Bird Call Heuristic-based Identification and Recognition Program (B-CHIRP)

A project in audio classification using machine learning

Presented by:
Alexander Cohen
CHNALE005

Prepared for:
Dr Simon Winberg
Dept. of Electrical and Electronics Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in Electrical and Computer Engineering

October 14, 2015

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signature:.....
Alexander Cohen

Date:.....

Terms of Reference

This project involves the implementation of an application that is capable of automatically identifying a small number of local South African birds based on a recording of its song/call. The project relates to the fields of signal processing, software development, algorithm development, and machine learning.

The terms of reference were discussed as an agreement between Dr Simon Winberg (project supervisor) and Alexander Cohen (student).

The expected outcomes of the project are as follows:

- The student should gain an understanding of the techniques used in audio classification, as well as machine learning. This should be done through academic research from sources such as academic journals, university-level papers, and approved internet sources.
- The student should design and build a proof-of-concept system that is capable to identify and classify a small number of local South African birds.
- This report is the primary deliverable which is to document the entire research and design process of the application. It should also document all experimentation and results and provide a final conclusion and recommendations for future work.

Acknowledgements

I would like to thank the following people for their help and support during this project:

Dr Simon Winberg - Thank you for providing valuable advice and assistance throughout the project, and for always being willing to help. The weekly GRT meetings helped to guide me in the right direction from the start of the project.

Dr Fred Nicols - Thank you for your advice with regards to machine learning and feature extraction.

My Family - Thank you for all the motivation and understanding. You have always believed in me and pushed me just the right amount in order to succeed.

Josh Karpul - Thanks for all the discussions, the nervous laughter, and the sarcasm we shared throughout our projects.

ElecComps & Lev - Thank you for being an amazing group of friends to have studied with for the past four years. I think it's safe to say we made it. Couldn't have done it without you guys.

Abstract

This project report documents a twelve week long design project, which aimed to develop a computer application which is capable of identifying a small number of local South African birds based on their call. The project draws inspiration from bird-watchers needing to identify birds which they can hear, but cannot see, as well as the need for biologists and environmentalists to be able to automate the process of tracking bird populations in various locations.

The design focuses around three main subsystems: syllable extraction, feature extraction, and classification. Syllable extraction is the process of locating and extracting bird chirps from a recording. An automated means of achieving this considered. Feature extraction involves extracting discriminatory features from data which can be used to classify the data. Finally, classification is achieved through the use of machine learning algorithms, namely the k-Nearest Neighbours Algorithm.

The final design implemented the above subsystems, with a graphical user interface encapsulating the system into a user application. The final system achieved an impressive overall classification accuracy of 68.4%, fulfilling all the objects set out at the start of the project.

With further refinement and design, the system could be improved in the future to identify more birds, with a higher accuracy.

Contents

List of Figures

List of Tables

Nomenclature

FFT – Fast Fourier Transform
GUI – Graphical User Interface
IDE – Integrated Development Environment
KNN – K-Nearest Neighbours Classification
LDA – Linear Discriminant Analysis
MP3 – MPEG-2 Audio Layer III
NN – Neural Network
PCM – Pulse Code Modulation
WAV – Waveform Audio File

Chapter 1

Introduction

Bird watching is a very popular pastime for many people. Often, one can be surrounded by birds which are hidden in the trees. One may want to identity these birds, but cannot do so by visual clues. Thus, identifying a bird by its call/song may be of interest. This project aims to develop a bird call identification program which can be used to identify a small number of birds found locally in South Africa.

1.1 Background to the Report

In everyday life, one hears the sounds of numerous animals such as dogs, insects, and birds. Many animals produce sounds as a form of communication or as a result of natural processes. Birds in particular have been of interest to people for hundreds of years, if not more. They have fascinated people with their vast collection of colours in their plumage, as well as captivated people with the songs which they sing. Bird watching is a growing pastime, and a means of identifying a bird by the sounds that it makes not only benefits bird-watchers, but proves to be a valuable tool for biologists and environmentalists as well. Often, biological and environmental research aims to asses the impact of human development on animal species. One way of achieving this it to identify and count the number of birds in a particular location over time. Bird vocalisations have evolved over time to become species specific, allowing for birds to be classified based solely on their vocalisations.

Most bird vocalisations involve short 'calls' rather than long and complex songs. These calls are used by birds to communicate among other things, aggression, alarm, and dis-

tress. The calls made by different birds can vary widely, and the acoustic make-up of these calls can be used to identify the bird making them.

1.2 Objectives

The main objective of this project is to create a program which is able to identify a few birds found locally in South Africa by their call/song. Below is a list of all the objectives which will be attempted.

1. Obtain a collection of bird recordings.
2. Identify classification methods and compare two.
3. Incorporate the better performing classifier into a fully functional program.
4. Produce a simple and intuitive graphical user interface for the program.
5. Determine how modifying certain aspects of the recordings affects accuracy and performance.

These objectives were refined after conducting the literature review. Refer to Chapter 3 and Chapter 4 for a more contextualised and refined view of the objectives.

1.3 Scope and Limitations

The design scope that was presented in the project outline was as follows: "This project will mainly focus on algorithm development and testing, collecting a variety of bird songs (of local birds) at different volumes and distortions, and then experimenting with filters and audio pattern matching methods, or possibly machine learning / neural networks to attempt to autonomously decide a type of bird based on the sounds it makes." The scope was further narrowed and more limitations imposed during the early stages of the project.

The collection of bird recordings was limited to a small size due to the lack of high-quality recordings of local South African birds, as well as the need to manually extract

chirps from the recordings. The process of manually extracting chirps is extremely time consuming for large datasets.

The system is designed for the purpose of academic research rather than commercial deployment. As such, the system is designed in a way which facilitates development, and is developed as an application running on a PC, rather than a smartphone application.

The external project constraints are listed below.

- Twelve weeks have been allocated for the design, development, testing, and documenting of the project.
- The project is to be carried out by a single student.
- A budget of R1000 is issued for the purchase of necessary hardware and/or software, if any.

1.4 Approach

The project is split into six distinct phases. Each phase relied on information and experience from completing the previous phases. Therefore work followed the order of the phases. The phases are listed below.

1. Phase 1 - Research and Literature Review.
2. Phase 2 - System Design and Planning.
3. Phase 3 - Setup and Development.
4. Phase 4 - Experimentation and Testing.
5. Phase 5 - Data Collection and Analysis.
6. Phase 6 - Conclusions and Recommendations.

The original timing plan for completion of these phases is presented in Figure 1.1 below. The time allocation is shown in days and deadlines were followed as closely as possible throughout the project.

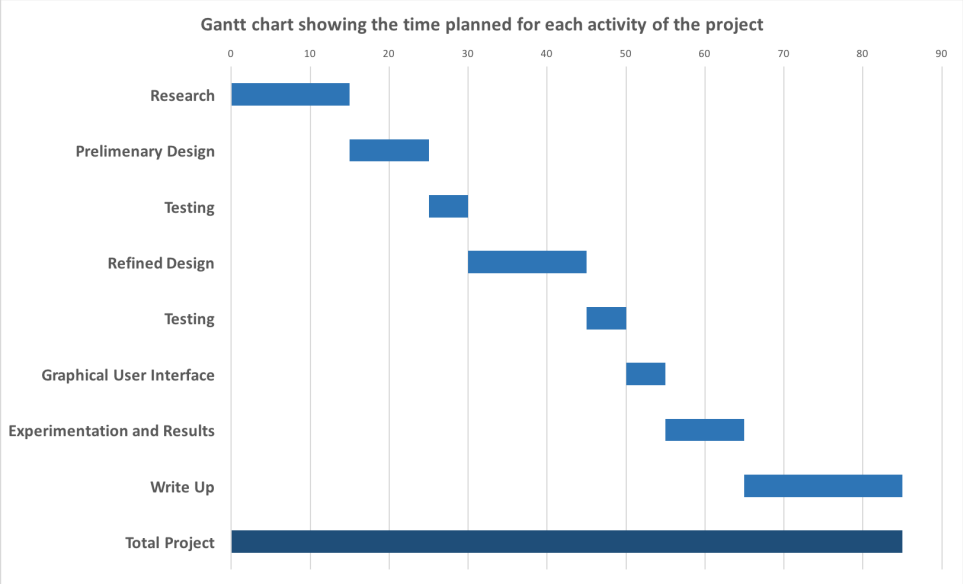


Figure 1.1: A Gantt chart showing the timeline of the project and the expected time to be spent on each phase of the project (in days).

1.5 Report Overview

This report is divided up into seven chapters. Each chapter covers a particular aspect of the project, the contents of which are discussed below.

Chapter 2 covers a review of the current literature in the field of audio classification and identification, as well as bird sounds, and machine learning.

Chapter 3 covers the methodology used in achieving the objectives of the project, and details the design process.

Chapter 4 covers the design and implementation of the B-CHIRP classification system.

Chapter 5 covers the results of the experimentation and testing performed on the system.

Chapter 6 presents conclusions based on the results obtained.

Chapter 7 presents recommendations for future work on the system.

Appendix A presents the results of a short survey conducted to asses the system.

Appendix B provides accreditation to the recording artists of the bird recordings used in this project.

Chapter 2

Literature Review

This chapter reviews the relevant literature required to gain a more in-depth understanding of the topics and designs presented in this report. It begins by explaining audio as a general topic, followed by audio in the context of bird vocalisations. It then goes on to explore how audio is represented digitally and stored in computers, followed by the methods of analysing audio, namely frequency analysis. The topic of machine learning is then covered with a broad overview of the current theory, followed by two machine learning techniques used in pattern recognition and classification problems. Finally, relevant software tools which will be of use during this project are explored.

2.1 Audio

2.1.1 Acoustics

Sound is produced by vibrations that propagate through molecular structures such as gasses (eg. air), liquids and solids. For this report, only sound wave propagation through air is considered. A vibrating object disrupts the air molecules around it. When the vibrating object is moving outwards, it pushes the molecules and thus compresses them and increases the pressure, forming a crest of a wave. The crest of the wave continues to move away from the object, as the molecules in the crest push against other molecules and disturb them. As the vibrating object moves inwards, it pulls molecules towards it and thus causes a decrease in pressure (rarefaction), forming a trough. The trough also moves away from the object in the same manner as the compression. These compressions and

rarefactions in the disturbed air produce crests and troughs respectively. It is important to note that the air molecules themselves do not move along with the crests and troughs, but rather displace adjacent molecules in a knock-on effect and then return to their resting position. The vibrating object producing the sound vibrates a number of times a second and thus sends out waves of pressure fluctuations which are interpreted as sound [1]. Figure 2.1 below provides a visual example of sound waves.

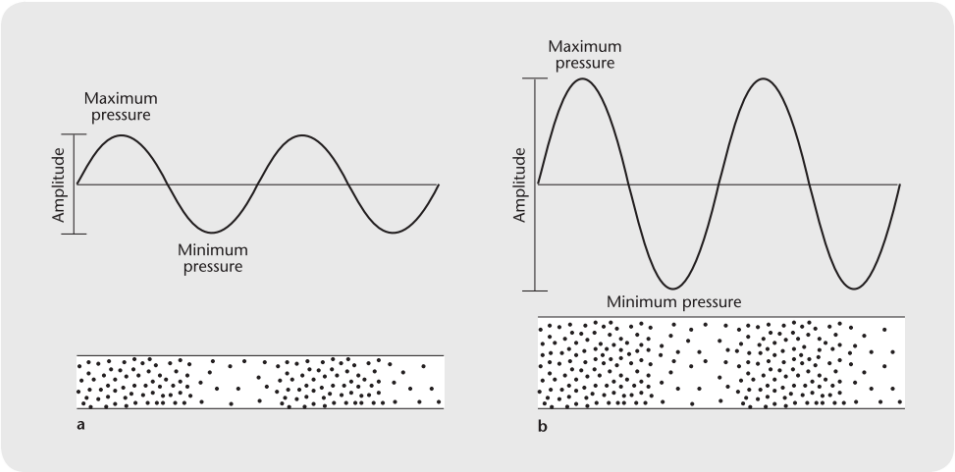


Figure 2.1: The number of molecules which are displaced by the vibrating object determines the amplitude (loudness) of the sound wave. In (b) there are more air molecules being displaced, which correspondingly causes a larger amplitude. [1, p. 3]

As shown in 2.1, the sound wave is represented on a graph, with amplitude on the vertical axis and time on the horizontal axis, making up the audio waveform plot. The corresponding compressions and rarefactions are shown beneath. The sound wave shown is a sine wave, and is the result of regular vibrations from a vibrating object. This is termed simple harmonic motion.

The rate at which an object vibrates is also the frequency of the sound produced. This frequency (f) is measured in Hertz (Hz) where 1 Hz is 1 cycle per second. Humans with excellent hearing are capable of hearing frequencies from 20 Hz to 20 kHz (20,000 Hz). The hearing range of birds is most sensitive between 1 and 4 kHz, with few birds showing sensitivity for infrasound (less than 20 Hz) and none showing any sensitivity to ultrasound (above 20 kHz) [2].

The decibel (dB) is a unit of measurement which compares the ratio of two values. This is used in relation to acoustic energy such as sound pressure, as well as electric energy such as voltage and power. In relation to acoustic energy, it is often used as a measurement of the loudness of a sound. If a reference value is known, the decibel can be used as an absolute unit of measure. It is defined mathematically as 10 times the logarithm to the

base 10 of the ratio between two signals, and is written as $dB = 10\log_{10}(P_1/P_0)$ where P_0 is the reference signal. For sound pressure levels, there is a worldwide defined value for P_0 which is given as $2 \times 10^{-5} Nm^{-2}$. [3]

2.1.2 Bird Vocalisation

The following information has been taken from [4, 5, 6]. Bird vocalisations can typically be divided into two categories: songs and calls. Songs are spontaneous complex vocalisations produced by songbirds. Songbirds account for approximately half of all birds and, unlike non-songbirds, have the ability to sing. Furthermore, songbirds have greater control in sound production, and are able to generate more complex sounds than non-songbirds. Hence the songs they produce are often much more complex in nature to the calls of non-songbirds. Typically, songs are produced by males during the breeding season and tend to be long and complex vocalisations aimed at attracting females. Some songbirds however, continue to sing throughout the year, including the female birds in some cases.

Simple voiced sounds can be characterised by both fundamental frequency and harmonics. These voiced sounds can be related to vowel sounds in human language. They form a crucial part of a bird's song. It has been found that the fundamental frequency of these voiced sounds often lies between 100Hz and 1kHz, depending on the species of bird. Birds are also able to produce pure tonal or whistled sounds, which do not include any harmonics. Both the voiced sounds and whistles can be changed in terms of the pitch and intensity. Birds are able to change the amplitude of the sound produced, both in terms of the fundamental as well as different harmonics. Furthermore, frequency changes can be made in a continuous manner, or as abrubt changes, yielding a wide array of possible sounds. Apart from voiced sounds and whistles, birds are also able to produce noisy sounds, whose waveforms are chaotic. These typically make up bird calls.

Calls are shorter, simpler vocalisations and typically produced by both males and females throughout the year. Calls are usually made in particular contexts and are in relation to specific functions such as flight, threat, or alarm. Calls are therefore less spontaneous than songs. Birds often have more than one version of their songs and calls, some have many. The collection of songs and calls which a bird uses is called a 'repertoire', with some birds having large repertoires.

Bird songs and calls can be broken up into a number of divisions such as phrases, syllables and elements. Figure 2.2 below illustrates these divisions.

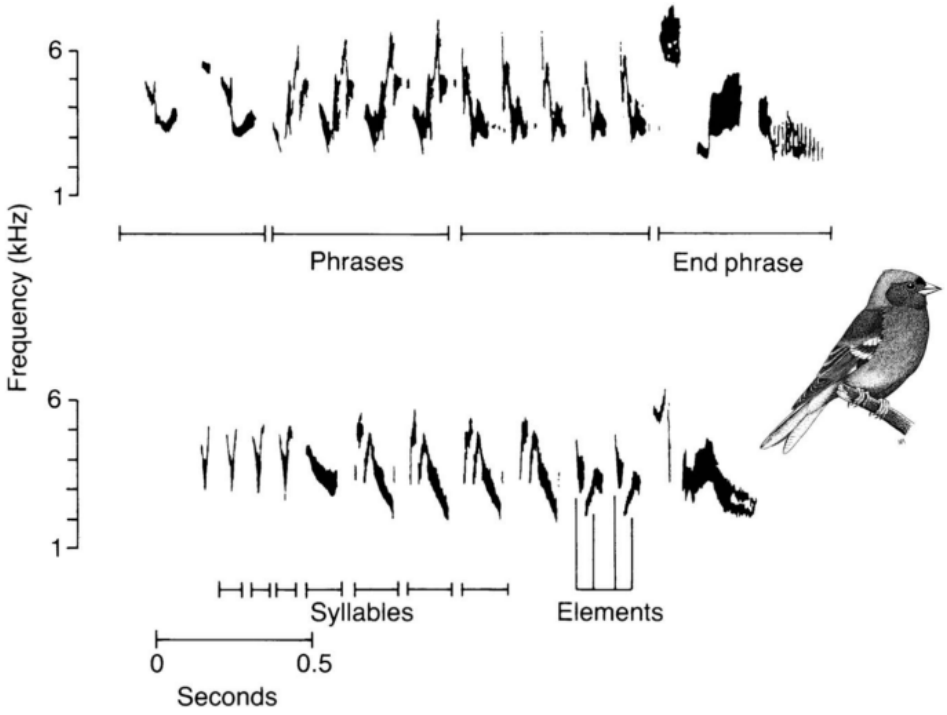


Figure 2.2: Spectrogram showing two song types of a Chaffinch bird with divisions into phrases, syllables and elements [6, p. 9].

The duration of a syllable is usually between a few milliseconds to a few hundred milliseconds, and can be seen as the building blocks of songs. Syllables have been identified as a suitable means of classifying birds, as classifying by syllable is technically more feasible than classifying by an entire song. Often birds sing at the same time, therefore isolating an entire song for classification is more difficult than isolating an individual syllable. Syllables can often be referred to as chirps, and from here on the terms are used interchangeably.

2.1.3 Digital Representation of Audio

The following information has been taken from [3, 7, 8]. An analogue signal is a signal which varies continuously in amplitude and frequency. A sound signal is just one example of an analogue signal. With the advent of computers, the need to digitally represent audio signals arose. This is achieved by converting the continuous analogue signal into a discrete signal. Computers store numbers using a finite number of bits, meaning that there is a limit to the precision, which prevents the storing of continuous values.

The process of converting an analogue signal to a digital one begins with the use of

a transducer, which is a device capable of converting one form of energy to another. In the case of audio digitisation, a microphone is used to convert the acoustic energy into electrical energy. The continuous energy values are then sampled at specific time intervals. These time intervals are defined by the sampling rate, the number of samples taken per second. The sampling rate should be at least twice the maximum frequency present in the signal, in order to allow the signal to be fully reconstructed without aliasing (distortion).

The next phase is quantisation, which is the process of converting the continuous value of a sample into a discrete, finite value. This can be seen as capturing the amplitude of each sample. The number of bits per sample determines how many different amplitude levels can be captured for each sample. If R bits are used to represent each sample, then 2^R different amplitude levels are possible.

Once all the quantisation values have been captured for each sample, the digital representation of the audio signal can be stored. The simplest form of digital storage for signals is pulse code modulation (PCM). In PCM, each sample is independently represented by the same number of bits. PCM is therefore the least efficient method of storing an audio signal, as no compression is involved. Compression involves the elimination of less audible or meaningful sounds in order to reduce the number of bits needed to represent an audio signal. Compression can be lossy, where information is permanently lost, preventing the signal from being restored to its original quality; or lossless, where the audio signal can be restored to its exact state before compression.

Different audio file formats are available for storing audio signals on computers. Some store the audio signal in an uncompressed format, while others store it in a compressed format. The Waveform Audio File Format (WAV) stores audio in an uncompressed manner, using PCM with 16 bits per sample, and a sampling rate of 44,100 Hz. The MP3 format (MPEG-2 Audio Layer III) compresses audio using lossy compression, resulting in significantly less storage space needed for each file.

2.1.4 Frequency Analysis

The concepts of the frequency domain and frequency representation are important aspects in signal processing. These concepts are discussed below.

The Fourier Transform

The following information has been taken from [9, 10, 11]. Sounds are made up of a complicated mixture of different frequencies with different amplitudes. By taking a linear combination of sines and cosines, it is possible to represent any continuous, periodic function. This representation is known as a Fourier Series Expansion. The process of finding the Fourier Series Expansion of a function is to transform it from the time domain to the frequency domain. This moving from time domain to frequency domain and vice versa is known as The Fourier Transform. The advantage of working in the frequency domain stems from the fact that each linear operation in the time domain has a corresponding operation in the frequency domain, and vice versa. Some operations are easier and less complex to perform in the frequency domain.

Suppose there exists a function $f(t)$, by taking a linear combination of sines and cosines, its Fourier Series Expansion can be written as:

$$f(t) = \sum_{k=1}^{n} (A_k \cos(2\pi\omega_k t) + B_k \sin(2\pi\omega_k t)) \tag{2.1}$$

where A is the amplitude, and ω is the frequency. A visual example of the Fourier Series Expansion is shown in Figure 2.3 below.

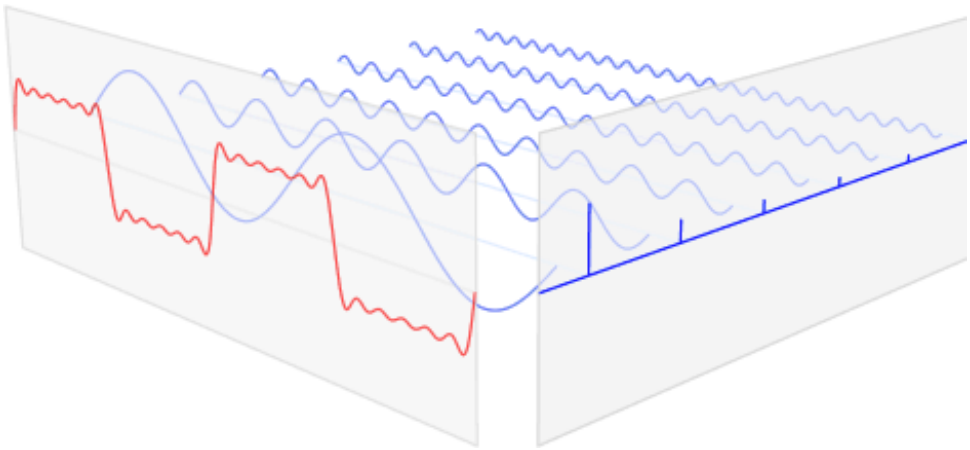


Figure 2.3: This figure shows how various sines and cosines (shown in blue) combine to form a signal (shown in red) [12].

The signal shown in red above is the function, and the signals shown in blue are the multiple sines and cosines which together, form a linear combination of the function.

Similarly to the Fourier Transform, the Discrete Fourier Transform (DFT) transforms a signal from the time domain to the frequency domain however, it works with a finite

amount of data, that being a sampled signal. Since the data is finite, the DFT can be implemented in computers. The Fast Fourier Transform (FFT) is an algorithm which computes the DFT much faster than using the mathematical definition of the DFT, and is widely used in signal processing.

Frequency Representation

The signal which has been transformed from the time domain to the frequency domain can be represented by the frequency spectrum. Figure 2.4 below provides an exampled of how a time domain signal may appear in the frequency spectrum.

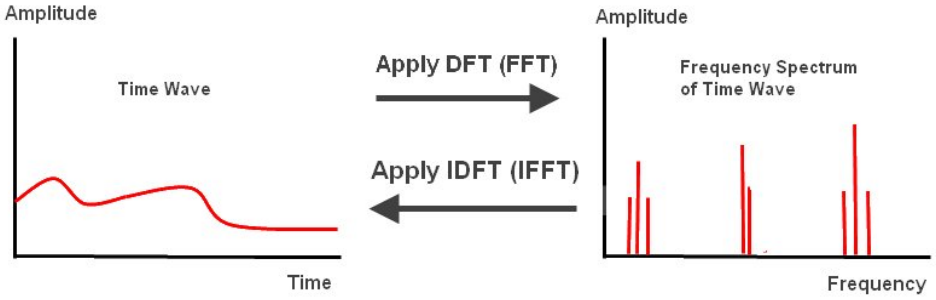


Figure 2.4: This figure shows the transformation of a signal from the time domain to the frequency domain [13].

In the figure above, the time domain signal on the left has been transformed to the frequency domain on the right. The frequency spectrum effectively shows the presence of frequency components which make up the original signal.

A spectrogram is also a visual representation of the frequencies present in a signal however, it shows how the frequencies vary with time. Figure 2.5 below is an example of a spectrogram.

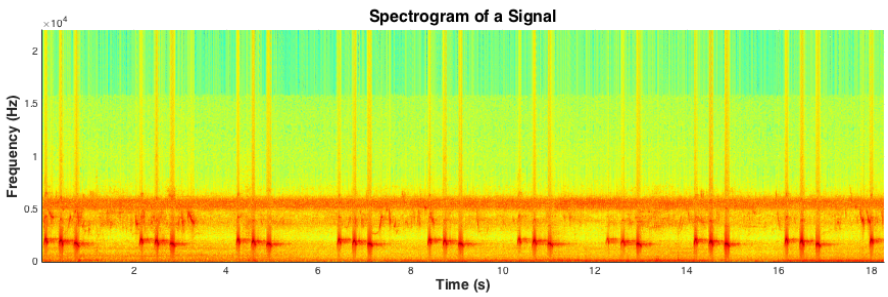


Figure 2.5: This figure shows the spectrogram of a signal.

The spectrogram above has time on the horizontal axis, frequency on the vertical axis, and a third dimension which shows the amplitude of the frequency as colour intensity.

2.1.5 Mel-Frequency Cepstrum Coefficients

The following information has been taken from [14, 15, 16]. Mel-frequency cepstrum coefficients (MFCC) are short-term spectral-based features, which represent the short-term power spectrum of a sound. Their values are based on the Mel Scale which was developed by Stevens et al. in 1937 as a scale of pitches which, when judged by listeners are considered to be equal distance from each other. The scale is therefore a perceptual scale, and a number of formulas exist to convert frequency to mels. The most common formula used is given as:

$$m = 2595 log_{10}(1 + \frac{f}{700}) \tag{2.2}$$

where m is in mels, and f is the frequency.

MFCCs have been the most widely used features in speaker recognition (determining *who* is speaking) due to its efficient computation schemes and its ability to represent speech in a compact form. MFCCs are commonly calculated according to the following algorithm:

MFCC Algorithm

1. Multiply the signal with a tapered window (most commonly a Hamming or Hanning window).
2. Compute the power spectrum using the FFT.
3. Map the powers of the spectrum onto the Mel Scale by passing the FFT through a triangular filter bank.
4. Compute the log value of the outputs from the filter bank.
5. Find the discrete cosine transform (DCT) of the log values.
6. The amplitudes of the DCT spectrum are the MFCCs.

2.2 Machine Learning

The following information has been taken from [17, 18]. The ability of discovering patterns in data is fundamental to understanding the data. Exceptional pattern recognition in humans has long been an advantage over other creatures. In the digital age, it is advantages to perform pattern recognition using computers, which are capable of processing data at an incredible rate compared to humans. Here, the concepts of machine learning are discussed, and how this leads to pattern recognition and classification techniques.

2.2.1 Overview

Machine learning today is abundant, although often hidden. With more data available than ever, due to the rise in computers and the internet, machine learning has had more data to work with, allowing us to draw deep insights and conclusions into the data. Machine learning appears in many forms, some of which are discussed below.

Pattern recognition, classification, and estimation are all forms of machine learning. Search engine web page results are presented from most relevant to least relevant. Machine learning is responsible for *learning* to recognise which web pages are relevant to certain search queries. Face recognition and speech recognition are classification problems which machine learning is applied to. The song identification application 'Shazam' is another example.

The basic premise behind machine learning is that there exists some dependence between a set of observations, call these x, and some desired output or response, call this y. Machine learning is the process of *learning* the non-trivial and often extremely complicated dependency between x and y.

2.2.2 Learning Problems

The range of learning problems which machine learning can be applied to is extremely large. A selection of these problems is discussed below.

Binary classification is a common problem in machine learning, where the response is usually ± 1 indicating the two possibilities. For example, given a set of images of cats

and dogs, it may be desirable to determine whether a picture is of a cat or a dog.

Multiclass classification is an extension to binary classification where the response can be one of many, rather then one of two. Figure 2.6 below provides a simple illustrative example of binary and multiclass classification.

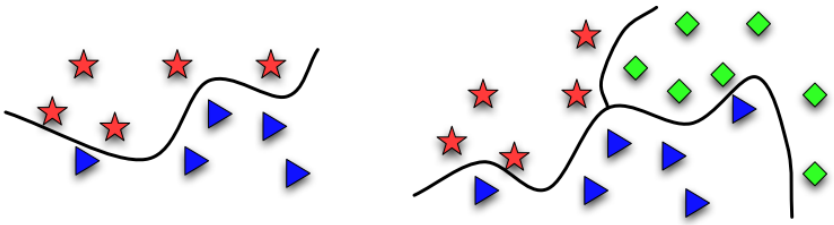


Figure 2.6: This figure shows an example of binary classification (left) and three-class classification (right) [17, p. 10].

Note that the three-class classification in the right becomes exponentially more difficult because, not only must the distinction between stars and diamonds be established, but *both* of these shapes need to also be distinguished from triangles.

2.2.3 Feature Extraction

Features are values which are fast to compute and are useful in discriminating between classes in a classification problem. For example, the average colour over a small sub-region in an image can be used as a feature. Features then form the input to a machine learning algorithm. The algorithm is trained on these features, in order make predictions or classifications from features of new, unseen data in the future.

Feature Evaluation

Methods of determining a feature's discriminative ability on a set of classes is desirable as it allows for the selection of features which will yield the best classification results. Fisher's Linear Discriminant Analysis is a method for measuring this discriminating ability. The method of computing the Fisher's score is presented in [19] and is as follows:

Compute

$$J = det(\frac{S_B}{S_W}) \tag{2.3}$$

where S_B is the 'between class scatter matrix' and S_W is the 'within class scatter matrix'. These scatter matrices are computed as follows:

$$S_B = \sum_c (\mu_c - \bar{x})(\mu_c - \bar{x})^T \tag{2.4}$$

$$S_W = \sum_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T \tag{2.5}$$

where $\bar{x}$ is the overall mean of the data, and μ_c is the mean within a class.

The higher the value obtained for J, the better a feature is at discriminating between the classes.

2.2.4 Neural Networks

The following information has been taken from [20, 21, 22]. Neural networks (NN) refer to multilayer networks which attempt to mimic connections in the human brain in order to solve problems such as pattern recognition and classification. Neural networks are therefore a form of machine learning.

The basic neural network is composed of three layers: an input layer, a hidden layer, and an output layer. Each of these layers consists of a number of nodes. The nodes from the input layer are connected to nodes in the hidden layer, and nodes in the hidden layer are connected to nodes in the output layer. Each connection has an associated weight, which defines how influential the connection is. Training the NN involves setting the weights, and this can be done in a supervised or unsupervised manner. Here, only supervised learning is discussed.

The nodes in the input layer accept the raw data into the NN. Each input node is connected to every hidden node, and every hidden node is connected to the output node. This connection scheme is depicted in Figure 2.7 below.

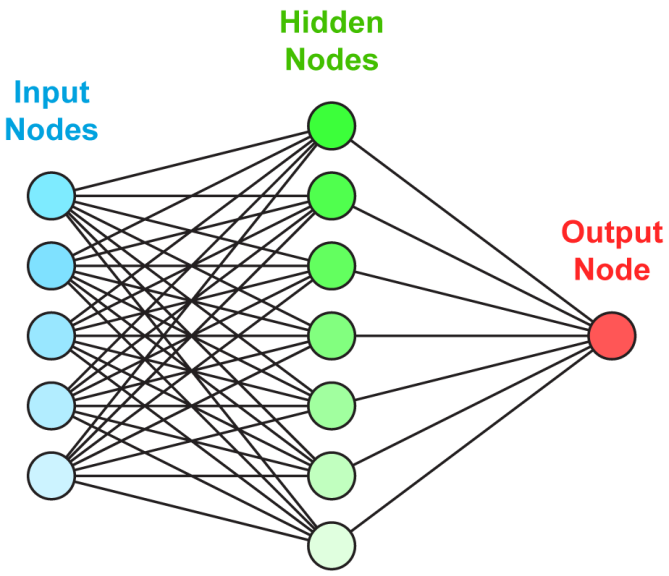


Figure 2.7: This figure shows an example layout and connection scheme of a typical neural network [23].

The values from the input nodes are sent to the hidden nodes after being modified according to the connection weight. Finally, the same process is repeated for the hidden nodes sending their values to the output node.

Picking the number of hidden layers, as well as the number of hidden nodes per layer is a design choice. For most problems, one hidden layer is enough. Choosing the right number of nodes for this layer though is a more difficult choice. Monitoring the progress of the NN during training is essential, and the number of hidden nodes can be modified if the NN results are not satisfactory.

One of the most popular NN training algorithms is the back propagation algorithm. Training involves a forward pass and a backwards pass through the network. Initially, the weights are randomly set. The algorithm itself consists of three stages:

1. **Feed-forward computation** - Data is provided to the network, and the outputs are compared to the desired outputs. By comparing these, an error rate is calculated.

2. **Back propagation** - In this stage, the error rates of individual nodes is calculated, moving from the output layer to the hidden layer, and then to the input layer.

3. **Update Weights** - Finally, the node error rates are used to calculate new weights for the connections. All of the newly calculated weights are updated simultaneously.

This process is repeated, with each iteration computing the overall error. If the error is reduced, the weights are kept, if it increases or remains unchanged, the weights are discarded and a new iteration takes place to calculate the weights. The process repeats until some pre-defined error value is reached, or a certain number of iterations has been reached.

2.2.5 k-Nearest Neighbours Algorithm

The following information has been taken from [24, 25]. The k-Nearest Neighbours Algorithm (KNN) is a method used for classification and estimation. The input to algorithm is a set of training data points, for which the class each data point belongs to is known. The output of the algorithm, when inputting new data is the prediction of which class the data belongs to. The classification of the new data is based on how similar it is to the training data.

As with the NN, the data input into a KNN classifier is a set of feature values, extracted from the original data. The number of features determines the dimensionality of the space in which the individual feature values will be plotted. Once the training feature values are positioned in this space, the features of the new, unseen data are plotted in the same space. The algorithm then predicts which class the new data belongs to by choosing the k nearest training data points, and taking the most common class among these.

To determine which neighbouring data points are the closest, a distance function is needed. The most commonly used distance function is the Euclidean distance function. Using this function, the distance d between two points x and u is calculated as:

$$d = \sqrt{\sum_{i=1}^{n}(x_i - u_i)^2} \tag{2.6}$$

The simplest case of the KNN classifier is $k = 1$, where the data point from the training set that is closest to the new data point is taken as the class of the new data. As k increases, more and more closest points are taken into consideration. The class which appears most often in these close points is taken to be the class of the new data.

The weighted KNN is an extension of the KNN, where training data points that are

particularly close to the new data points are weighted higher than points which are further away. This method still uses the k nearest points in the classification decision however, the class with the largest added weight is used as the classification outcome (as opposed to taking the class which appears most often within the k nearest points).

2.2.6 Measuring Classifier Accuracy

It is important to understand how to interpret information presented by confusion matrices. First, the concepts of *true positive* (TP), *true negative* (TN), *false positive* (FP), and *false negative* (FN) must be understood. These terms are characterised by the prediction made by a classifier in relation to the correct answer. This is summarised in Table 2.1 below.

Table 2.1: Machine Learning Predictions

	Prediction of Model: Positive	Prediction of Model: Negative
Truth: Positive	TP	FN
Truth: Negative	FP	TN

A simple confusion matrix is presented in Figure 2.8 below, and the important information it provides is discussed.

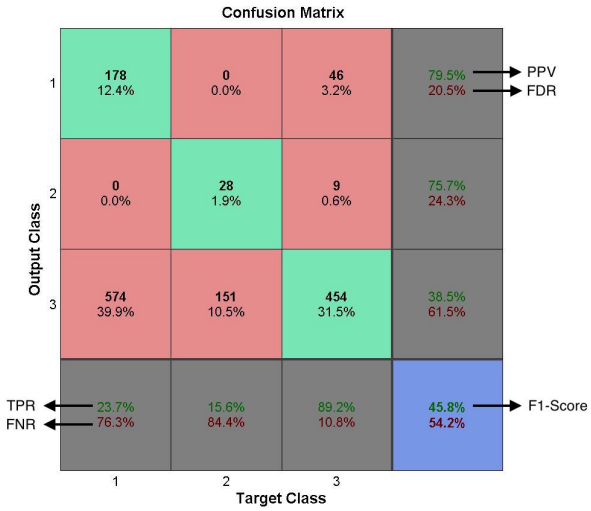


Figure 2.8: Example confusion matrix with important values highlighted.

In the rightmost column of the matrix, the green result indicates the *positive predictive value* (PPV) or *precision*. This value indicates the proportion of positive results which

are true positives, and is defined as:

$$PPV = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false positives}}. \tag{2.7}$$

Below the PPV value is the *false discovery rate* (FDR) which gives the percentage of false positives. It is defined as:

$$FDR = \frac{\text{number of false positives}}{\text{number of false positives} + \text{number of true positives}}. \tag{2.8}$$

In the bottommost row of the matrix, the green result indicates the *true positive rate* (TPR) or *recall*. It measures the proportion of positives that are correctly identified and is defined as:

$$TPR = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}}. \tag{2.9}$$

Below the TPR value is the *false negative rate* (FNR) which measures the miss rate - the times when the classifier predicts a value does not belong to a particular class, but in fact does.

$$FNR = \frac{\text{number of false negatives}}{\text{number of false negatives} + \text{number of true positives}}. \tag{2.10}$$

Finally, green result in the bottom-right cell indicates the F1-Score, which is the overall accuracy of the classification system. It is the harmonic mean of PPV and TPR and is defined as:

$$F1 = \frac{2 \times \text{number of true positives}}{2 \times \text{number of true positives} + \text{number of false positives} + \text{number of false negatives}}. \tag{2.11}$$

2.3 Software Tools and Libraries

This section provides a brief overview of the software tools and libraries considered for use.

2.3.1 Audacity

Audacity is a free, open-source audio editing application. It is capable of processing a wide variety of popular audio file types such as '.mp3' and '.wav', as well as many others through installation of optional libraries such as the FFmpeg library.

Audacity also provides features such as noise reduction, equalisation, normalisation, batch conversions and many more. It is therefore considered as a method of pre-processing or normalising audio for use within this project.

2.3.2 Matlab

The following information has been taken from [26]. Matlab is a fourth-generation, proprietary program language and development environment. It is developed by Mathworks and was initially released in 1984; it is currently in its 9th release version. Matlab is used widely in the fields of engineering and signal processing, mostly due to optional packages which can be installed to provide numerous additional features.

Matlab allows for matrix multiplication, easy graph plotting, algorithm implementation, as well as the creation of graphical user interfaces. Its functionality is vastly improved by the way it is able to handle resizeable arrays called cell arrays. These allow for multidimensional data to be stored and used with ease.

The online Matlab community is large and there exists a code-sharing portal where Matlab users are able to share their code and offer advice. Online seminars called 'Webinars' are also available which introduce users to the tools and features available within Matlab.

Finally, Matlab offers a number of toolboxes which provide additional algorithms, functions, and systems which can be used to aid development. The 'Neural Networking Toolbox' and 'Statistics and Machine Learning Toolbox' are of particular interest in the context of this project.

Chapter 3

Methodology

This chapter details the methodology and steps taken during each phase of the project, from inception to completion. It describes the tasks undertaken in each phase, and explains the iterative process followed during certain phases in order to accomplish the project objectives set out in subsection 1.2. Figure 3.1 below provides a visual representation of the major project phases.

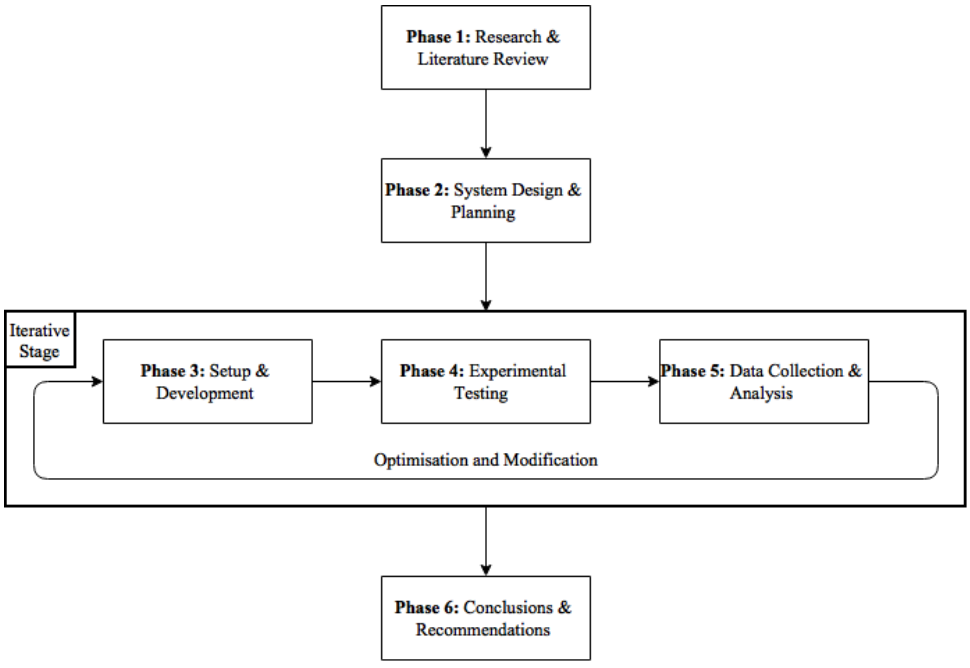


Figure 3.1: Phases of the Research Project

3.1 Phase 1: Research and Literature Review

The first phase of the project involved a comprehensive review of both past and current literature in the fields of bird songs and calls, audio analysis, and classification systems. The process of investigating these fields led to the refinement of the initial objects outlined in Section 1.2. The important aspect of feature extraction emerged during this process, which led to further research into potential features which could be used to classify bird sounds in particular. A suitable number of features were chosen to be tested for use within the system. The need to segment the recordings into small recordings each containing a syllable was also determined.

The classification systems chosen to be implemented and tested in the system were neural networks (NN) and KNN classifiers. This led to research into the software tools which could be used to implement NNs and KNNs. The Matlab computing language and development environment was chosen. Further research was conducted into additional software tools and packages which could be used in conjunction with Matlab for use within the project.

3.2 Phase 2: System Design and Planning

After reviewing the appropriate literature, a top-level system design was created and the necessary subsystems identified: syllable extraction, feature extraction, classifier. These are described in detail in Chapter 4. During this planning phase, the expected input requirements for each subsystem were considered. This led to the identification of necessary outputs for each subsystem, as many were to be used as inputs to interworking subsystems. A comprehensive data representation and variable-naming convention was set up to ease development, as well as facilitate future development.

3.3 Phase 3: Setup and Development

This section briefly describes the setup of the development environment used throughout the project, as well as the subsystems which were to be developed. It has been broken up into sections to distinguish between the development environment, and the development itself.

3.3.1 Development and Test Machines

Two Mac-OS computers were used during development, with both running the OS-X Yosemite (v10.10) operating system. The primary computer used was a MacBook Pro laptop; the testing and experimentation results were all performed on this machine for consistency. The secondary computer used was an iMac. The specifications for both machines is shown in Table 3.1 below.

Table 3.1: Specifications of Computers Used During Development

Class	Details (MacBook Pro)	Details (iMac)
System	iMac (27-inch, Late 2013)	MacBook Pro (13-inch, Mid 2010)
Operating System	OS-X Yosemite (v10.10)	OS-X Yosemite (v10.10)
Processor	3,2 GHz Intel Core i5	2,4 GHz Intel Core 2 Duo
Memory	8 GB 1600 MHz DDR3	4 GB 1067 MHz DDR3
Storage	1TB Fusion Drive (128GB SSD)	128GB SSD
Graphical Processing Unit	NVIDIA GeForce GT 755M 1024 MB	NVIDIA GeForce 320M 256 MB

3.3.2 Programming Languages

The Matlab development environment was used for the entire development of the system. The same Matlab distribution - R2015a (8.5.0.197613) - was installed on both computers.

Matlab was selected as the programming language of choice as it offers:

- A large collection of built-in algorithms for signal processing.
- Clearly written documentation with many examples, as well as online resources such as web seminars ("Webinars")[1].
- A large online community and file exchange portal where custom applications, code examples, and functions are freely shared.
- Short learning curve.

[1] http://www.mathworks.com/products/matlab/webinars.html

- The ability to test and modify algorithms without recompiling, and the ability to run code sections.
- The graphical output is designed for easy customisation of plots and figures.
- The ability to expand functionality further by the addition of toolboxes.

Matlab toolboxes provide additional algorithms, functions, and systems which can be used to aid development. The Neural Network Toolbox (v8.3) as well as the Statistics and Machine Learning Toolbox (v10.0) were used to design and develop the NN and KNN respectively.

3.3.3 Dataset

The dataset refers to the recordings obtained with which to train the classifiers in the system. It therefore makes up the training data which are used. In the subsections which follow, the steps used in obtaining the dataset and preparing it for use are explained.

Obtaining Recordings

Classification and recognition problems such as the topic of this project, require a comprehensive dataset on which to train. One of the project requirements however, is that *local* South African birds need to be identified by the final system. Clean recordings of bird calls and songs are difficult to obtain, as in most cases recording is done out in the field; there is often more than one type of bird in each recording. This, together with the requirement for local birds, made building a comprehensive dataset challenging. Thus it was decided that the dataset be compiled from eight different local birds, with a minimum of five recordings per bird. The exact number of recordings of each bird are further discussed in Chapter 4.

A number of online bird sound collections were consulted; the source chosen for building the dataset was Xeno-Canto[2]. The website hosted by Xeno-Canto is a community-based sharing platform which describes itself as "...a website dedicated to sharing bird sounds from all over the world. Whether you are a research scientist, a birder, or simply curious ..."

[2]http://www.xeno-canto.org

Xeno-Canto was chosen over other collections due to the following:

- Large recording set, with over 200,000 recordings.
- The ability to search for recordings by location.
- Use of recordings is governed by the Creative Commons License.

The Creative Commons License used by Xeno-Canto allows for the copying and redistribution of the recordings in any medium or format, as long as the following terms are adhered to:

- Appropriate credit must be given to the recordist.
- The material may not be used for commercial purposes.
- If the material has been modified, it may not be distributed.

Recordist attribution, as well as a link to the full license is found in Appendix B.

Preparing the Dataset

The recordings downloaded were all in MP3 format; it was believed that recordings in WAV format would be easier to work with. The recordings were therefore all converted into WAV format, with a sampling rate of 44.1 kHz and 16-bits per sample. The properties are summarised in Table 3.2 below. Furthermore, noise reduction and normalisation were performed on all recordings. All conversion and processing operations were performed using the 'conversion train' feature in the Audacity (v2.1.1) audio editor. This allows a set of modifications to be defined and then performed on any number of audio files.

Table 3.2: Properties of Audio Recordings in Dataset

Property	Specification
Format	WAV
No. Bits per Sample	16
Sampling Rate	44.1 kHz

3.3.4 Subsystems

The design will constitute three main subsystems which have been identified as: syllable extraction, feature extraction, and classification. A visual representation is given later in Figure 4.1 in Chapter 4. The subsystems were developed in this order, with minor modifications and optimisations being made a later stage where needed. A brief overview of the development process for each subsystem is given below.

Syllable Extraction

Syllable extraction is an extremely important aspect of the system as a whole, as discussed in Chapter 2. During this process bird chirps are identified in a recording, and then isolated and extracted. This ensures that classification is performed on identified syllables rather than parts of a recording which do not contain bird sounds.

It was decided that the syllable extraction method detailed in [27] was to be used. This method allows for automatic extraction of syllables from a recording. Certain threshold values can be changed to adjust the sensitivity of the algorithm employed by the extraction method. It was found that during syllable extraction of the training data, the variety of the bird calls and volumes in the recordings required tailoring the threshold values for each recording. Following this, erroneously extracted syllables were removed.

As the syllables are extracted, a corresponding table with the bird type is built, indicating the type of bird each syllable came from. This forms the *target matrix* which is used by the classification subsystem.

Feature Extraction

In order to classify the syllables, relevant features need to be extracted from each one. Initially, a single feature - the average frequency - was extracted for each syllable. This was to allow work to begin on the classifier and allow for incremental testing.

Feature extraction involves constructing a *feature matrix*. The feature matrix has as columns different features, and has feature values for each syllable as row entries. The process of selecting features, training the classifier, and evaluating performance was iterative. Features were added, removed, and combined to determine how classification

accuracy was affected. The features included in the final design are discussed in Chapter 4.

Classification

Two classification systems were set up in Matlab - the NN and KNN - using applications from the additional toolboxes mentioned previously. Refer to Section 2.2 for a description of these classification systems.

The NN was set up using the Neural Pattern Recognition application; it is started by typing 'nprtool' into the Matlab command line. Within this application, the feature and target matrices are chosen as inputs and targets respectively. Following this, the application auto-generates basic code which can then be modified to tailor the NN for the required task. The network parameters and settings chosen are discussed in Chapter 4.

The KNN was set up using the Classification Learner application; it is started by typing 'classificationLearner' into the Matlab command line. The classification learner requires that the targets be a single column appended to the feature matrix, thus a new feature matrix was easily obtained by appended such a column with values 1-8 representing the bird type of each syllable. This new feature matrix was imported into the classification learner, and the weighted-KNN was chosen as the classifier. Advanced options were set as followed: number of neighbours set to 10, distance metric set to Euclidean, distance weight set to squared inverse. Following this, the classifier was trained and exported.

Graphical User Interface

The Graphical user interface (GUI) was created using Matlab's graphical user interface design environment (GUIDE). GUIDE allows for simplistic visual and interactive elements to be added together to form a GUI. The desired elements were chosen and sized as needed, after which GUIDE produced executable code for the GUI. Functionality was added to the GUI by modifying the code in the 'callback' functions. Here, the subsystems mentioned above were integrated into the GUI such that the final system was complete.

3.4 Phase 4: Experimentation and Testing Plan

Experimentation and testing was performed on each of the subsystems as they were developed. This ensured that each subsystem was in a working state before development began on the next subsystem. This allowed for rapid optimisations and modifications to be made to previous subsystems in order to observe the effects. First, the final experiments that were conducted are discussed in the subsections below. Following this, the testing and minor experimentation performed during the development stage of the subsystems is discussed.

3.4.1 Experiments

The final experiments to be performed on the system as a whole are discussed below. Each of the experiments is to be performed for both the NN and the KNN, using 19 test recordings. It is important to note that these experiments are designed to test the system as a whole. Minor experiments were conducted to test aspects of the subsystems throughout the development stage; where necessary, these results are discussed in Chapter 4.

Experiment 1 - Timing

This experiment focuses on obtaining timing results for each of the individual subsystems in the B-CHIRP system. The timing results were captured multiple times and the average of each result was computed and tabulated. These can be found in Chapter 5. These results are used to judge system performance, and determine where bottlenecks may exist in the system. It is expected that the feature extraction process is likely to require the most computing time, and thus is the likely area where a bottleneck may occur.

Experiment 2 - Classification Accuracy per Syllable (Automatic Extraction)

This experiment aims to determine the classification accuracy by classifying on a *per-syllable* basis, while using the automatically extracted syllables. This result will be compared with that of Experiment 4 (which uses manually extracted syllables). It expected that the classification accuracy using automatic syllable extraction will be poorer than

that of manual syllable extraction.

Recall that classification accuracy is a measure of the the closeness of the systems prediction to the correct value. For this project, that is how good the system is at predicting the correct bird. The classification result for this experiment, as well as the remainder of the experiments is calculated from a confusion matrix. The example confusion matrix presented is Section 2.2 is reproduced in Figure 3.2 below to further illustrate how the classification results will be extracted.

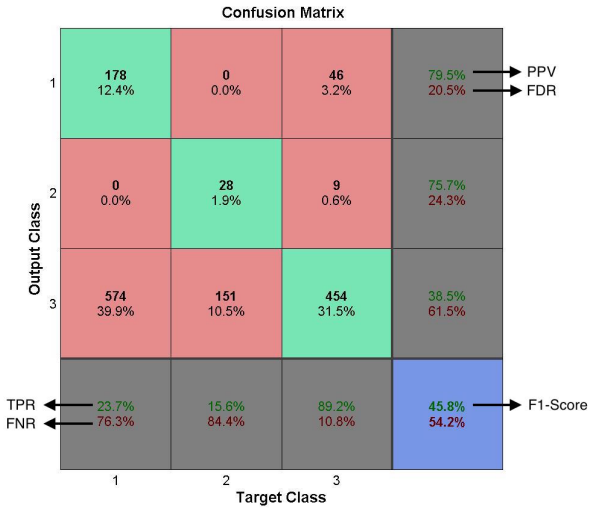


Figure 3.2: Example confusion matrix with important values highlighted.

An F1-Score, as shown in the figure above, is the overall classification result of a classification system, and is the score which will be used to asses the performance of the system. The TPR and PPV values can be used to draw insights into the classification accuracy of individual classes.

Experiment 3 - Classification Accuracy per Recording (Automatic Extraction)

This experiment follows on from Experiment 2 however, classification accuracy is calculated on a *per-recording* basis. The classification results per syllable in a recording are merged to form a classification result for the recording as a whole. This aims to highlight the system's accuracy which will be experienced by the user, as the final design classifies on a *per recording* basis.

It is expected that the classification accuracy calculated on a *per recording* basis, will be better than that calculated on a *per syllable* basis, for both the NN and KNN.

Experiment 4 - Classification Accuracy per Syllable (Manual Extraction)

This experiment aims to determine the classification accuracy by classifying on a *per-syllable* basis, while making use of the manually extracted syllables. By comparing this result with that of Experiment 2, the importance of correct syllable extraction can be inferred. It is expected that the classification accuracy will be higher than that of Experiment 2, for both the NN and KNN.

Experiment 5 - How Sampling Rate Affects Computation Time and Classification Accuracy

This experiment focuses on modifying the sampling rate of test recordings to determine how both the timing results and classification results are affected. It is expected that computation time is likely to decrease as a result of reducing the sampling rate. Furthermore, it is also expected that the classification accuracy will either remain unchanged or decrease. An increase in classification accuracy is considered to be unlikely.

3.4.2 Syllable Extraction

Testing and experimentation done while working on the syllable extraction involved manually listening to the extracted syllables and optimising the syllable extractor for each recording, as discussed above. This experimentation followed a qualitative testing procedure whereby the quality of the syllables extracted was determined by listening to each one individually.

Visual representations of syllables in recordings were obtained for certain test recordings to show the identified syllables. This is shown by plotting a spectrogram of a recording and a time-value plot of the recording with markers for each syllable identified. This is discussed further in Chapter 5.

3.4.3 Feature Extraction

A number of features were experimented with throughout the project. The features were evaluated by training the classifiers and testing them to determine how accuracy and

performance was affected by the addition and removal of certain features. Furthermore, LDA was used to determine the discriminative power of certain features, thereby giving an indication of their usefulness.

3.4.4 Classification

Classification is the end-goal of the system; a number of experiments were conducted to determine how classification accuracy and performance changes with a modification to other parts of the system. Automatic syllable extraction and manual syllable extraction were performed on test recordings to see how the classifier was affected. Furthermore, results were obtained on a *per syllable* basis as well as on a *per recording* basis. The sampling rate was also adjusted to determine how classification would be affected.

3.4.5 Graphical User Interface

The GUI utilises the previous subsystems and therefore only required acceptance testing to ensure that it worked as expected.

3.5 Phase 5: Data Collection and Analysis

This phase includes carrying out the experiments presented above and capturing the results, as well as analysis of the results. Timing results were obtained by using the `tic();` and `toc();` commands in Matlab. These were then tabulated to highlight the timing results of each subsystem.

The classification results for both the NN and KNN were collected in the form of confusion matrices. A confusion matrix plots the classification results of all test data, where each column of the matrix represents the actual class, and each row represents the predicted class. This allows for careful analysis of how the classifier performs for each class (or bird type in this case). Confusion matrices were obtained for each of the experiments mentioned above through use of the `plotconfusion` function in Matlab. This function takes as arguments the target matrix and the output matrix (containing the predictions of the classifier on the input data).

3.6 Phase 6: Conclusions and Recommendations

Careful analysis of the results included among other things, looking for common trends, looking for commonly misclassified birds, and determining bottleneck areas in the system. The analysis of the results was carried out with consideration to the project objectives and led to conclusions being drawn. The conclusions were formulated in the final phase of the project and are discussed in Chapter 6. Recommendations based on these conclusions were then considered and are presented along with possible future work in Chapter 7.

Chapter 4

System Design and Implementation

This chapter details the implementation of all modules and subsystems contained within the B-CHIRP system. It first presents an overview of the entire B-CHIRP system. It then describes the process of obtaining data for the system, and the implementation of the syllable extraction subsystem. It then presents the features considered for the feature extraction subsystem, as well as the process of evaluating these features. Finally, the implementation of the classification methods as well as the user interface is presented.

4.1 Overview of Design

The B-CHIRP system is made up of three main subsystems: syllable extraction, feature extraction, and classification. The subsystems are modular in their design, thereby allowing for them to be modified or upgraded, while maintaining functionality of the entire system as a whole.

All three subsystems are used during set up and training of the system, as well as during normal operation of the system. This concept, as well as the connection between the systems is shown in Figure 4.1 below.

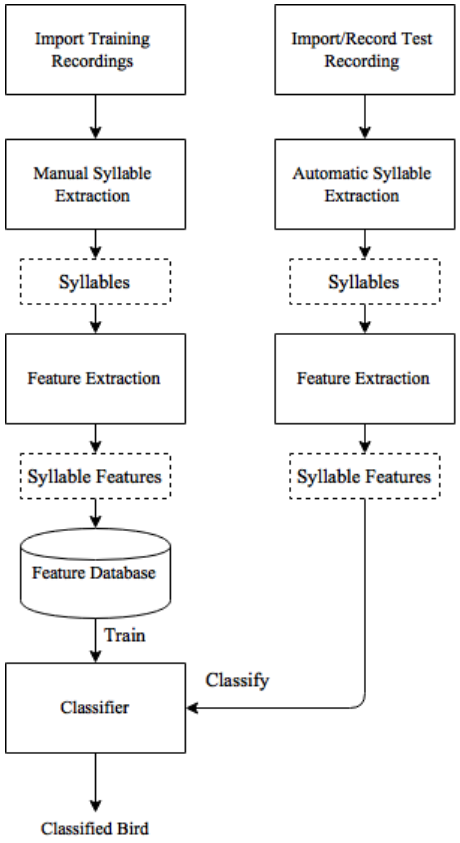


Figure 4.1: Block diagram showing the three main subsystems and their contribution to the system as a whole.

The training process depicted in the left of Figure 4.1 above is performed before the system can be used. Use of the system (shown on the right) is achieved through the use of a GUI.

Matlab was chosen as the programming language and IDE to be used throughout development for reasons already discussed in Section 3.3.2. All of the necessary functions and algorithms were implemented using Matlab, as well as the design and implementation of the GUI. The implementation of the main subsystems, as well as smaller modules contained in the system is discussed below.

4.2 Dataset

As discussed in Chapter 2, machine learning requires a training set of data. Testing the system requires test data; data which the system has not seen before. The dataset of recordings used throughout the development and experimentation phases consisted of 47

training recordings and 19 test recordings. Table 4.1 below presents the type of birds included in the dataset, as well as the number of training recordings and test recordings obtained for each bird type. There are 8 birds listed, and from here on, the bird type may be referred to by its name, or a number from 1 to 8.

Table 4.1: Training and Test Recordings

Bird Name	No. Training Recordings	No. Test Recordings
BlackSmith Lapwing	5	3
Blue Crane	7	2
Egyptian Goose	5	2
Grey Go-away-bird	5	3
Hadeda Ibis	7	3
Red-chested Cuckoo	6	2
Ring-necked Dove	6	2
South African Shelduck	6	2
Total	47	19

It is important to further clarify the distinction between training recordings and test recordings. Training recordings are those used to train the classifier; once the classifier has been trained on these recordings, they are no longer needed. Test recordings are recordings which the classifier has never seen before and are used to measure the classifier's accuracy.

The birds chosen to be included in the dataset were decided using two simple criteria:

- There are at least 7 usable recordings of the bird on Xeno-Canto.
- The bird is found in South Africa.

Browsing the Xeno-Canto database revealed many recordings of birds fitting the above criteria, however many of the recordings were of such a low quality, with high volumes of background noise, that they could not be included in the dataset. All of the recordings are raw field recordings taken by amateur recordists, and have been taken in parks, game reserves, and gardens. Most contain some calls from different birds. The majority of the calls though, belong to the main bird of the recording, making the recording usable. There is also background noise present in many of the recordings, to varying degrees. The background noise noted has been from wind, cars, and people talking.

4.3 Data Importing

Importing of recordings into Matlab is only required during the development phase of the B-CHIRP system. The implementation of this is discussed below. From a user perspective, importing recordings is not necessary, as the loading of a recording is facilitated through the use of an intuitive GUI (further discussed in Section 4.9).

In order to use the recordings during the development process, they first needed to be loaded into the Matlab 'workspace'. The Matlab workspace consists of variables and objects stored in memory, which are created and used by Matlab functions and code.

To import the recording data, a Matlab function called `importData.m` was created. This function takes no arguments and returns two variables: `allFiles`, which is a list of the file names of the recordings, and `soundData`, which is a cell array containing vectors which represent the recordings. The function prototype is as follows: `function [soundData,allFiles] = importData()`. By default, the function looks for the recordings in a folder 'Recordings WAV', which is alongside the folder containing all the Matlab scripts however, the ability to change this easily is facilitated by the `importData.m` function.

The file names of the recordings are stored in the format of 'Bird Name_X.wav', where X is the number of the recording for a particular bird. This file naming convention is important, as it allows the `allFiles` list to be populated; the `allFiles` list is later used to assign each syllable extracted to a particular bird type.

To import the audio recordings, first a list of all the audio file names is created using the `dir();` command. Each file in the list is then imported using the `audioread();` command and is then converted to mono by taking the sum of both stereo channels, divided by two.

4.4 Syllable Extraction

Syllable extraction forms an important part of the B-CHIRP system, as it refines the dataset into syllables to be used. From here on, the 'dataset' refers to the set of extracted syllables, rather than the set of recordings from which they were extracted.

As mentioned in Chapter 2, training a classifier on syllables is much more effective than training on a recording as a whole. This however, requires that syllables are correctly identified and extracted from a recording; which can be a difficult task. The use of an automatic syllable extraction method is presented below. Following this is a description of how such an extraction method was used to then manually extract syllables.

4.4.1 Automatic Syllable Extraction

Recall that syllable extraction is the process of analysing an audio waveform, detecting the location of a chirp, and extracting the chirp. The process behind this involves detecting spikes in volume in the waveform and extracting the portion of the audio surrounding these spikes.

The syllable extraction subsystem was implemented in a Matlab file `syllable_extraction.m`. Its function prototype is:
`function [syllableData,birdNumbers, fileNum] = syllable_extraction(soundData, Fs,allFiles,cutoff)`. The function takes in as arguments `soundData`, the cell array produced from importing the recordings; `Fs`, the sampling rate of the recordings; `allFiles`, the file list mentioned above; and `cutoff`, the threshold of the extraction algorithm.

The function returns `syllableData`, a cell array containing all of the extracted syllables; `birdNumbers`, a list of the same length as `syllableData`, which contains the bird number each syllable is from; and `fileNum`, a list describing which file a syllable was extracted from.

The function makes use of a function called `harmaSyllableSeg.m` [28] to perform the syllable extraction. This function is found on the Matlab Mathworks code-sharing website, and is free to use. Where needed, the code was modified in order to integrate it with the other functions in this subsystem. The `harmaSyllableSeg.m` function was developed according to the syllable extraction algorithm proposed by [27]. The algorithm is presented below:

Syllable Extraction Algorithm

1. Compute a spectrogram of a song segment using FFT. We denote a spectrogram matrix $S(f, t)$, where f represents frequency and t is time.

2. Repeat steps 3-7 for $n = 0, 1, ..., N-1$, where N is the total number of syllables extracted.

3. Find f_n and t_n, such that $|S(f_n, t_n)|$ is the maximum value in the spectrogram. This position represents the maximum amplitude position of nth sinusoidal syllable.

4. Store frequency parameter $w_n(0) = f_n$ and amplitude $a_n(O) = 20log_{10}|S(f_n, t_n)|$ $[dB]$.

5. Starting from $|S(f_n, t,)|$, trace the maximum peak of $S(f, t)$ for $t > t_0$ and for $t < t_0$ until $a_n(t - t_0) < a_n(0) - TdB$, were the stopping criteria T is typically 30 dB. It will determine how the sinusoidal syllable starts and ends at times t_s and t_e, respectively around the amplitude maximum t_0.

6. Store obtained frequency and amplitude trajectories corresponding to the nth syllable in functions $\omega_n(\tau)$ and $a_n(\tau)$, where $\tau = t_0 - t_s, ...t_0 + t_e$.

7. Set $S(f, [t_s, t_s + 1, ..., t_e]) = 0$ to delete the area of the nth syllable.

A portion of the `syllable_extraction.m` code is shown in Listing 4.1 below.

Listing 4.1: Excerpt of syllable_extraction.m

```
for k = 1:numFiles
  % Break up the current recording into sylables
  [x_syllables, x_Fs, x_S, x_F, x_T, x_P] = harmaSyllableSeg(soundData{k
     ,1},Fs,kaiser(512),128,1024,cutoff);

  for i = 1:length(x_syllables)
    % Prevents erroneously short syllables from being added to the dataset
    if (length(x_syllables(i).signal) > 100)

          count2 = count2 + 1; % index
          syllableData{count2,1} = x_syllables(i).signal; % input the
             syllable into the new soundData holder
          tempName = strtok(allFiles(k).name, '_'); % extract the bird name
              from the file name
          tempIndex = find(strcmp(birdTags, tempName)); % find the location
              in the list of birds (birdTags) - this will be the bird
             number (1-8)
          birdNumbers = [birdNumbers; birdTags{tempIndex, 2}]; % Populate
             the birdNumbers vector with numbers corresponding to the type
             of bird
          fileNum = [fileNum; k]; % Used to identidy which recording a
             syllable was extracted from.
    end
  end
end
```

The `syllable_extraction.m` script processes the recordings in a loop, and uses the `harmaSyllableSeg.m` script to extract the syllables. At the same time, the `syllable_extraction.m` script extracts the file name of the recording being processed, from the `allFiles` list. This is used to create a new list containing the bird number (1-8) that each syllable belongs to. Simultaneously, the same process is used to create the list `fileNum` which stores the recording number that each syllable is extracted from. This is later used to asses classification accuracy on a *per recording* basis.

The `cutoff` argument is used as the stopping criteria value T which is passed to the `harmaSyllableSeg.m` script. In practise, it was found that a value of 20 was most successful in extracting syllables automatically. Figures 4.2 and 4.3 below show the spectrogram representation of two test recordings, as well as the audio waveform with the identified syllables marked in red.

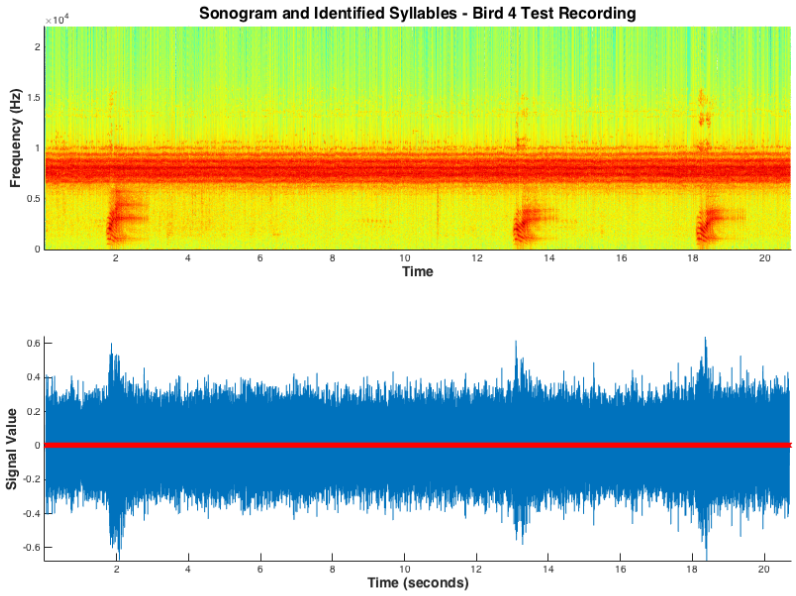


Figure 4.2: This figure shows the spectrogram representation and audio waveform representation with identified syllables of a test recording of Bird 4.

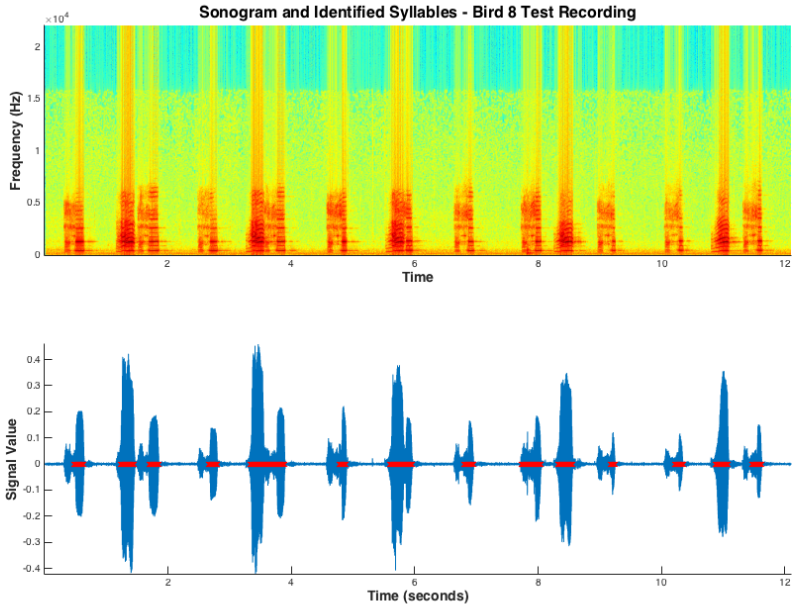


Figure 4.3: This figure shows the spectrogram representation and audio waveform representation with identified syllables of a test recording of Bird 8.

In Figure 4.2 above, three syllables can be seen in both the spectrogram and the audio waveform at times $t = 2, 14, 18$; the syllables have not been extracted correctly though. The solid red line in the audio waveform indicates that the entire recording has been regarded as one syllable. This erroneous extraction is likely due to the high volume of background noise, which is visible in both the spectrogram (as the thick orange bar), and the audio waveform (as noise around the zero line).

Figure 4.3 however, shows correctly identified syllables. It is also evident that there are very low levels of background noise, as the audio waveform is seen to return to near-zero values in between syllables.

4.4.2 Manual Syllable Extraction

Although the automatic syllable extraction process was successful for many of the recordings, it was found that the `cutoff` value would need to be varied for different recordings in order to maximise the syllable extraction accuracy for each recording. This led to the need for manual syllable extraction. Although the manual syllable extraction process uses the `harmaSyllableSeg.m` extractor, it is referred to as 'manual' due to the `cutoff` value needing to be manually changed for each recording. Furthermore, each extracted

syllable was listened to and it was decided whether it was acceptable for inclusion in the dataset. The entire process is depicted in Figure 4.4 below.

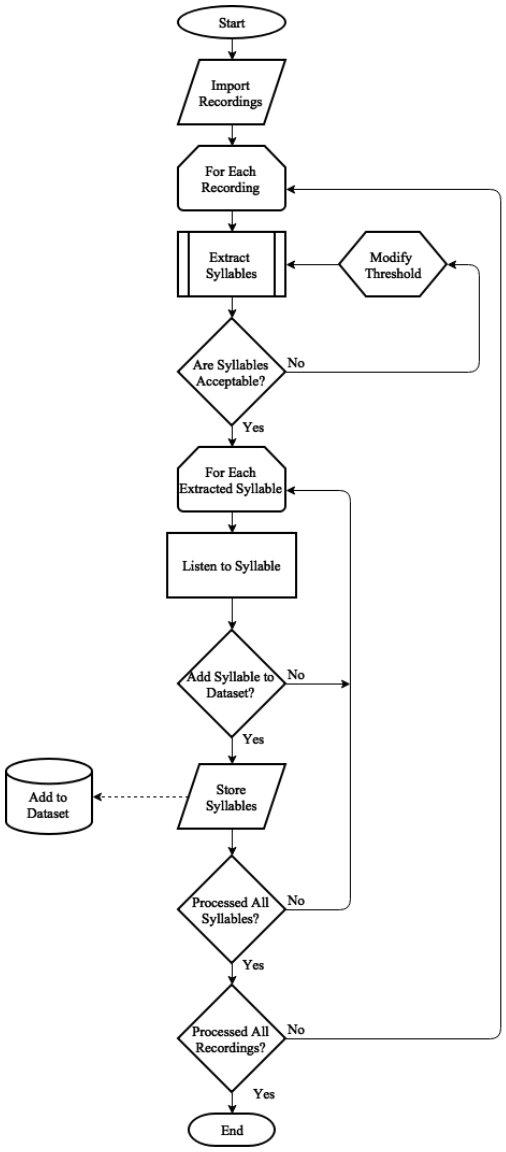


Figure 4.4: Flow chart showing the process of extracting syllables from a recording and building up the dataset.

The process is further explained using a case study involving the process of manually extracting syllables from a single audio recording. In this example, a recording of Bird 1, the BlackSmith Lapwing is considered. Figure 4.5 below shows a spectrogram of the recording, as well as the identified syllables shown in red. The `cutoff` value was kept at the default of 20.

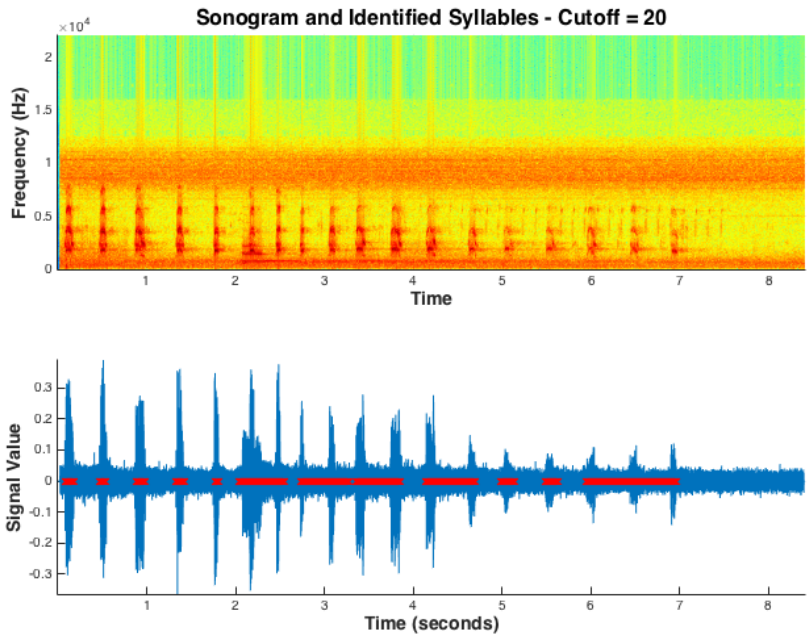


Figure 4.5: This figure shows the spectrogram and corresponing audio waveform with identified syllables marked in red, using a cutoff value of 20.

From the spectrogram above, 18 syllables can be seen. The automatic syllable extractor however, detected only 12 syllables. A portion of syllables in the middle of the recording, as well as at the end have been grouped together.

Figure 4.6 below shows the extracted syllables after the `cutoff` value was reduced to 18.

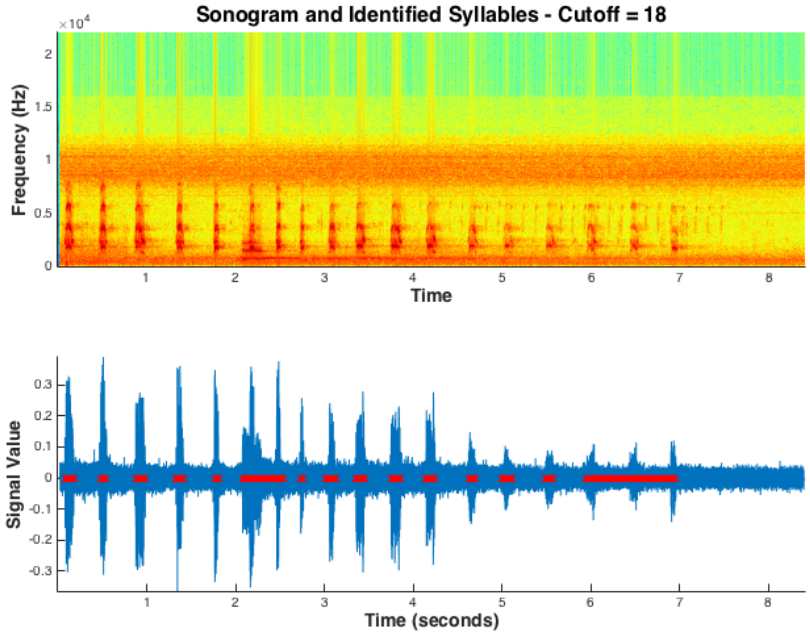


Figure 4.6: This figure shows the spectrogram and corresponing audio waveform with identified syllables marked in red, using a cutoff value of 18.

The extraction process identified 15 syllables, a marked improvement over the previous attempt. There remains however two sets of grouped syllables. At this point, the syllables were listened to individually, and it was decided whether a syllable be included in the dataset or not. Code Listing 4.2 below shows how this was achieved.

Listing 4.2: Except of Manual Syllable Extraction

```
%%  Manually select which syllables to add to dataset
% This code block can be used to listen to all extracted syllables and
% remove the clearly erroneous ones which have been extracted.

prompt = 'Inlude syllable in dataset? (1)';

for k = 1:length(temp_syllableData)
        sound(temp_syllableData{k,1},44100);
        x = input(prompt);
        if (x == 1) % then include
                % Add the syllables to the dataset
                syllableData = [syllableData; temp_syllableData{k,1}];
                birdNumbers = [birdNumbers; temp_birdNumbers(k)];
        end
end
```

After following the above procedure, both sets of grouped syllables were excluded from

the dataset, as the first contained a loud call from a different bird type, and the second contained soft calls which were unlikely to be grouped separately by further modifying the `cutoff` value. The final result was 13 extracted syllables which were added to the dataset.

4.4.3 Extracted Syllables

The number of syllables extracted during both the automatic and manual extraction processes are shown below for both the training and test recordings in Tables 4.2 to 4.5.

Table 4.2: Automatic Syllable Extraction of Training Recordings

Bird Name	No. Recordings	No. Syllables
BlackSmith Lapwing	5	198
Blue Crane	7	549
Egyptian Goose	5	249
Grey Go-away-bird	5	306
Hadeda Ibis	7	76
Red-chested Cuckoo	6	575
Ring-necked Dove	6	358
South African Shelduck	6	357
Total	47	2668

Table 4.3: Manual Syllable Extraction of Training Recordings

Bird Name	No. Recordings	No. Syllables
BlackSmith Lapwing	5	103
Blue Crane	7	81
Egyptian Goose	5	108
Grey Go-away-bird	5	25
Hadeda Ibis	7	54
Red-chested Cuckoo	6	335
Ring-necked Dove	6	53
South African Shelduck	6	43
Total	47	802

Table 4.4: Automatic Syllable Extraction of Test Recordings

Bird Name	No. Recordings	No. Syllables
BlackSmith Lapwing	3	49
Blue Crane	2	66
Egyptian Goose	2	268
Grey Go-away-bird	3	31
Hadeda Ibis	3	39
Red-chested Cuckoo	2	59
Ring-necked Dove	2	104
South African Shelduck	2	38
Total	19	654

Table 4.5: Manual Syllable Extraction of Test Recordings

Bird Name	No. Recordings	No. Syllables
BlackSmith Lapwing	3	34
Blue Crane	2	16
Egyptian Goose	2	73
Grey Go-away-bird	3	9
Hadeda Ibis	3	13
Red-chested Cuckoo	2	42
Ring-necked Dove	2	23
South African Shelduck	2	31
Total	19	241

For syllable extraction of the training recordings, the number of manually extracted syllables was just 30% of the number of automatically extracted syllables. For the test recordings, the number of manually extracted syllables was 37% of the number of automatically extracted syllables. These results are possibly due to the poor quality of the audio recordings, and the levels of background noise and other sounds present in the recordings, resulting in many erroneously extracted syllables which were then discarded. Furthermore, these results highlight the importance of accurate syllable extraction.

4.5 Proposed Features

Feature extraction is an important aspect in classification; it is used to extract information from the dataset which can be used to distinguish between different types of data found in the dataset. A number of feature extraction methods were implemented and tested during the development stage of the project. A portion of those features were selected for inclusion into the final design. Below, all of the features implemented are discussed. Following this, the feature selection process is described and the final features selected for inclusion in the final system are listed.

4.5.1 Band Power Filter Bank

The FFT function in Matlab returns the Fourier Transform of a signal, which shows the frequencies present in the signal, as well as the power of those frequencies. Band power refers to the average power in a particular frequency range. Different birds produce sounds at different frequencies, and it was thought that the average power over a number of frequency ranges could be used as a feature of a bird syllable. This lead to the simple design of a filter bank which divides the frequency spectrum of a syllable into a number of bands and calculates the percentage power (of the total power) within each band. Each of these values is considered a feature.

By overlaying the frequency plots of all syllables onto a single plot, it is possible to determine where the majority of the frequency content of the signals lies, as well as the highest frequency present in the signals. Through this observation, the range over which the filter banks should be placed in order to extract the most information is determined. Figure 4.7 below shows the frequency plots of all syllables.

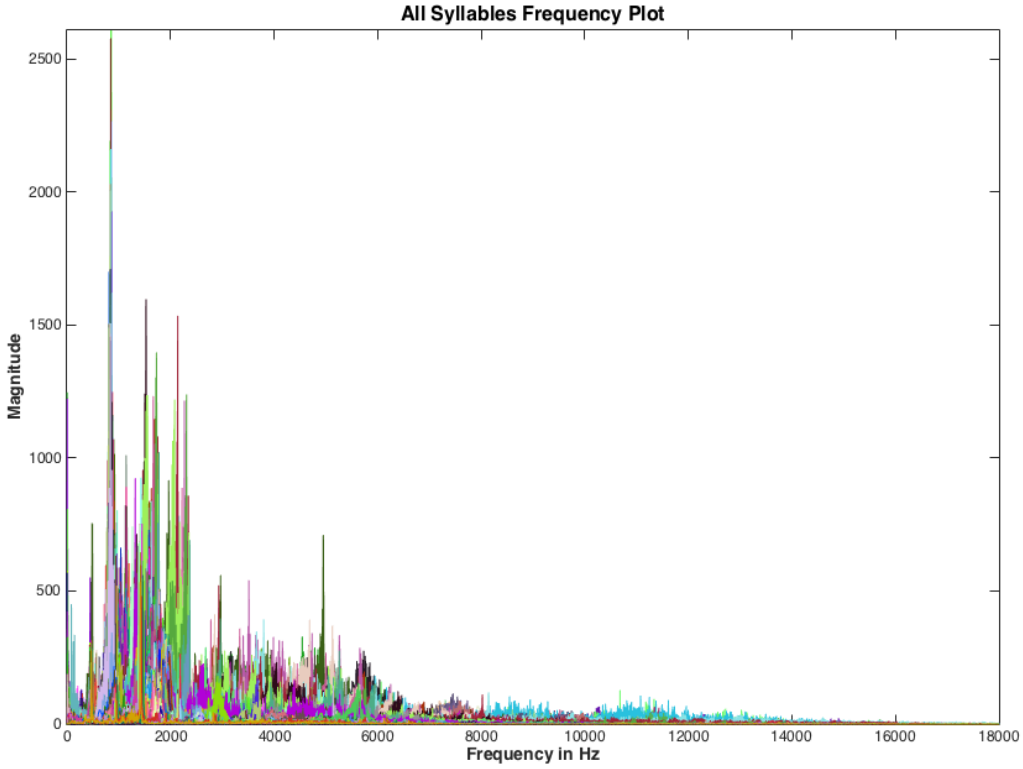


Figure 4.7: This figure shows the frequency components of *all* syllables extracted from the various recordings.

The majority of the frequency content is seen to lie between 0 Hz and 12 kHz, as seen in Figure 4.7 above. Thus, 12 kHz was taken as the end point of the band power filter bank.

The Matlab function `bandpower();` provides a simple means to calculating the band power in a given frequency range of a signal. This function was incorporated into a script `filter_bank.m`. The function prototype is: `function filter_bank_powers = filter_bank(syllable, numBPF)`, which takes in as arguments a single `syllable`, and the number of bands the frequency spectrum of the signal should be divided into, `numBPF`. The number of bands determines the number of features to be returned by the script in the vector `filter_bank_powers`.

4.5.2 Mel-Frequency Cepstrum Coefficients

As identified in Chapter 2, MFCCs are useful in audio classification. MFCCs were therefore proposed as a useful set of features. Calculating MFCCs is a complex task; due to

this, as well as the time constraints of the project, open-source code was used to calculate the MFCCs.

The Matlab script `melfcc.m` produced by [29] has been used to calculate the MFCCs. This script allows for numerous arguments to be set, all of which affect how the script processes the data, and thus affects the results. For the purpose of this project, only the maximum frequency (`maxfreq`) and number of cepstral coefficients (`numcep`) arguments were altered. The `maxfreq` sets the highest band edge of the mel filters and was set at 12 kHz, for the same reasons as the band power filter bank in Section 4.5.1. In order to determine the number of MFCCs to use, the value `numcep` was altered from 1 to 50. The MFCCs produced were then used as features in early implementations of the NN and KNN. It was found that little improvement in classification accuracy occurred beyond 30 MFCCs and thus this value was chosen to be used throughout the rest of development.

4.5.3 Spectral Centroid

The spectral centroid is a measure used in determining where the centre of mass lies within a frequency spectrum. The position of the spectral centroid affects the timbre of a sound. It was thought that the spectral centroid may be common to birds of the same type, and thus be a useful feature.

It is calculated as the weighted mean of the frequencies present in a signal. The Fourier Transform is used for determining the frequencies and magnitudes, and does so by dividing the signal up into bins. Mathematically, the spectral centroid is defined as:

$$SC = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1} x(n)}. \tag{4.1}$$

Where $x(n)$ is the frequency value of the nth bin, and $f(n)$ is the centre frequency of the nth bin.

The open-source Matlab function `SpecCentroid.m` was used to calculate the spectral centroid of each syllable.

4.5.4 Bandwidth

Signal bandwidth is the difference between a pair of upper and lower frequencies in a signal. For the purpose of extracting a feature from a syllable, the occupied bandwidth (OBW) is considered. The OBW is the bandwidth containing most of the total power in a signal. In this case, the OBW is calculated as the bandwidth containing a specified percentage of the total power.

OBW is calculated in Matlab using the built in function `obw();`, which computes the bandwidth containing 99% of the total power. This OBW value was calculated for each syllable.

4.5.5 Average Frequency

The average frequency, or mean frequency, of a frequency spectrum is calculated as the sum of the product of the magnitude and frequency, divided by the total sum of the magnitude. During development, the average frequency was thought to be included simply as a minor feature.

The average frequency for all syllables was calculated in Matlab using the `meanfreq();` function.

4.5.6 Frequency Peak Locations

The frequency spectrum of a signal contains numerous spikes in amplitude occurring where the magnitude of a particular frequency is larger than those of surrounding frequencies. Bird chirps of the same type are likely to exhibit peaks in similar locations in the frequency spectrum of their chirps.

The Matlab function `findpeaks();` can be used to find such spikes in the frequency spectrum. It takes in as arguments `NPeaks`, the number of peaks to return; `MinPeakDistance`, which forces the function to ignore peaks within a certain distance (in Hz); and `MinPeakProminence`, the minimum height a peak needs to be in order to be considered. The function was called with `NPeaks = 3`, `MinPeakDistance = 50`, and `MinPeakProminence = 40`. This returned three peaks to be used as features.

4.6 Feature Evaluation and Selection

In order to ensure reasonable computing times, as well as adequate performance, only a subset of the features mentioned above are included in the final design of the B-CHIRP system. Fisher's Linear Discriminant can be used as a measure for a feature's ability in discriminating between different classes. Here, the Fisher's Linear Discriminant value is calculated for the features spectral centroid, bandwidth, average frequency, and frequency peak locations. The higher the Fisher's Linear Discriminant value, the more discriminative power a feature has.

The Fisher-Score for these features is listed in Table 4.6 below.

Table 4.6: Fisher's Score for Features

Feature	Fisher's Score
Spectral Centroid	1.4036
Bandwidth	3.1998
Average Frequency	3.6513
Peaks	0.0032

The features spectral centroid, bandwidth, and average frequency all achieved scores which make them usable features. The frequency peak locations however, received an extremely low score. This is likely due to the noisy nature of the audio waveforms where many peaks are present, and thus the peaks detected do not correlate well with a particular bird type. The frequency peak locations was therefore not used as a feature in the final design.

For the band power filter bank, as well as the MFCC features, Fisher's Linear Discriminant does not perform well in determining the discriminative power of the features, as each of these includes numerous feature values. Alternative methods of determining the viability of these features is explored below.

4.6.1 Band Power Filter Bank

As mentioned in Subsection 4.5.1, the number of band power filters could be varied by altering the `numBPF` argument. An early implementation of the NN was used to test the performance achieved by using only the band power filter values as features. This imple-

mentation of the NN returned a `percentError` value, which indicated the classification error experience during testing of the NN. A lower value is better. The `numBPF` value was varied from 1 to 100, the NN trained and tested, and the `percentError` value stored. Figure 4.8 below shows the `numBPF` value plotted against the `percentError` returned by the NN.

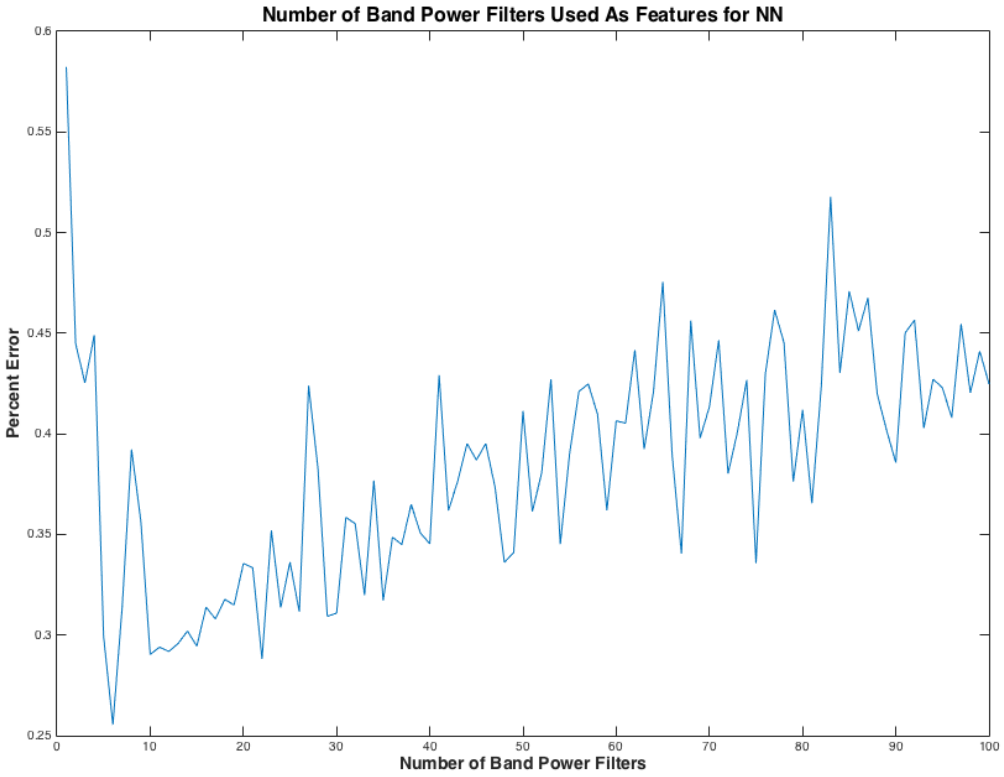


Figure 4.8: This figure shows the how the classification error of the NN varies with the number of band power filters used as features.

Evident in the figure above is the high degree of variation in the classification error. It's also important to note that for each `numBPF` value, the percent error was calculated 100 times in order to get a mean value. The execution time of calculating the band power filter values was also measured; calculating the band power filter values took over 30 seconds for some individual recordings. This was considered to be too long, as it would cause too big a delay between a user uploading a recording and the classifier returning a result. For these reasons, the filter bank powers were excluded from the final design.

4.6.2 Mel-Frequency Cepstrum Coefficients

To evaluate the performance of the MFCC features, the feature set was constructed and populated with MFCCs, where the number of MFCCs was varied from 1 to 40. For each case, both the NN and KNN performance was measured in terms of classification accuracy. It was found that 30 MFCCs produced satisfactory accuracy, and increasing the number of MFCCs beyond this yielded little improvement.

Furthermore, the success of [30, 31, 32] in bird classification while using MFCCs as features, further proves the benefits of using MFCCs. For this reason, as well as the adequate results obtained during early experimentation, 30 MFCCs were chosen to be included in the feature set.

4.6.3 Preparing Feature Matrix

Each of the features mentioned above were implemented in Matlab scripts. In order to provide an easy means to select which features to use, and then build a feature matrix, a function `extract_features.m` was created. The function prototype is: `function [featureMatrix] = extract_Features(syllableData, Fs , numBPF, numCep, varargin)`.

This function takes in as arguments `syllableData`, all the extracted syllables; `Fs`, the sampling rate of the audio files; `numBPF`, the number of band power filters to use; `numCep` , the number of MFCCs to calculate; and `varargin`, a collection of arguments used to select which features to include in the feature set. The complete `featureMatrix` is returned for use in the classifiers.

The argument `varargin` is used to turn features on by accepting a feature name as "`'feature name', 1`". For example, to select the MFCCs and average frequency, the function is called with "`'mfcc', 1, 'avgfreq', 1`".

Each feature which is selected creates its own matrix to hold the feature values it calculates. After all selected features have been extracted, the matrices are concatenated together to form the final `featureMatrix`. The final `featureMatrix` included 30 MFCCs, the spectral centroid, bandwidth, and average frequency. This resulted in a feature matrix with 33 columns, representing the 33 feature values.

4.7 Neural Network

The NN was created using the Matlab Neural Pattern Recognition application. A number of parameters can be set to customise the network to suit the needs of the classification problem. These parameters include training function, number of hidden layers, number of hidden nodes per layer, and pre/post-processing functions. These values were selected with consideration to the information obtained from the literature review, such as that described in [20, 22, 21].

The number of hidden layers was chosen to be 1. The number of hidden nodes in that layer was chosen to be 6. Figure 4.9 below shows the results of varying the number of hidden nodes, and how this affected the testing classification error. Little improvement is seen beyond 6 hidden notes, furthermore the more hidden nodes, the longer the training time. Thus 6 hidden nodes were chosen. The NN training method was chosen to be backpropagation as it one of the most common, stable and well-implemented approaches to training. Furthermore, it uses less memory than other approaches and is therefore suited to be used on the development machine (2010 MacBook Pro). The pre/post-processing performed on the data being fed into the NN was simply normalisation. This was achieved using the command `net.input.processFcns = {'removeconstantrows','mapminmax'};`, which also removes any duplicate entries.

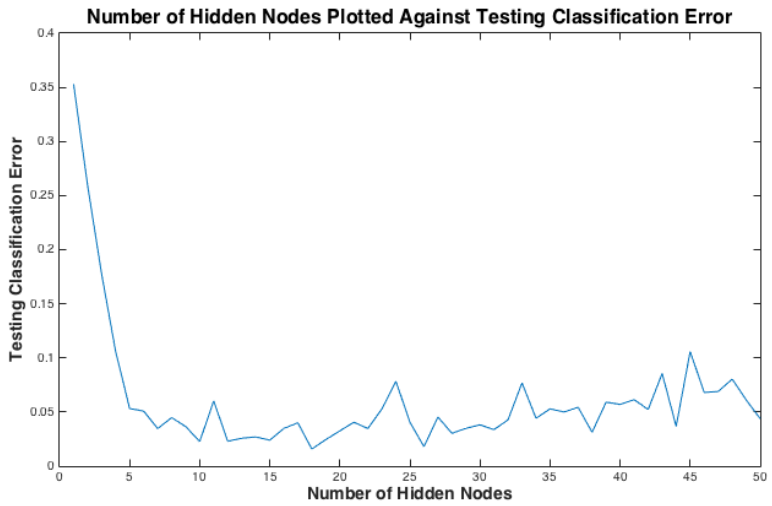


Figure 4.9: This figure shows the layout of the neural network, indicating the number of nodes in each layer.

The generalised visual representation of the network layout is shown in Figure 4.10 below.

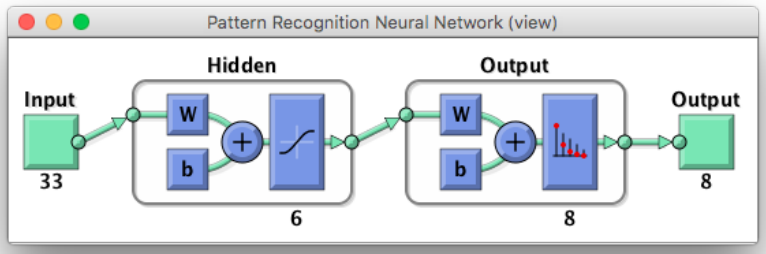


Figure 4.10: This figure shows the layout of the neural network, indicating the number of nodes in each layer.

The figure above shows how the network has 33 nodes in the input layer; these are the 33 features in the feature set. The hidden layer has 6 nodes. The 'w' and 'b' refer to the weights and biases respectively. These are computed and set automatically during network training. The activation function (shown to the right of the '+' symbol in the hidden layer) is shown to be a sigmoid function, the standard activation function used in neural networks. Further discussion of these concepts was discussed in Section 2.2. The output layer has 8 nodes, one for each bird type.

After the network was trained, it was exported as the file `net.mat`. This network was then used for all NN testing.

4.8 KNN Classifier

The KNN classifier used is a weighted KNN classifier, which was created using the Matlab Classification Learner application. The validation method used during the training process was 'cross validation', and the number of neighbours was set to 10, as this yielded the best performance and has been used in [24, 25].

The trained classifier was then exported as a `.mat` file. Preliminary results indicated that the KNN classifier outperformed the NN classifier. The results indicating this are presented in Chapter 5. The KNN classifier was therefore chosen as the classifier to be used in the final system.

4.9 Graphical User Interface

The GUI was developed as a means for users to interact with the system in an easy manner. All of the subsystems mentioned previously are used by the GUI to form the complete B-CHIRP system. A number of elements were added to the GUI to enhance the user experience, these were: A button to load a recording; a record button which flashes red when a recording is being made; an area showing a picture of the classified bird, as well as information about the bird; a plot of the audio recording waveform; and a bar graph showing the presence of different bird types detected in the recording

The preliminary GUI design was then presented to a small set of users for testing and recommendations. The users were then asked to fill out a short survey. The survey included three questions and a space to make recommendations. The survey, as well as the responses is found in Appendix A. The general consensus was that the user interface is easy to use and is functional. Furthermore the response time of the system was considered to be good. General comments and recommendations focused mainly on the need for a 'play' button to play a selected recording. This was seen as a necessary feature by the users and was therefore added.

The final GUI design is presented below in Figure 4.11.

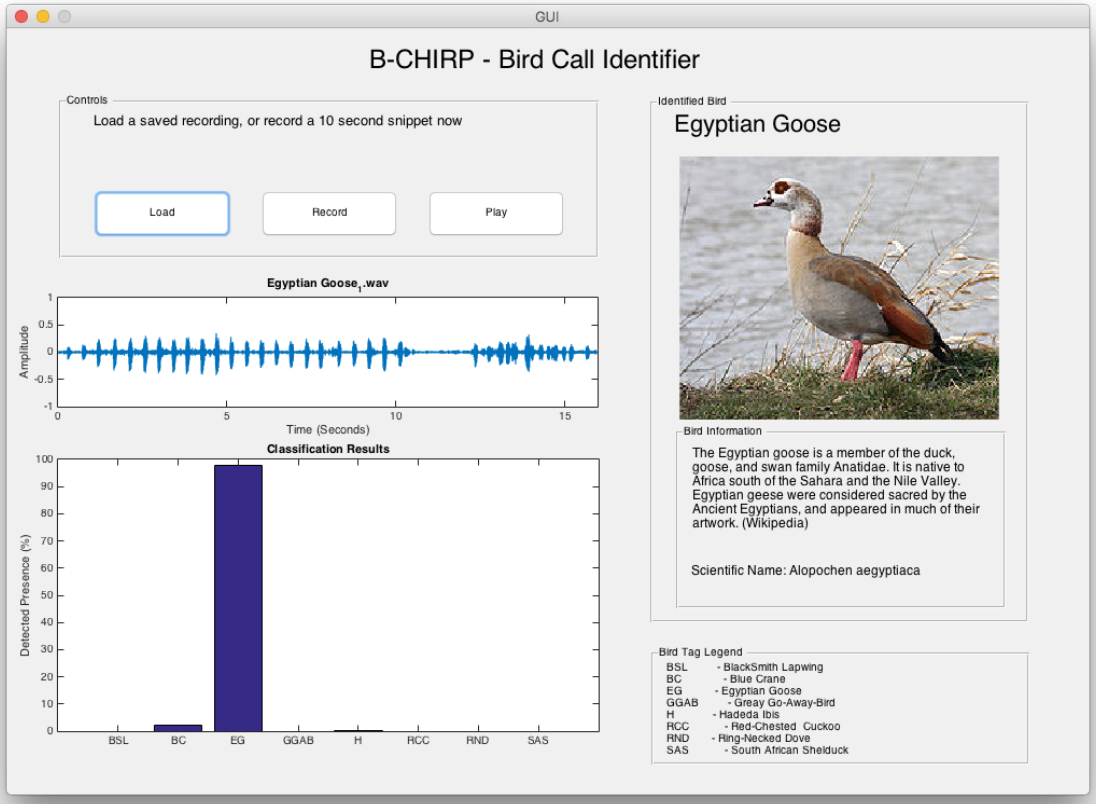


Figure 4.11: This figure shows the GUI after classification of a recording has taken place.

The 'Load' button opens up a file browser window, which allows the user to browse for a recording. The browser window allows the user to only upload '.MP3' and '.WAV' files, both of which the system is capable of handling. After selecting a file, it is automatically classified and the result is shown without any further interaction from the user. The 'Record' button uses the computer's built in microphone to record audio, and as with the 'Load' button, the classification result is shown automatically. The 'Play' button will play the the last audio file uploaded, or the last recording made.

4.10 Re-Use of Code and Maintenance Design

Code re-use is an important aspect of any software design, as it aims to increase the longevity of the project and allows for future work on the design. Throughout the development stage of the project, consideration was given to maintenance design and code re-use. All subsystems in the design are modular, allowing them to be upgraded or swapped out for a newer subsystem. Extensive use of code comments has been used to

explain the input/output requirements of each function, as well as the format of the data which is needed by each function. Furthermore, most of the functions have been made generic, allowing the entire development and training process to be performed using other types of audio recordings. Finally, a README file has been included with the collection of code as a guide for future developers.

Chapter 5

Results and Analysis

This chapter presents the results obtained from the five experiments detailed in Chapter 3. The chapter is broken into sections, each relating to one of the five experiments. The results are then analysed and discussed.

5.1 Experiment 1 - Timing

This experiment focused on obtaining timing results for each of the individual subsystems in the B-CHIRP system. The timing results presented in Figure 5.1 below were obtained using Matlab's built-in `tic();` and `toc();` commands. The timing results are shown for the system processing all 19 test recordings.

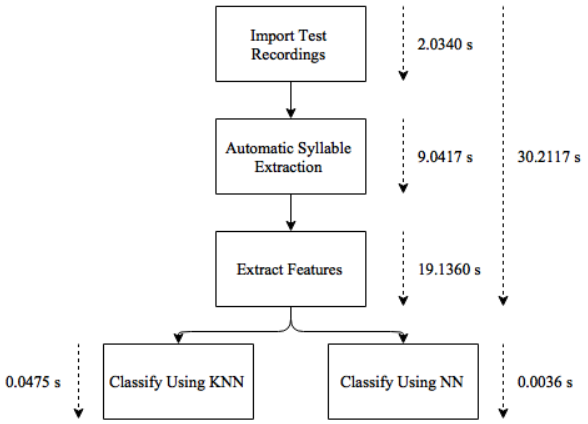


Figure 5.1: Timing diagram showing the time taken to complete each stage in the classification program, using all 19 test recordings.

The timing bottleneck occurs during syllable extraction and feature extraction This confirms the expected result. These two processes are the most computationally intensive; such a result is therefore expected. The total time taken to prepare all 19 test recordings for classification is 30,2117 s. This results in an average preparation time of 1,59 s per recording.

Following preparation, classification is completed in 47.5 ms using the KNN, and 3.6 ms using the NN. From a user perspective, these times are almost indistinguishable and, as such, the classification time is not considered when selecting which classifier to use in the final system. Rather, only the classification accuracy is considered.

5.2 Experiment 2 - Classification Accuracy per Syllable (Automatic Extraction)

This experiment aimed to determine the classification accuracy by classifying on a *per-syllable* basis, while using the automatically extracted syllables. The results of this experiment were obtained by passing all 19 test recordings through the entire system for classification. This involved importing the recordings, automatically extracting the syllables, extracting the features, and classifying using both the NN and KNN. The classification was done on a *per syllable* basis.

5.2.1 Experiment 2 - NN Case

Figure 5.2 below shows the results of Experiment 2 using the NN for classification.

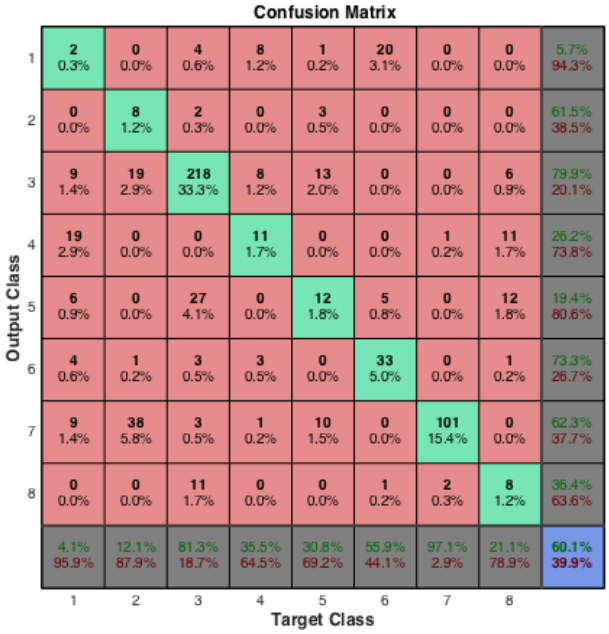


Figure 5.2: Confusion matrix showing the NN *per syllable* classification accuracy on all test recordings (using automatic syllable extraction).

An overall accuracy of 60.1% was achieved. This relatively high accuracy can be explained by the both bird 3 and bird 7 achieving a high TPR of 81.3% and 97.1% respectively. The number of syllables extracted for these birds using automatic syllable extraction was large (See Table 4.4 in Chapter 4 for exact number), and as such, the classification accuracy of these birds will affect the overall result significantly.

Birds 1, 2, and 8 experienced a high degree of misclassification, as indicated by the high FNR rate. Birds 1, 3, and 5 experienced a high FDR rate, indicating that other birds are often misclassified as these birds.

5.2.2 Experiment 2 - KNN Case

Figure 5.3 below shows the results of Experiment 2 using the KNN for classification.

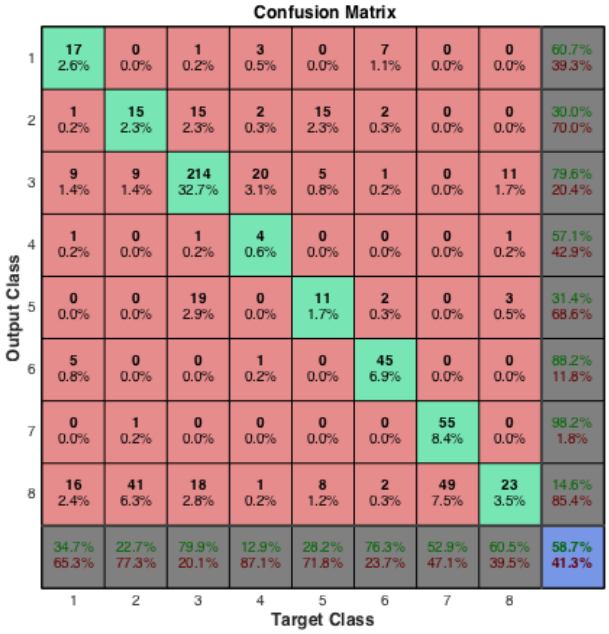


Figure 5.3: Confusion matrix showing the KNN *per syllable* classification accuracy on all test recordings (using automatic syllable extraction).

An overall accuracy of 58.7% was achieved. Again this can be explained by certain birds having more syllables extracted and achieving good classification and thus increasing the overall classification accuracy. Birds 1, 2, and 4 experienced a high degree of misclassification, as indicated by the high FNR rate. Birds 5 and 8 experienced high FDR rate, indicating that other birds are often misclassified as these birds.

5.2.3 Discussion - Experiment 2

Overall, the NN slightly outperformed the KNN in this experiment. Both classifiers struggled to classify Birds 1 and 2, and more often than not, classified them as other birds. Furthermore, other birds were often misclassified as Bird 5 in both classifiers.

The NN showed a higher degree of FDR among the birds than the KNN, indicating that it produces more false positives than the KNN. The KNN however showed higher FNR rates than the NN, indicating that it produces more false negatives than the NN.

5.3 Experiment 3 - Classification Accuracy per Recording (Automatic Extraction)

This experiment followed on from Experiment 2 however, classification accuracy was calculated on a *per-recording* basis. The results of this experiment were obtained by passing all 19 test recordings through the entire system for classification. Both the NN and KNN classifiers were however, set to classify on a *per recording* basis. This was achieved by classifying each automatically extracted syllable in a particular recording, and then finding the cumulative prediction for that recording, taking into account the prediction for all syllables in the recording.

5.3.1 Experiment 3 - NN Case

Figure 5.4 below shows the results of Experiment 3 using the NN for classification.

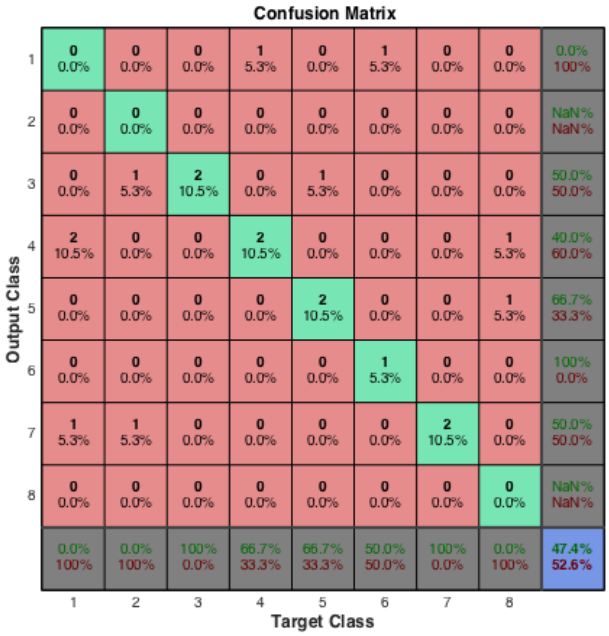


Figure 5.4: Confusion matrix showing the NN *per recording* classification accuracy on all test recordings (using automatic syllable extraction).

An overall accuracy of 47.4% was achieved. This is significantly lower than the *per syllable* classification accuracy achieved by the NN. None of the recordings of Birds 1, 2, and 8 were classified correctly. This result did not satisfy the expected result proposed

in Section 3.4.1 that classification accuracy would improve when classifying on a *per recording* basis.

5.3.2 Experiment 3 - KNN Case

Figure 5.5 below shows the results of Experiment 3 using the KNN for classification.

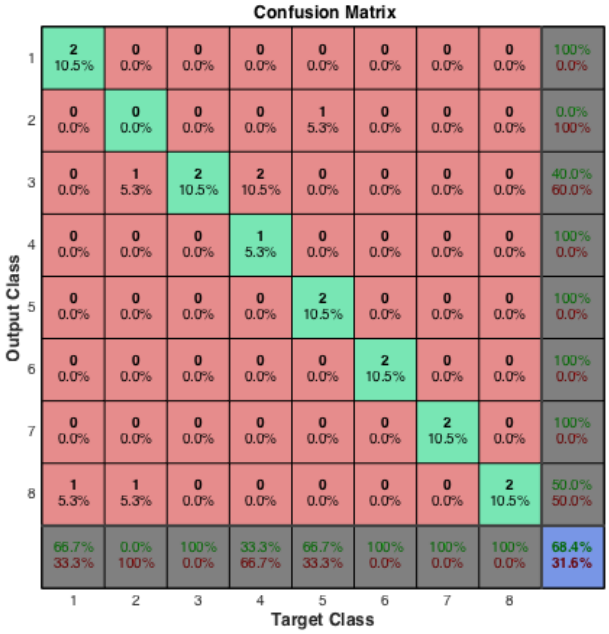


Figure 5.5: Confusion matrix showing the KNN *per recording* classification accuracy on all test recordings (using automatic syllable extraction).

An overall accuracy of 68.4% was achieved, which relates to a 9.7% increase in classification accuracy from the *per syllable* classification. This result satisfies the expected outcome presented in Section 3.4.1. Only Bird 2 was completely misclassified, with the majority of other birds being classified correctly.

5.3.3 Discussion - Experiment 3

The KNN performed significantly better than the NN in this experiment. Taking into account the FDR and FNR results from Experiment 2, the KNN outperformed the NN due to it having a lower FDR among most of the birds. A high FDR indicates many

false positive predictions - where the system is sure that a recording is of a certain bird, but in fact is not. Evident from the high overall classification accuracy of the KNN, a higher false negative rate rather than a higher false positive rate results in better overall classification. Minimising both of these rates however, would yield the best results.

5.4 Experiment 4 - Classification Accuracy per Syllable (Manual Extraction)

This experiment aimed to determine the classification accuracy by classifying on a *per-syllable* basis, while making use of the manually extracted syllables. The results of this experiment were obtained by: manually extracting syllables from each test recording, according to the process discussed in Section 3.3.4 in Chapter 3; and passing these syllables through the system for classification using both the NN and KNN.

5.4.1 Experiment 4 - NN Case

Figure 5.6 below shows the results of Experiment 4 using the NN for classification.

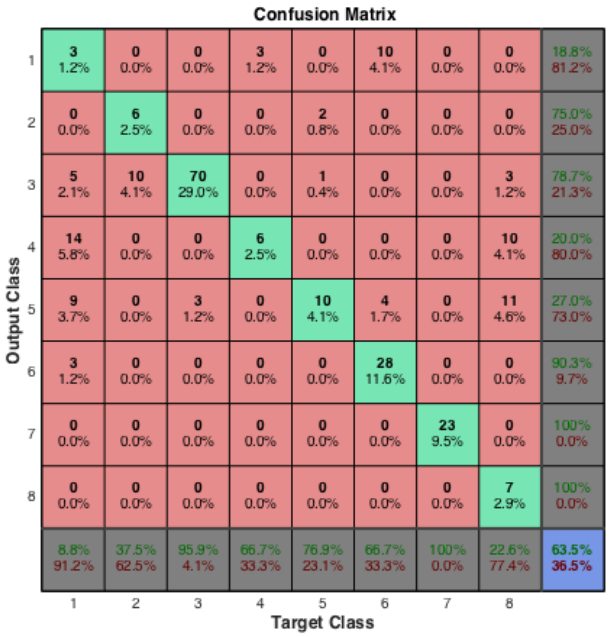


Figure 5.6: Confusion matrix showing the NN *per syllable* classification accuracy on all test recordings (using manual syllable extraction).

An overall accuracy of 63.5% was achieved, corresponding to a 3.4% increase in classification accuracy over automatic syllable extraction. Again Birds 1, 2, and 8 experienced a high degree of misclassification, as indicated by the high FNR rate. Birds 1, 4, and 5 experienced a high FDR rate, indicating that other birds are often misclassified as these birds.

5.4.2 Experiment 4 - KNN Case

Figure 5.7 below shows the results of Experiment 4 using the KNN for classification.

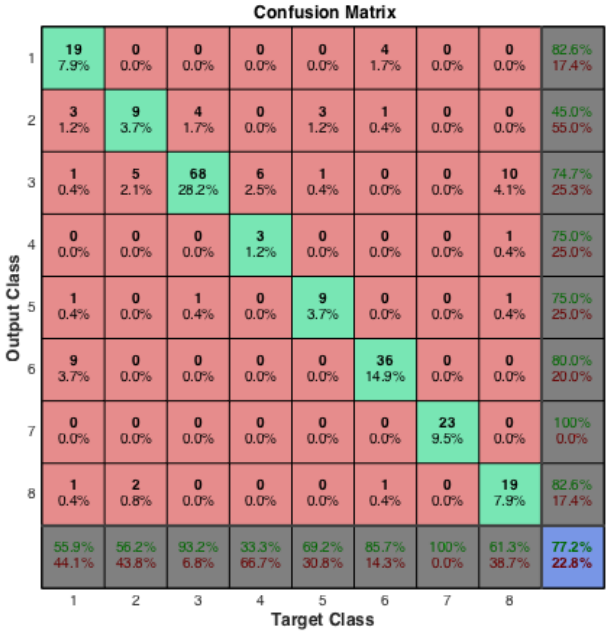


Figure 5.7: Confusion matrix showing the KNN *per syllable* classification accuracy on all test recordings (using manual syllable extraction).

An overall accuracy of 77.2% was achieved, corresponding to a 17.1% increase in classification accuracy over automatic syllable extraction. Only Bird 4 experienced a high FNR rate, indicating it was often misclassified as another bird.

5.4.3 Discussion - Experiment 4

Classification accuracy improved for both the NN and KNN, in comparison to the accuracy achieved using automatic syllable extraction. This further highlights the importance of correct syllable extraction. As in Experiment 2, the NN struggled to classify Birds 1, 2, and 8, often misclassifying these birds as other birds. The KNN also struggled with Bird 4, as it did in Experiment 2.

A pattern clearly emerges, and indicates that the NN and KNN have both strengths and weaknesses in classifying certain birds. Ultimately however, the KNN outperforms the NN on overall classification accuracy.

5.5 Experiment 5 - How Sampling Rate Affects Computation Time and Classification Accuracy

This experiment focused on modifying the sampling rate of test recordings to determine how both the timing results and classification results are affected. The sampling rate of an audio file affects how many samples are stored for each second of the recording; a higher sampling rate results in more data being stored to represent the information present in the recording. In many signal processing systems, less data results in faster computation time, as the system has less data to work through. Data reduction can however lead to the required information not being represented fully, and thus reduced accuracy of the system. In this experiment, the effect of reducing the sampling rate was observed.

5.5.1 Sample Rate Reduction - Using a Single Recording

The first stage in the experiment involved running a single recording through the classification system to obtain a classification result, as well as computation times. This was followed by re-sampling the recording at 3/4 of the original sampling rate, and then passing the recording through the system again. The same was done for the recording sampled at 1/2 the original sampling rate. The original sampling rate of all test recordings is 44.1 kHz.

The test recording used for this experiment was one of Bird 8 - The South African

Shelduck. Only the KNN classifier was used due to its increased performance over the NN classifier, as seen in the previous experiments.

Original Sampling Rate

In this phase of the experiment, the sampling rate was not adjusted. The timing results for processing the single recording are shown in Table 5.1 below.

Table 5.1: Original Sampling Rate

Processing Stage	Mean Time (s)
Syllable Extraction	0.2615
Feature Extraction	0.3184
Classify (KNN)	0.0068
Total	0.5867

The classifier predicted the correct bird with a prediction accuracy of 76.08%.

Three-quarters Sampling Rate

In this phase of the experiment, the sampling rate was reduced to 3/4 of the original sampling rate. The timing results are shown in Table 5.2 below.

Table 5.2: 3/4 Sampling Rate

Processing Stage	Mean Time (s)
Syllable Extraction	0.2132
Feature Extraction	0.2244
Classify (KNN)	0.0061
Total	0.4437

The classifier predicted the correct bird with a prediction accuracy of 81.61%.

Half Sampling Rate

In this phase of the experiment, the sampling rate was reduced to 1/2 of the original sampling rate. The timing results are shown in Table 5.3 below.

Table 5.3: 1/2 Sampling Rate

Processing Stage	Mean Time (s)
Syllable Extraction	0.1244
Feature Extraction	0.1740
Classify (KNN)	0.0059
Total	0.3043

The classifier predicted the correct bird with a prediction accuracy of 85.14%.

A reduction in computation time is observed when the sampling rate is reduced; halving the sampling rate reduced the computation time by 47.57%, just under half. Furthermore, for this recording, the prediction accuracy improved with a reduction in sampling rate. One possible explanation for this is the sampling rate reduction may act as a low-pass filter on the recording, which in this case improved classification accuracy. It cannot be said however, that such an affect would be observed for all recordings.

5.5.2 Sample Rate Reduction - Using All Test Recordings

Based on the results presented above, the classification part of the experiment was carried out for all test recordings, re-sampled at half the original sampling rate. The classification was again performed using only the KNN classifier, and was performed on a *per recording* basis. The *per recording* basis was chosen here as it is the classification result which the final system uses, and that which a user of the system would be presented with. Figure 5.8 below shows the results.

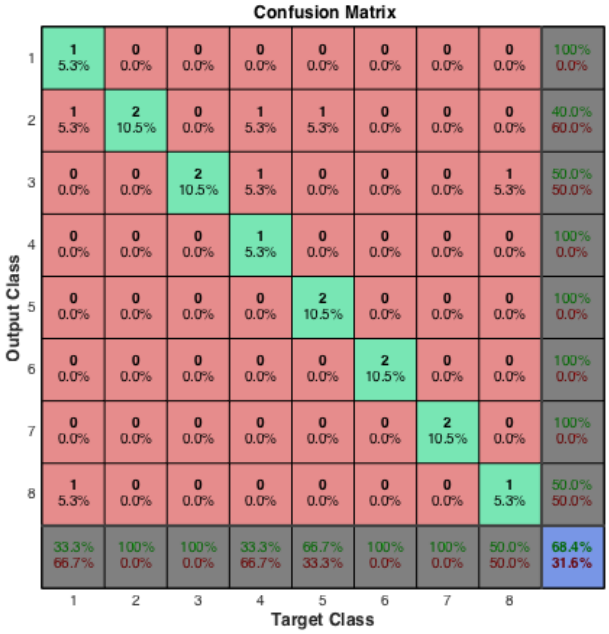


Figure 5.8: Confusion matrix showing the KNN *per recording* classification accuracy on all test recordings (using automatic syllable extraction) after being sampled at half the original sampling rate.

The overall accuracy remained unchanged at 68.4%, with the only difference being one less correct classification for Birds 1 and 8, and two more correct classifications for Bird 2. Again this might be explained by the reduction in sampling rate acting as a low-pass filter. This type of filtering may affect different types of bird calls in different ways, and thereby cause improved accuracy in some birds, while reducing the accuracy in others.

The results of this experiment show that overall, the system can perform just as well using audio which is sampled at 22.05 kHz (half the original sampling rate), compared to using the original sampling rate. Furthermore, by halving the sampling rate, the computation time is almost halved. Future designs could therefore be made more efficient by using a lower sampling rate.

Chapter 6

Conclusions

This project set out to obtain recordings of a small number of birds found locally in South Africa; to compare two methods for classifying birds based on their sounds; and incorporate the better performing method into a simple user application. This chapter presents succinct conclusions based on the objectives of this project, as well as conclusions which are drawn from the results and analysis in Chapter 5. First, conclusions relating to the results are discussed and, following this, conclusions relating to the objectives of the project are presented.

6.1 Experiment Conclusions

6.1.1 Timing

The system performed with reasonable computation times for smaller batches of recordings to be classified. The average time to prepare a recording for classification was 1.59 s. For the purpose of this project, a delay in classification of less than 2 seconds was achieved, and is thought to be adequate for all users of the system. If the timing were to be improved further, the syllable extraction and feature extraction subsystems should be focused on and optimised, as the timing bottleneck occurred in these subsystems.

6.1.2 Classification Accuracy

The final system, which classifies on a *per recording* basis achieved a high classification accuracy of 68.4% by implementing the KNN classifier which outperformed the NN classifier. Although the KNN classifier appeared to have a higher false negative rate than the NN classifier, it had a lower false positive rate. A higher false positive rate is likely to result in more classification errors, as the system is more likely to predict the incorrect bird. Therefore, to classify birds based on their call with a limited dataset, the KNN classifier is more suitable.

Certain bird types were misclassified more often than other bird types and had higher false negative rates, these were Birds 1 and 2 (The BlackSmith Lapwing and Blue Crane). This indicates that the current feature set may be limited in its discriminative ability for these birds. Bird 4 (The Grey Go-Away-Bird) appeared to have a higher false positive rate than other birds, indicating that the system may be biased towards it. This could be due to the features for this bird covering a wide range, therefore increasing the chance that other birds may be misclassified as it.

6.1.3 Syllable Extraction

Syllable extraction was seen to have a large effect on classification accuracy. The importance of correct syllable extraction cannot be overstated. This is evident from only 30% of the automatically extracted syllables being included in the original training dataset; the rest were discarded as they were erroneous. The difficulty in extracting syllables from recordings which contain multiple bird species, overlapping bird chirps, and lots of background noise, became increasingly apparent during development. Only a small portion of the syllables automatically extracted during this project were extracted correctly. Automatic syllable extraction is therefore considered to be the most difficult aspect to bird call recognition, and requires improvement in the future.

6.1.4 Feature Extraction

A small number of features were considered and evaluated during this project. Feature extraction becomes increasingly important when smaller datasets are considered. The feature extraction techniques used in this project were shown to be effective for use in a small classification system, resulting in a final classification accuracy of 68.4%.

6.1.5 Sampling Rate

The sampling rate was reduced to 3/4 the original sampling rate, and then further reduced to 1/2 the original sampling rate. The reductions in the sampling rate were seen to have no effect on the overall classification accuracy of the system however, the classification accuracy of certain individual birds was slightly affected. Reducing the sampling rate to half the original sampling rate resulted in a 47.57% reduction in computation time. For these reasons, the use of a 44.1 kHz sampling rate was unnecessary; a sampling rate of half this would suffice.

6.2 Overall Conclusions

The final system was able to classify 8 local birds by their call, with a classification accuracy of 68.4%. The objectives presented in Chapter 1 were successfully met in the following manner:

1. A collection of local bird recordings for use within the project was obtained.
2. Two classification methods were implemented and compared.
3. The better performing method was incorporated into a fully functional program.
4. A simple and intuitive graphical user interface was designed for the program.
5. It was determined that reducing the sampling rate had no effect on overall classification accuracy, but did improve system computation time.

Furthermore, the system followed a modular design approach, paving the way for future work. The B-CHIRP application produced during this project is therefore considered a success.

Chapter 7

Recommendations

The following recommendations are based on the results and conclusions presented in the previous chapters. Methods for increasing both performance and accuracy in the system are suggested.

7.1 Timing

A faster and more accurate syllable extraction algorithm should be implemented in order to reduce the timing bottleneck. Furthermore, a reduced feature set should be considered. Methods for feature set reduction were not considered in this report however, methods such as singular value decomposition (SVD) and principal component analysis (PCA) should be considered as feature reduction methods to be implemented in future work in order to further reduce computation time.

7.2 Syllable Extraction

In this project, an energy-based time-domain syllable segmentation method was used to extract syllables. It was discovered that this method was not suitable for recordings with lots of background noise, and with multiple bird species present. A time-frequency syllable segmentation method such as that proposed by [33] may offer better results at correctly identifying and extracting syllables, and should be implemented in future work.

7.3 Feature Extraction

A number of birds were misclassified by the system more often than others (The Black-Smith Lapwing and Blue Crane). In order to potentially improve classification accuracy and reduce misclassifications, more features should be evaluated. Recommended features to be included are: location of where the recording was captured, and the date and time of the recording. These features could increase classification accuracy as certain birds migrate during the year, therefore they will be more, or less, likely to be in certain locations at a particular time.

7.4 Sampling Rate

Future implementations should use a sampling rate of 22.05 kHz as this will improve system computation time, while maintaining the classification accuracy. Attempts to reduce the sampling rate further are likely to result in aliasing of the audio signal and reduced accuracy.

7.5 Future Work

This project succeeded as a proof-of-concept project and provides a valuable starting point for future work into the important field of bird classification. Further iterations in the design could yield improved results and benefit both the bird-watching community and ornithologists alike.

References

[1] S. R. Alten, *Audio Basics*. Boston, MA: Uhl, Lyn, first edit ed., 2012.

[2] R. Beason, *What Can Birds Hear?* USDA Wildlife Services, National Wildlife Research Center, 2004.

[3] F. Rumsey and T. McCormick, *Sound and Recording*. Focal Press, 2009.

[4] S. Fagerlund, P. U. K. Laine, and D. T. A. Härmä, "Automatic Recognition of Bird Species by Their Sounds," *Department of Electrical and Communications Engineering\nLaboratory of Acoustics and Audio Signal Processing*, vol. Master of, 2004.

[5] E. a. Brenowitz, D. Margoliash, and K. W. Nordeen, "An introduction to birdsong and the avian song system. Special Issue: The Neurobiology of Birdsong, Brenowitz E.A., Margoiash,D., Nordeen,K.W. (eds)," *Journal of Neurobiology*, vol. 33, pp. 495–500, 1997.

[6] C. K. Catchpole and P. J. B. Slater, *Bird Song - Biological Themes and Variations*. No. 1, New York: Cambridge University Press, New York, second ed., 2014.

[7] M. S. university) Bosi and R. E. T. B. G. Goldberg, *introduction to Digital Audio Coding and Standards*. Dordrecht: Kluwer Academic Publishers, 2003.

[8] J. Sueur, "A very short introduction to sound analysis for those who like elephant trumpet calls or other wildlife sound Contents," *Muséum national d'Historie naturelle*, pp. 1–17, 2014.

[9] H. B. U. Shatkay, "The Fourier Transform - A Primer." 1995.

[10] K. L. Stratos, "The fast fourier transform and its applications.".

[11] R. N. Bracewell, "The Fourier Transform and its Applications," *Transform*, vol. 42, pp. 1–4, 1986.

[12] L. V. Barbosa, "Fourier transform time and frequency domains," 2013. https://commons.wikimedia.org/wiki/File:Fourier_transform_time_and_frequency_domains_(small).gif.

[13] AlwaysLearn, "A DFT and FFT TUTORIAL." http://www.alwayslearn.com/dftandffttutorial/DFTandFFT_BasicIdea.html.

[14] M. Sahidullah and G. Saha, "Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition," *Speech Communication*, vol. 54, no. 4, pp. 543–565, 2012.

[15] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[16] B. Logan, "Mel Frequency Cepstral Coefficients for Music Modeling," *International Symposium on Music Information Retrieval*, vol. 28, p. 11p., 2000.

[17] A. Y. L. Smola, S. V. N. D. o. S. Vishwanathan, and C. S. P. University), *Introduction to machine learning (OIP)*. No. February, Cambridge: cambridge university press, 252 ed., 2010.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning*, vol. 4. New York: Springer, 2006.

[19] M. Welling, "Fisher Linear Discriminant Analysis," *Science*, vol. 1, no. 2, pp. 1–3, 2009.

[20] M. Cilimkovic, "Neural Networks and Back Propagation Algorithm," *Fett.Tu-Sofia.Bg*, 2010.

[21] N. Benvenuto and F. Piazza, "The backpropagation algorithm," *IEEE Transactions on Signal Processing*, vol. 40, no. 4, pp. 967–969, 1992.

[22] G. Panchal, A. Ganatra, Y. Kosta, and D. Panchal, "Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers," *International Journal of Computer Theory and Engineering*, vol. 3, no. 2, pp. 332–337, 2011.

[23] Unknown, "Neural Network Data Mining Explained," 2015.

[24] O. Sutton, "Introduction to k Nearest Neighbour Classification and Condensed Nearest Neighbour Data Reduction." 2012.

[25] K. Hechenbichler and K. Schliep, "Weighted k-Nearest-Neighbor Techniques and Ordinal Classification," *Molecular Ecology*, vol. 399, p. 17, 2004.

[26] T. Language and T. Computing, "MATLAB The Language of Technical Computing," *Components*, vol. 3, no. 7, p. 750, 2004.

[27] A. Harma, "Automatic identification of bird species based on sinusoidal modeling of syllables," *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, vol. 5, pp. 0–3, 2003.

[28] M. Lindemuth, "harmaSyllableSeg," 2014. `http://uk.mathworks.com/matlabcentral/fileexchange/29261-harma-syllable-segmentation`.

[29] D. P. W. Ellis, "PLP and RASTA (and MFCC, and inversion) in Matlab," 2005. `http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/`.

[30] C.-H. Lee, Y.-K. Lee, and R.-Z. Huang, "Automatic recognition of birdsongs using mel-frequency cepstral coefficients," *In this paper we propose a method to automatically identify birds from the sounds they generate. First, each syllable corresponding to a piece of vocalization is segmented. For each syllable, the averaged LPCCs (ALPCC) and averaged MFCCs (AMFCC) over all*, vol. 1, no. 1, pp. 17–23, 2006.

[31] C.-H. Lee, C.-C. Han, and C.-C. Chuang, "Automatic Classification of Bird Species From Their Sounds Using Two-Dimensional Cepstral Coefficients," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 8, pp. 1541–1550, 2008.

[32] W.-H. Tsai, Y.-Y. Xu, and W.-C. Lin, "Bird Species Identification Based on Timbre and Pitch Features of Their Vocalization," *Journal of Information Science and Engineering*, vol. 30, no. 6, pp. 1927–1944, 2014.

[33] L. Neal, F. Briggs, R. Raich, and X. Z. Fern, "Time-frequency segmentation of bird song in noisy acoustic environments," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 2012–2015, 2011.

Appendix A

System Evaluation Survey

The survey questions and responses are shown in Figures A.1 to A.4 below.

B-CHIRP Evaluation

*Required

How functional do you think the user interface is? *
(How functional in terms of what it sets out to do)

- ○ Excellent
- ○ Good
- ○ Fair
- ○ Poor
- ○ Terrible

How easily were you able to read the classification result? *

- ○ Very Easily
- ○ Easily
- ○ Normal
- ○ Hard
- ○ Very Hard

How did you feel about the delay in the classification result? *

- ○ Perfect time
- ○ Not too long
- ○ Okay
- ○ Long
- ○ Too long

Any further comments?

Submit

Figure A.1: This figure shows the survey questions.

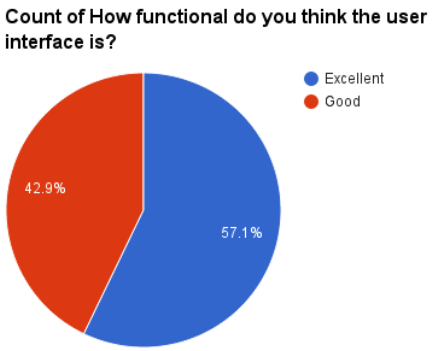


Figure A.2: This figure shows the answers to question 1.

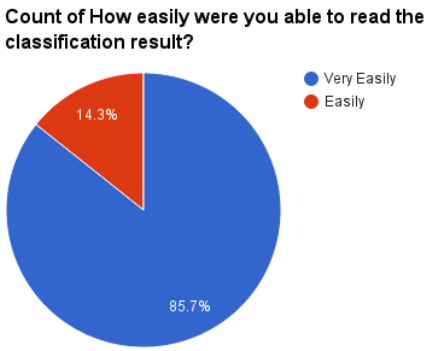


Figure A.3: This figure shows the answers to question 2.

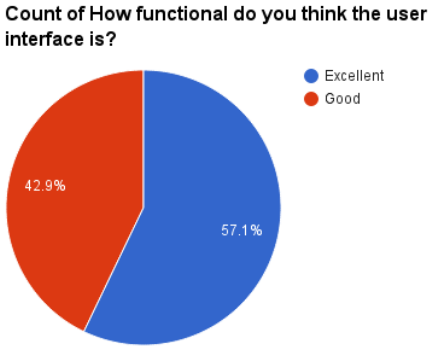


Figure A.4: This figure shows the answers to question 3.

Appendix B

Recordist Accreditation

Here follows the license used by the Xeno-Canto bird sound sharing website, and the names of recordists from whom the recordings were obtained.

Below is a summary of the Creative Commons License (the full license may be found here: http://creativecommons.org/licenses/by-sa/4.0/legalcode).

You are free to:

Share copy and redistribute the material in any medium or format **Adapt** remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms. Under the following terms:

Attribution You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

ShareAlike If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

No additional restrictions You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. Notices:

You do not have to comply with the license for elements of the material in the public

domain or where your use is permitted by an applicable exception or limitation. No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material.

Below is a list of the training recordings, and the recording artist:

```
BlackSmith_Lapwing_2.mp3 - Sander Bot
BlackSmith_Lapwing_4.mp3 - Sander Bot
BlackSmith_Lapwing_5.mp3 - Sander Bot
BlackSmith_Lapwing_6.mp3 - Niall Perrins
BlackSmith_Lapwing_7.mp3 - Daniel Danckwerts
Blue_Crane_1.mp3 - Charles Hesse
Blue_Crane_2.mp3 - Charles Hesse
Blue_Crane_3.mp3 - Sander Bot
Blue_Crane_4.mp3 - Sander Bot
Blue_Crane_5.mp3 - Sander Bot
Blue_Crane_7.mp3 - Niall Perrins
Blue_Crane_8.mp3 - Lynette Rudman
Egyptian_Goose_1.mp3 - Don Jones
Egyptian_Goose_2.mp3 - Sander Bot
Egyptian_Goose_3.mp3 - Sander Bot
Egyptian_Goose_5.mp3 - maudoc
Egyptian_Goose_6.mp3 - Lynette Rudman
Grey_Go-away-bird_1.mp3 - Bram Piot
Grey_Go-away-bird_2.mp3 - Fernand DEROUSSEN
Grey_Go-away-bird_3.mp3 - maudoc
Grey_Go-away-bird_4.mp3 - Jeremy Hegge
Grey_Go-away-bird_5.mp3 - Gabriel A. Jamie
Hadeda_Ibis_1.mp3 - (Unknown)
Hadeda_Ibis_2.mp3 - Sander Bot
Hadeda_Ibis_3.mp3 - Sander Bot
Hadeda_Ibis_4.mp3 - Sander Bot
Hadeda_Ibis_5.mp3 - Sander Bot
Hadeda_Ibis_6.mp3 - Sander Bot
Hadeda_Ibis_7.mp3 - Jacek Betleja
Red-chested_Cuckoo_1.mp3 - Eduardo Patrial
Red-chested_Cuckoo_2.mp3 - Fernand DEROUSSEN
Red-chested_Cuckoo_3.mp3 - George Wagner
Red-chested_Cuckoo_4.mp3 - Niall Perrins
Red-chested_Cuckoo_5.mp3 - Lynette Rudman
Red-chested_Cuckoo_6.mp3 - Jeremy Hegge
Ring-necked_Dove_1.mp3 - Charles Hesse
Ring-necked_Dove_2.mp3 - Don Jones
Ring-necked_Dove_3.mp3 - Bram Piot
Ring-necked_Dove_4.mp3 - Sander Bot
Ring-necked_Dove_5.mp3 - Sander Bot
Ring-necked_Dove_6.mp3 - Jeremy Hegge
South_African_Shelduck_1.mp3 - Derek Solomon
South_African_Shelduck_2.mp3 - Alan Collett
South_African_Shelduck_3.mp3 - Don Jones
South_African_Shelduck_4.mp3 - Sander Bot
South_African_Shelduck_5.mp3 - Lynette Rudman
South_African_Shelduck_6.mp3 - Lynette Rudman
```

Below is a list of the test recordings, and the recording artist:

```
Blacksmith Lapwing_Test_1.mp3 - Tom Wulf
Blacksmith Lapwing_Test_2.mp3 - Daniel Danckwerts
Blacksmith Lapwing_Test_3.mp3 - George Wagner
Blue Crane_Test_1.mp3 - (Unknown)
Blue Crane_Test_2.mp3 - Peter Boesman
```

```
Egyptian Goose_Test_1.mp3 - (Unknown)
Egyptian Goose_Test_2.mp3 - Rory Nefdt
Grey Go-away-bird_Test_1.mp3 - Rory Nefdt
Grey Go-away-bird_Test_2.mp3 - Charles Hesse
Grey Go-away-bird_Test_3.mp3 - Bram Piot
Hadeda Ibis_Test_1.mp3 - (Unknown)
Hadeda Ibis_Test_2.mp3 - (Unknown)
Hadeda Ibis_Test_3.mp3 - Peter Boesman
Red-chested Cuckoo_Test_1.mp3 - (Unknown)
Red-chested Cuckoo_Test_2.mp3 - (Unknown)
Ring-necked Dove_Test_1.mp3 - (Unknown)
Ring-necked Dove_Test_2.mp3 - (Unknown)
South African Shelduck_Test_1.mp3 - (Unknown)
South African Shelduck_Test_2.mp3 - (Unknown)
```

EBE Faculty: Assessment of Ethics in Research Projects

Any person planning to undertake research in the Faculty of Engineering and the Built Environment at the University of Cape Town is required to complete this form before collecting or analysing data. When completed it should be submitted to the supervisor (where applicable) and from there to the Head of Department. If any of the questions below have been answered YES, and the applicant is NOT a fourth year student, the Head should forward this form for approval by the Faculty EIR committee: submit to Ms Zulpha Geyer (**Zulpha.Geyer@uct.ac.za**; Chem Eng Building, Ph 021 650 4791). Students must include a copy of the completed form with the thesis when it is submitted for examination.

Name of Principal Researcher/Student: Alexander Cohen Department: Electrical Engineering

If a Student: Degree: Supervisor: Dr Simon Winberg

If a Research Contract indicate source of funding/sponsorship:

Research Project Title:

Overview of ethics issues in your research project:

Question 1: Is there a possibility that your research could cause harm to a third party (i.e. a person not involved in your project)?	YES	(NO)
Question 2: Is your research making use of human subjects as sources of data? If your answer is YES, please complete Addendum 2.	YES	(NO)
Question 3: Does your research involve the participation of or provision of services to communities? If your answer is YES, please complete Addendum 3.	YES	(NO)
Question 4: If your research is sponsored, is there any potential for conflicts of interest? If your answer is YES, please complete Addendum 4.	YES	(NO)

If you have answered YES to any of the above questions, please append a copy of your research proposal, as well as any interview schedules or questionnaires (Addendum 1) and please complete further addenda as appropriate.

I hereby undertake to carry out my research in such a way that

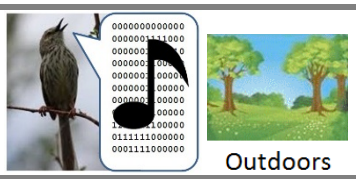
- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

Signed by:

	Full name and signature	Date
Principal Researcher/Student: Alexander Cohen	Alexander Cohen	12/10/15

This application is approved by:

Supervisor (if applicable):	DR SIMON WINBERG	12 OCT 2015
HOD (or delegated nominee): Final authority for all assessments with NO to all questions and for all undergraduate research.		
Chair : Faculty EIR Committee For applicants other than undergraduate students who have answered YES to any of the above questions.		

ID:	**SW-7**	
TITLE:	**Bird Call Heuristic-based Identification and Recognition Program (B-CHIRP)** *An outdoor theme project*	
DESCRIPTION:	This project is aimed to be an initial pathfinder to experiment with the prospect of developing a smartphone app that could either automatically identify a bird based on recording its song, or to assist a human in identifying the bird by providing cues and playing back recordings of bird sounds. This project will mainly focus on algorithm development and testing, collecting a variety of bird songs (of local birds) at different volumes and distortions, and then experimenting with filters and audio pattern matching methods, or possibly machine learning / neural networks to attempt to autonomously decide a type of bird based on the sounds it makes. At this stage, a sophisticated user interface and comprehensive software structure is not required, the main purpose is to experiment with algorithms and establish prototype code that can later be refined. Decent module interfaces and comments in the code are desired to make the effort usable in a future application.	
DELIVERABLES:	Application running on PC (or on mobile platform/smartphone)	
SKILLS/REQUIREMENTS:	Your choice of language; e.g. Matlab, Julia, Octave, PC-based Java program, or Java Mobile development using Android SDK.	
EXTRA INFORMATION:	Some existing applications that help birders identify birds: http://www.whatbird.com/ http://appcrawlr.com/android/merlin-bird-id-by-cornell-lab	
AREA:	Signal processing (optional: Embedded/Mobile app development)	