. [Accessed 20 March 2022].

[19] J. P. Teixeira and A. Gonçalves, "Algorithm for Jitter and Shimmer Measurement in Pathologic Voices," *Procedia Computer Science,* vol. 100, pp. 271-279, 2016.

[20] "PointProcess: Get jitter (local)," Fon.hum.uva.nl, [Online]. Available: https://www.fon.hum.uva.nl/praat/manual/PointProcess__Get_jitter__local____.html. [Accessed 20 March 2022].

[21] " Definition of glottis | Dictionary.com," [Online]. Available: https://www.dictionary.com/browse/glottis. [Accessed 20 March 2022].

[22] D. G. Silva, L. C. Oliveira and M. Andrea, "Jitter Estimation Algorithms for Detection of Pathological Voices," *EURASIP Journal on Advances in Signal Processing,* no. 1, 2009.

[23] "Frequency Domain," DeepAI, [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/frequency-

domain#:~:text=The%20Frequency%20Domain%20refers%20to,among%20each%20given%20frequency%20band.. [Accessed 21 March 2022].

[24] E. Cheever, “Introduction to the Fourier Transform,” [Online]. Available: https://lpsa.swarthmore.edu/Fourier/Xforms/FXformIntro.html. [Accessed 10 December 2021].

[25] J. O. Smith, “Mathematics of the Discrete Fourier Transform (DFT) with Audio Applications, Second Edition,” 2007. [Online]. Available: http://ccrma.stanford.edu/~jos/mdft/. [Accessed 22 March 2022].

[26] J. Allen, “A unified approach to short-time Fourier analysis and synthesis,” *Proceedings of the IEEE,* vol. 65, no. 11, pp. 1558-1564, 1977.

[27] E. W. Weisstein, “Leakage,” MathWorld - A Wolfram Web Resource, [Online]. Available: https://mathworld.wolfram.com/Leakage.html. [Accessed 10 December 2021].

[28] L. Roberts, “Understanding the Mel Spectrogram,” Medium, 2021. [Online]. Available: https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53. [Accessed 22 March 2022].

[29] P. Pedersen, “The Mel Scale,” *Journal of Music Theory,* vol. 9, no. 2, pp. 295-300, 1965.

[30] M. Yusnita, M. Paulraj, S. Yaacob, R. Yusuf and A. Shahriman, “Analysis of Accent-Sensitive Words in Multi-Resolution Mel-Frequency Cepstral Coefficients for Classification of Accents in Malaysian English,” *International Journal of Automotive and Mechanical Engineering,* vol. 7, pp. 1053-1073, 2013.

[31] K. S. Rao and A. K. Vuppala, “Speech Processing in Mobile Environments,” *SpringerBriefs in Electrical and Computer Engineering,* 2014.

[32] M. Mohri, A. Rostamizadeh and A. Talwalkar, Foundations of Machine Learning, Second Edition, MIT Press, 2018.

[33] M. Sewak, M. Karim and P. Pujari, Practical Convolutional Neural Networks: Implement advanced deep learning models using Python, Packt Publishing, 2018.

[34] “Understanding of Convolutional Neural Network (CNN) — Deep Learning,” Medium, 2021. [Online]. Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148. [Accessed 11 December 2021].

[35] N. M. Aszemi and P. Dominic, “Hyperparameter Optimization in Convolutional Neural Network using Genetic Algorithms,” *International Journal of Advanced Computer Science and Applications,* vol. 10, no. 6, 2019.

[36] V. Velardo, 2020. [Online]. Available: https://github.com/musikalkemist/DeepLearningForAudioWithPython/blob/master/12-

%20Music%20genre%20classification:%20Preparing%20the%20dataset/code/extract_data.py. [Accessed 11 December 2021].

[37] V. Velardo, 2020. [Online]. Available: https://github.com/musikalkemist/DeepLearningForAudioWithPython/tree/master/16-%20How%20to%20implement%20a%20CNN%20for%20music%20genre%20classification/code. [Accessed 11 December 2021].

[38] "Classification: True vs. False and Positive vs. Negative | Machine Learning Crash Course | Google Developers," Google Developers, [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative. [Accessed 25 March 2022].

[39] "decibel," Encyclopedia Britannica, 28 February 2020. [Online]. Available: https://www.britannica.com/science/decibel. [Accessed 25 March 2022].

[40] J. Brownlee, "Overfitting and Underfitting With Machine Learning Algorithms," Machine Learning Mastery, [Online]. Available: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/. [Accessed 27 March 2022].

[41] Z. Li, W. Li, Z. Yao and Y. Li, "Speech-Based Five Features Extraction and Hardware Trade-Off Design for Mental Fatigue Monitoring," *2011 International Conference of Information Technology, Computer Engineering and Management Sciences,* 2011.

8 Appendix

A Initial Feature Extraction Code

```
import librosa
import librosa.display
import IPython.display as ipd
from librosa.feature.spectral import rms
import matplotlib.pyplot as plt
import numpy as np
import os


def amplitude_envelope(signal, frame_size, hop_length):
    amplitude_envelope = []

    for i in range(0, len(signal), hop_length):
        current_frame_amplitude_envelope = max(signal[i:i+frame_size])
        amplitude_envelope.append(current_frame_amplitude_envelope)

    return np.array(amplitude_envelope)


def plot_magnitude_spectrum(signal, title, xlabel, ylabel, sr, f_ratio=1):
    ft = np.fft.fft(signal)
    magnitude_spectrum = np.abs(ft)
    plt.figure(figsize=(8, 5))
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)

    frequency = np.linspace(0, sr, len(magnitude_spectrum))
    n_frequency_bins = int(len(frequency) * f_ratio)
    plt.plot(frequency[:n_frequency_bins],
             magnitude_spectrum[:n_frequency_bins])
    plt.title(title)


def plot_spectrogram(Y, sr, HOP_SIZE, title, xlabel, ylabel, barlabel,
y_axis='linear'):
    plt.figure(figsize=(8, 5))
    plt.title(title)
    librosa.display.specshow(
        Y, sr=sr, hop_length=HOP_SIZE, x_axis='time', y_axis=y_axis)
    plt.colorbar(format='%+2.f', label=barlabel)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)


def calculate_split_frequency_bin(spectrogram, split_frequency, sample_rate):
    frequency_range = sample_rate / 2
```

```
    frequency_delta_per_bin = frequency_range / spectrogram.shape[0]
    split_frequency_bin = np.floor(split_frequency / frequency_delta_per_bin)
    return int(split_frequency_bin)


def calculate_band_energy_ratio(spectrogram, split_frequency, sample_rate):
    split_frequency_bin = calculate_split_frequency_bin(
        spectrogram, split_frequency, sample_rate)
    power_spec = np.abs(spectrogram) ** 2
    power_spec = power_spec.T

    band_energy_ratio = []
    for frequencies_in_frame in power_spec:
        sum_power_low_frequencies = np.sum(
            frequencies_in_frame[:split_frequency_bin])
        sum_power_high_frequencies = np.sum(
            frequencies_in_frame[split_frequency_bin:])

        ber_current_frame = sum_power_low_frequencies /
sum_power_high_frequencies
        band_energy_ratio.append(ber_current_frame)

    return np.array(band_energy_ratio)


def main():
    FRAME_SIZE = 1024
    HOP_LENGTH = 256

    audio_file = "SVD/healthy_sentence_SVD/4-phrase.wav"
    audio, sr = librosa.load(audio_file)
    sample_duration = 1/sr
    duration = sample_duration * len(audio)

    # Amplitude Envelope
    ae_envelope = amplitude_envelope(audio, FRAME_SIZE, HOP_LENGTH)
    print("Number of frames: ", len(ae_envelope))

    frames = range(ae_envelope.size)
    t = librosa.frames_to_time(frames, hop_length=HOP_LENGTH)

    # RMS Energy
    rms_audio = librosa.feature.rms(
        audio, frame_length=FRAME_SIZE, hop_length=HOP_LENGTH)[0]

    # Zero Crossing Rate
    zcr_audio = librosa.feature.zero_crossing_rate(
        audio, frame_length=FRAME_SIZE, hop_length=HOP_LENGTH)[0]
```

```
# PLOTTING

# Raw signal
plt.figure(figsize=(10, 4))
librosa.display.waveplot(audio)
plt.title("Raw Audio Signal")
plt.xlabel('Time/s')
plt.ylabel('Amplitude')

# Plot Zero Crossing Rate
plt.figure(figsize=(10, 4))
plt.ylabel('Amplitude')
plt.plot(t, zcr_audio)
plt.title("Zero Crossing Rate")
plt.xlabel('Time/s')

# Plot rms
plt.figure(figsize=(10, 4))
plt.plot(t, rms_audio)
plt.ylabel('Amplitude')
plt.title("RMS Energy")
plt.xlabel('Time/s')

# DISCRETE FOURIER TRANSFORM
# Plot Fourier Transform
plot_magnitude_spectrum(audio, "FFT of Signal",
                        'Frequency/Hz', 'Amplitude', sr, 0.5)

# Linear Spectograms
s_audio = librosa.stft(audio, n_fft=FRAME_SIZE, hop_length=HOP_LENGTH)
y_audio = np.abs(s_audio) ** 2
plot_spectrogram(y_audio, sr, HOP_LENGTH,
                 'Linear Spectrogram', 'Time/s', 'Frequency/Hz', 'Power')

# Mel Spectrograms
mel_spectrogram = librosa.feature.melspectrogram(
    audio, sr=sr, n_fft=FRAME_SIZE, hop_length=HOP_LENGTH)
log_mel_spectrogram = librosa.power_to_db(mel_spectrogram)

plt.figure(figsize=(8, 5))
plt.title('Mel Spectrogram')
librosa.display.specshow(
    log_mel_spectrogram, x_axis='time', y_axis='mel', sr=sr)
plt.xlabel('Time/s')
plt.ylabel('Frequency/Mels')
plt.colorbar(format='%+2.f', label='Power')

# Spectral Centroid
```

```
    sc_audio = librosa.feature.spectral_centroid(
        y=audio, sr=sr, n_fft=FRAME_SIZE, hop_length=HOP_LENGTH)[0]

    # FIGURE 9 - SPECTRAL CENTROID
    frames = range(len(sc_audio))
    t = librosa.frames_to_time(frames)
    plt.figure(figsize=(8, 5))
    plt.plot(t, sc_audio)
    plt.title("Spectral Centroid")
    plt.xlabel("Time/s")
    plt.ylabel("Frequency/Hz")
    plt.tight_layout()
    plt.show()


if __name__ == "__main__":
    main()
```

B MATLAB Code to Split Recordings

```
clc
close all
folder ='SVD/healthy_sentence_SVD';
audio_files = dir(fullfile(folder,'*.wav'));

for k=1:numel(audio_files)


    filename = audio_files(k).name;
    [audioIn,Fs] = audioread(filename);
    num_segments = floor(length(audioIn)/fs);


    start = 1;
    for i = 1:num_segments
        data = (audioIn(start:fs*start));
        name = erase(filename,".wav");
        new_name = sprintf("%s%d.wav",name,i);
        new_name = "SVD/splithealthy/" + new_name;
        audiowrite(new_name,data,Fs);
    end
end
```

C Jitter Extraction Python Code

```
import os
import librosa
```

```
import json
import parselmouth
import math

DATASET_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\SVD\\split_files'
JSON_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\data\\jitter_data.json'

DURATION = 1
SAMPLE_RATE = 22050

period = 1/44


def save_mfcc(dataset_path, json_path):
    data = {
        'mapping':  [],
        'jitter': [],
        'labels': []
    }

    for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):
        if dirpath is not dataset_path:
            dirpath_components = dirpath.split('\\')
            label = dirpath_components[-1]
            data['mapping'].append(label)
            print('\nProccesing {}'.format(label))

            for f in filenames:
                jitter_list = []
                file_path = os.path.join(dirpath, f)

                for j in range(44):
                    snd = parselmouth.Sound(file_path)
                    pitch = snd.to_pitch()
                    pulses = parselmouth.praat.call(
                        [snd, pitch], "To PointProcess (cc)")
                    pointProcess = parselmouth.praat.call(
                        snd, "To PointProcess (periodic, cc)", 75, 300)

                    jitter_local = parselmouth.praat.call(
                        pulses, "Get jitter (local)", j * period, (j+1) *
period, 0.00001, 0.02, 1.3) * 100

                    if math.isnan(jitter_local):
                        jitter_local = 0

                    jitter_local = [jitter_local]
```

```
                    jitter_list.append(jitter_local)

                data['jitter'].append(jitter_list)
                data['labels'].append(i-1)
                print('{}'.format(file_path))

    with open(json_path, 'w') as fp:
        json.dump(data, fp, indent=4)


if __name__ == '__main__':
    save_mfcc(DATASET_PATH, JSON_PATH)
```

D Log Mel Spectrogram Extraction Python Code

```
import os
import librosa
import json


DATASET_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\SVD\\split_files'
JSON_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\data\\log_mel_data.json'

DURATION = 1
SAMPLE_RATE = 22050


def save_mfcc(dataset_path, json_path, n_mfcc=13, n_fft=2048, hop_length=512,
num_segments=1):
    data = {
        'mapping':  [],
        'mel': [],
        'labels': []
    }

    for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):
        if dirpath is not dataset_path:
            dirpath_components = dirpath.split('\\')
            label = dirpath_components[-1]
            data['mapping'].append(label)
            print('\nProccesing {}'.format(label))

            for f in filenames:
                file_path = os.path.join(dirpath, f)
                signal, sr = librosa.load(file_path, sr=SAMPLE_RATE)
```

```
                mel = librosa.feature.melspectrogram(
                    signal, sr=sr, n_fft=n_fft, hop_length=hop_length)
                log_mel_spectrogram = librosa.power_to_db(mel)
                log_mel_spectrogram = log_mel_spectrogram.T

                data['mel'].append(log_mel_spectrogram.tolist())
                data['labels'].append(i-1)
                print('{}'.format(file_path))

    with open(json_path, 'w') as fp:
        json.dump(data, fp, indent=4)


if __name__ == '__main__':
    save_mfcc(DATASET_PATH, JSON_PATH, num_segments=1)
```

E Mel Spectrogram Extraction Python Code

```
import os
import librosa
import json


DATASET_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\SVD\\split_files'
JSON_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\data\\mel_data.json'

DURATION = 1
SAMPLE_RATE = 22050


def save_mfcc(dataset_path, json_path, n_mfcc=13, n_fft=2048, hop_length=512,
num_segments=1):
    data = {
        'mapping':  [],
        'mel': [],
        'labels': []
    }

    for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):

        if dirpath is not dataset_path:
            dirpath_components = dirpath.split('\\')
            label = dirpath_components[-1]
            data['mapping'].append(label)
            print('\nProccesing {}'.format(label))
```

```
            for f in filenames:
                file_path = os.path.join(dirpath, f)
                signal, sr = librosa.load(file_path, sr=SAMPLE_RATE)
                mel = librosa.feature.melspectrogram(
                    signal, sr=sr, n_fft=n_fft, hop_length=hop_length)
                mel = mel.T
                data['mel'].append(mel.tolist())
                data['labels'].append(i-1)
                print('{}'.format(file_path))

    with open(json_path, 'w') as fp:
        json.dump(data, fp, indent=4)


if __name__ == '__main__':
    save_mfcc(DATASET_PATH, JSON_PATH, num_segments=1)
```

F MFCC Extraction Python Code

```
import os
import librosa
import json


DATASET_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\SVD\\split_files'
JSON_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\data\\mfcc_data.json'

DURATION = 1
SAMPLE_RATE = 22050
SAMPLES_PER_FILE = SAMPLE_RATE * DURATION


def save_mfcc(dataset_path, json_path, n_mfcc=13, n_fft=2048, hop_length=512,
num_segments=1):
    data = {
        'mapping':  [],
        'mfcc': [],
        'labels': []
    }

    num_samples_per_segment = int(SAMPLES_PER_FILE/num_segments)

    for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):
        # we want to make sure we are not at the root level
```

```
        if dirpath is not dataset_path:
            dirpath_components = dirpath.split('\\')
            label = dirpath_components[-1]
            data['mapping'].append(label)
            print('\nProccesing {}'.format(label))

            for f in filenames:

                file_path = os.path.join(dirpath, f)
                signal, sr = librosa.load(file_path, sr = SAMPLE_RATE)
                signal = signal[0:22049]

                for s in range(num_segments):
                    start_sample = num_samples_per_segment * s
                    end_sample = start_sample + num_samples_per_segment
                    mfcc =
librosa.feature.mfcc(signal[start_sample:end_sample],sr=sr, n_fft=n_fft,
                                                 n_mfcc=n_mfcc,hop_length=hop_l
ength)
                    mfcc = mfcc.T
                    data['mfcc'].append(mfcc.tolist())
                    data['labels'].append(i-1)
                    print('{}'.format(file_path))

    with open(json_path, 'w') as fp:
        json.dump(data, fp, indent=4)

if __name__ == '__main__':
    save_mfcc(DATASET_PATH, JSON_PATH, num_segments=1)
```

G Raw Signal Extraction Python Code

```
from cmath import nan
import os
import librosa
import json
import numpy as np
import parselmouth
import math

DATASET_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\SVD\\split_files'
JSON_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\data\\raw_data.json'
DURATION = 1
SAMPLE_RATE = 22050
period = 1/44
```

```
def save_mfcc(dataset_path, json_path):

    data = {
        'mapping':  [],
        'raw': [],
        'labels': []
    }

    for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):
        if dirpath is not dataset_path:
            dirpath_components = dirpath.split('\\')
            label = dirpath_components[-1]
            data['mapping'].append(label)
            print('\nProccesing {}'.format(label))

            for f in filenames:
                raw_list = []
                file_path = os.path.join(dirpath, f)
                signal, sr = librosa.load(file_path, sr=SAMPLE_RATE)
                signal = signal[0:22049]

                for j in range(44):
                    signal_segment = signal[j * 501: (j+1) * 501]
                    raw_list.append(signal_segment.tolist())

                data['raw'].append(raw_list)
                data['labels'].append(i-1)
                print('{}'.format(file_path))

    with open(json_path, 'w') as fp:
        json.dump(data, fp, indent=4)


if __name__ == '__main__':
    save_mfcc(DATASET_PATH, JSON_PATH)
```

H RMS Extraction Python Code

```
import os
import librosa
import json


DATASET_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\SVD\\split_files'
JSON_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\data\\rms_data.json'
```

```
DURATION = 1
SAMPLE_RATE = 22050


def save_mfcc(dataset_path, json_path, n_mfcc=13, n_fft=2048, hop_length=512,
num_segments=1):

    data = {
        'mapping':  [],
        'rms': [],
        'labels': []
    }

    for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):
        if dirpath is not dataset_path:
            dirpath_components = dirpath.split('\\')
            label = dirpath_components[-1]
            data['mapping'].append(label)
            print('\nProccesing {}'.format(label))

            for f in filenames:

                file_path = os.path.join(dirpath, f)
                signal, sr = librosa.load(file_path, sr=SAMPLE_RATE)
                rms = librosa.feature.rms(signal, hop_length=hop_length)[0]
                rms = rms.T
                data['rms'].append([rms.tolist()])
                data['labels'].append(i-1)
                print('{}'.format(file_path))

    with open(json_path, 'w') as fp:
        json.dump(data, fp, indent=4)


if __name__ == '__main__':
    save_mfcc(DATASET_PATH, JSON_PATH, num_segments=1)
```

I Shimmer Extraction Python Code

```
from cmath import nan
import os
import librosa
import json
import numpy as np
import parselmouth
import math
```

```
DATASET_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\SVD\\split_files'
JSON_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\data\\shimmer_data.json'

DURATION = 1
SAMPLE_RATE = 22050
period = 1/44


def save_mfcc(dataset_path, json_path):

    data = {
        'mapping':  [],
        'shimmer': [],
        'labels': []
    }

    for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):
        if dirpath is not dataset_path:
            dirpath_components = dirpath.split('\\')
            label = dirpath_components[-1]
            data['mapping'].append(label)
            print('\nProccesing {}'.format(label))

            for f in filenames:
                shimmer_list = []
                file_path = os.path.join(dirpath, f)

                for j in range(44):
                    snd = parselmouth.Sound(file_path)
                    pitch = snd.to_pitch()
                    pulses = parselmouth.praat.call(
                        [snd, pitch], "To PointProcess (cc)")
                    pointProcess = parselmouth.praat.call(
                        snd, "To PointProcess (periodic, cc)", 75, 300)

                    shimmer_local = parselmouth.praat.call(
                        [snd, pointProcess], "Get shimmer (local)", j *
period, (j+1) * period, 0.00001, 0.02, 1.3, 1.6)

                    if math.isnan(shimmer_local):
                        shimmer_local = 0

                    shimmer_local = [shimmer_local]
                    shimmer_list.append(shimmer_local)

                data['shimmer'].append(shimmer_list)
                data['labels'].append(i-1)
```

```
                print('{}'.format(file_path))

    with open(json_path, 'w') as fp:
        json.dump(data, fp, indent=4)


if __name__ == '__main__':
    save_mfcc(DATASET_PATH, JSON_PATH)
```

J Spectral Centroid Extraction Python Code

```
import os
import librosa
import json


DATASET_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\SVD\\split_files'
JSON_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\data\\centroid_data.json'
DURATION = 1
SAMPLE_RATE = 22050


def save_mfcc(dataset_path, json_path, n_mfcc=13, n_fft=2048, hop_length=512,
num_segments=1):
    data = {
        'mapping':  [],
        'centroid': [],
        'labels': []
    }

    for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):
        if dirpath is not dataset_path:
            dirpath_components = dirpath.split('\\')
            label = dirpath_components[-1]
            data['mapping'].append(label)
            print('\nProccesing {}'.format(label))
            for f in filenames:
                file_path = os.path.join(dirpath, f)
                signal, sr = librosa.load(file_path, sr=SAMPLE_RATE)
                centroid = librosa.feature.spectral_centroid(
                    signal, sr=sr, n_fft=n_fft, hop_length=hop_length)[0]
                centroid = centroid.T
                data['centroid'].append([centroid.tolist()])
                data['labels'].append(i-1)
                print('{}'.format(file_path))
```

```
    with open(json_path, 'w') as fp:
        json.dump(data, fp, indent=4)


if __name__ == '__main__':
    save_mfcc(DATASET_PATH, JSON_PATH, num_segments=1)
```

K ZCR Extraction Python Code

```
import os
import librosa
import json


DATASET_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\SVD\\split_files'
JSON_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\data\\zcr_data.json'

# DATASET_PATH = 'NEW\\recordings'
# JSON_PATH = 'NEW\\test_zcr_data.json'

DURATION = 1
SAMPLE_RATE = 22050


def save_mfcc(dataset_path, json_path, n_mfcc=13, n_fft=2048, hop_length=512,
num_segments=1):

    data = {
        'mapping':  [],
        'zcr': [],
        'labels': []
    }

    for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):

        if dirpath is not dataset_path:
            dirpath_components = dirpath.split('\\')
            label = dirpath_components[-1]
            data['mapping'].append(label)
            print('\nProccesing {}'.format(label))

            for f in filenames:
                file_path = os.path.join(dirpath, f)
                signal, sr = librosa.load(file_path, sr=SAMPLE_RATE)
```

```
                zcr = librosa.feature.zero_crossing_rate(
                    signal, hop_length=hop_length)[0]
                zcr = zcr.T
                data['zcr'].append([zcr.tolist()])
                data['labels'].append(i-1)
                print('{}'.format(file_path))

    with open(json_path, 'w') as fp:
        json.dump(data, fp, indent=4)


if __name__ == '__main__':
    save_mfcc(DATASET_PATH, JSON_PATH, num_segments=1)
```

L CNN Classifier Python Code

```
import json
from statistics import mean
import numpy as np
from sklearn.model_selection import train_test_split, StratifiedShuffleSplit
import tensorflow.keras as keras
import random

DATA_PATH =
'C:\\Users\\semih\\OneDrive\\Uni\\Project\\Code\\data\\mfcc_data.json'


def load_data(data_path):
    with open(data_path, 'r') as fp:
        data = json.load(fp)

    #X = np.array(data['mfcc'],dtype=object)
    X = np.array(data['mfcc'])
    Y = np.array(data['labels'])
    return X, Y


def prepare_datasets(test_size, validation_size):
    # load in the data
    X, Y = load_data(DATA_PATH)

    # # create train/test split
    # X_train, X_test, Y_train, Y_test = train_test_split(
    #     X, Y, test_size=test_size, stratify=Y)

    # Indices of healthy and pathological samples
    healthy = []
```

```
    pathological = []

    for i in range(len(Y)):
        if Y[i]:
            pathological.append(i)
        else:
            healthy.append(i)
    # Indices of random group of 83 samples from path and healthy
    healthy_test = random.sample(healthy, 83)
    path_test = random.sample(pathological, 83)

    all_test = healthy_test + path_test

    # Remaining indices that are not in test set
    healthy_train = [x for x in healthy if x not in healthy_test]
    path_train = [x for x in pathological if x not in path_test]

    all_train = healthy_train + path_train

    X_test = []
    X_train = []
    Y_test = []
    Y_train = []

    for index in all_test:
        X_test.append(X[index])
        Y_test.append(Y[index])

    for index in all_train:
        X_train.append(X[index])
        Y_train.append(Y[index])

    X_test, X_train, Y_test, Y_train = np.array(X_test), np.array(
        X_train), np.array(Y_test), np.array(Y_train)

    # create the train/validation split
    X_train, X_validation, Y_train, Y_validation = train_test_split(
        X_train, Y_train, test_size=validation_size, stratify=Y_train)

    # 3d array for each sample needed in tensorflow
    X_train = X_train[..., np.newaxis]
    X_validation = X_validation[..., np.newaxis]
    X_test = X_test[..., np.newaxis]

    return X_train, X_validation, X_test, Y_train, Y_validation, Y_test


def build_model(input_shape):
```

```
    # create model
    # CNN with 3 convolutional layers followed by max pooling
    model = keras.Sequential()

    # 1st conv layer - relu is recitified linear unit -- research
    model.add(keras.layers.Conv2D(
        32, (3, 3), activation='relu', input_shape=input_shape))
    # max pooling layer
    model.add(keras.layers.MaxPool2D((3, 3), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # 2nd conv layer
    model.add(keras.layers.Conv2D(
        32, (3, 3), activation='relu', input_shape=input_shape))
    # max pooling layer
    model.add(keras.layers.MaxPool2D((3, 3), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # 3rd conv layer - kernel size and max pooling size changed
    model.add(keras.layers.Conv2D(
        32, (2, 2), activation='relu', input_shape=input_shape))
    # max pooling layer
    model.add(keras.layers.MaxPool2D((2, 2), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # flatten output and feed to dense layer
    model.add(keras.layers.Flatten())
    # add dense layer - fully connected layer for classification - 64 neurons
used
    model.add(keras.layers.Dense(64, activation='relu'))
    # dropout for combatting overfitting - research
    model.add(keras.layers.Dropout(0.3))

    # output layer - number of neurons here equals number of classification
groups
    model.add(keras.layers.Dense(2, activation='softmax'))

    return model


def predict(model, X, Y):

    real_indexes = []
    predicted_indexes = []
    true_positives = 0
    true_negatives = 0
    false_positives = 0
    false_negatives = 0
    for i in range(len(X_test)):
```

```
            X = X_test[i]
            real_index = Y_test[i]

            X = X[np.newaxis, ...]
            prediction = model.predict(X)  # needs 4D array
            # 1D array which tells us the prediction
            predicted_index = np.argmax(prediction, axis=1)
            real_indexes.append(real_index)
            predicted_indexes.append(predicted_index)

        for j in range(len(real_indexes)):
            real = int(real_indexes[j])
            predicted = int(predicted_indexes[j][0])

            # if expected 1 and predicted 0
            if real == 1 and predicted != 1:
                false_negatives += 1
            # if expected 1 and predicted 1
            elif real == 1 and predicted == 1:
                true_positives += 1
            # if expected 0 and predicted 1
            elif real != 1 and predicted == 1:
                false_positives += 1
            # if expected 0 and predicted 0
            else:
                true_negatives += 1

        sensitivity = true_positives / (true_positives + false_negatives)
        specificity = true_negatives / (true_negatives + false_positives)
        # print("TP {} TN {} FP {} FN {}".format(true_positives,
        #       true_negatives, false_positives, false_negatives))

        n_pos = 0
        n_neg = 0
        for index in real_indexes:
            if index:
                n_pos += 1
            else:
                n_neg += 1

        print('Positives: {},  Negatives: {}'.format(n_pos, n_neg))

        return sensitivity, specificity


if __name__ == '__main__':

    accuracy_list = []
    specificity_list = []
```

```
    sensitivity_list = []
    for test_number in range(10):
        # create train, validation and test sets
        X_train, X_validation, X_test, Y_train, Y_validation, Y_test =
prepare_datasets(
            0.25, 0.2)

        # build the CNN net
        print(X_train.shape)
        input_shape = (X_train.shape[1], X_train.shape[2], X_train.shape[3])

        print('Input shape for model is: ', input_shape)
        model = build_model(input_shape)

        # compile the network
        optimiser = keras.optimizers.Adam(learning_rate=0.0001)
        model.compile(optimizer=optimiser,
loss='sparse_categorical_crossentropy',
                      metrics=['accuracy'])

        # train the CNN
        model.fit(X_train, Y_train, validation_data=(
            X_validation, Y_validation), batch_size=100, epochs=30)

        # evaluate the CNN on test set
        test_error, test_accuracy = model.evaluate(X_test, Y_test, verbose=1)
        print('Accuracy on test set is: {}'.format(test_accuracy))

        sensitivity, specificity = predict(model, X_test, Y_test)
        print("Sensitivity = {}".format(sensitivity))
        print("Specificity = {}".format(specificity))

        accuracy_list.append(test_accuracy)
        specificity_list.append(specificity)
        sensitivity_list.append(sensitivity)

    mean_accuracy = mean(accuracy_list)
    mean_sensitivity = mean(sensitivity_list)
    mean_specificity = mean(specificity_list)
    print("Mean Accuracy = {}".format(mean_accuracy))
    print("Mean Sensitivity = {}".format(mean_sensitivity))
    print("Mean Specificity = {}".format(mean_specificity))
```