REAL TIME BIRD CALL CLASSIFIER

Kavilan D. Nair
1076342

School of Electrical & Information Engineering, University of the Witwatersrand, Private Bag 3, 2050, Johannesburg, South Africa

Abstract: This report documents the design, implementation and analysis of a real time bird call classifier. A convolutional neural network was developed using the Keras deep learning library and performed with an accuracy of 77% when tested on 10 species of South African birds. A mobile application that communicates with a server as an API endpoint was developed to achieve the functionality required by a real time bird call classifier. The application was developed for the iOS platform and can provide classification of recorded audio clips within 30 seconds and is considered real time. The server provided the ability to generate Mel-Frequency Cepstral Coefficients, classify this information and return the classification data to the device. In addition to this the ability to save the classification and location data to a database was also implemented.

Key words: Convolutional Neural Network, Mel-Frequency Cepstral Coefficients, Mobile Application

1. INTRODUCTION

The classification of birds is often used by ornithologists and conservationists to monitor bird populations and migratory paths. Birds can be classified by their visual appearance or by their unique calls associated with each species. Visual classification is difficult to perform as birds are often far away or obscured by trees. On the contrary, bird calls can travel far distances and direct line of sight of the bird is not required. Current bird audio detection and classification systems are not fully automated and require human labour to manually classify and label the data. Machine learning techniques can be applied to this problem to help automate the process. This document presents the design, implementation and critical analysis of a real time bird call classifier implemented on a mobile device. This was achieved through the use of Mel-Frequency Cepstral Coefficients (MFCCs) as visual representations of audio signals which are then classified by a Convolutional Neural Network (CNN).

2. BACKGROUND

2.1 *Requirements*

A machine learning model must be chosen and developed to provide the classification of bird audio recordings. The model must be implemented with a mobile application that allows for the user to record a bird call through the devices internal microphone and provide real time classification. An additional requirement was to integrate the mobile application with a database to store the classification and location information.

2.2 *Assumptions and Constraints*

It is assumed that all bird call data obtained for the purpose of training and testing the model has been correctly labelled. The mobile application will be developed for a single mobile phone operating system. The dataset will be limited to only bird recordings obtained within the country of South Africa.

2.3 *Success Criteria*

The project will be considered a success if all the requirements mentioned are met and the implemented system makes use of a machine learning model that performs with an accuracy of 80%. Additionally the system must return classification information within 30 seconds to be considered real time.

2.4 *Literature Review and Contextualisation*

Bird sounds can take the form of both vocal and non-vocal sounds, with the latter including sounds such as woodpecker's beak on wood. Additionally, vocal sounds can be subdivided into bird calls and bird songs, where bird songs often relate to mating songs that are different from the bird's typical call [1]. For the purpose of this report, only bird sounds classified as bird calls will be considered.

Previous work in the field of bird audio detection and classification has been carried out using machine learning techniques as a result of the Bird Audio Detection challenge run by the Queen Mary University of London and the LifeCLEF Bird Identification Task in 2016 [2, 3]. Many submissions to these challenges involved the use of CNNs. A CNN is a form of an Artificial Neural Network (ANN). The main difference is that CNNs are used for image and pattern detection primarily. CNNs are constructed of three different types of layers, convolutional layers, pooling layers and fully-connected layers [4].

Sprengel et al [5] developed a classifier using a CNN and it performed best in the birdCLEF challenge. A key takeaway from this approach was that data augmentation was applied to the input data to improve the models overall accuracy.

The Bird Audio Detection (BAD) challenge involved identifying the presence or absence of a bird call within a short audio recording [2]. The purpose of this challenge was to develop an automatic bird sound detection algorithm that performed with a high accuracy. This is different in the sense that the task was to classify the presence of bird audio as opposed to the classification of bird species. However, the approaches and techniques used can provide valuable insight in the context of bird audio classification.

Schluter et al [6] developed two CNNs for the submission of the BAD challenge. The proposed CNN architecture performed the best out of all competitor submissions and involved the use of Mel-scaled log-magnitude spectrograms as the data representation of the audio recordings. Another top performing network was presented by Pellegrini [7] which achieved 3rd place. This approach involved extracting 56 static F-BANK features and training a densely connected CNN.

Cakir et al [1] produced a classifier that won the judges award for the BAD challenge. Their approach involved converting the audio files into spectrograms to obtain spectro-temporal features contained in the bird call. The spectrograms were then used in a supervised machine learning approach that involved the use of a Convolutional Recurrent Neural Network (CRNN) which is comprised of a CNN followed by recurrent layers. The CRNN achieved an evaluation accuracy of 88.5%, which was marginally higher than a CNN by 3%. Adavanne et al [8] also had success implementing a CRNN. A CNN architecture was chosen to be implemented over the CRNN as it is less complex, easier to implement and has shown to be successful in image classification.

Cai et al [9] investigated the performance of bird species classification when using varied sets of features when applied to neural networks. The features that performed the best were Mel-Frequency Cepstral Coefficients (MFCCs) which are often used in the application of human speech recognition. The advantages of using MFCCs are that it is consistent with the human auditory perception and that MFCCs can be extracted from both aperiodic and periodic signals. Since bird audio sounds occupy a similar frequency range as the human ear, MFCCs were chosen to represent bird call audio.

Most classification models are developed using desktop grade hardware and programming languages such as Python, R and MATLAB. However, due to the usefulness and popularity of machine learning, many companies are making machine learning more accessible for mobile devices. Apple recently released Core ML, a framework which has support for pre-trained Keras and Caffe neural networks. Google's Android platform has also added support for TensorFlow trained models through the TensorFlow lite library.

An investigation to perform the signal processing and feature extraction using the device was undertaken. Options for both the Android and iOS platform were investigated and no libraries were found that provided the ability to generate the MFCC plots as required.

An attempt was made to use the Python programming language on a mobile device to allow for the use of pre-existing libraries to perform the signal processing. Kivy and the Pydroid 3 frameworks were both tested and could not make use of the Python packages and in the case of Kivy, permissions to access the mobile device's microphone could not be enabled. Both approaches were deemed infeasible. The Pyswift framework, is a Swift library that allows for the embedding of Python code on an iOS application. Python code was successfully ported across to the device but use of other Python libraries was not successful. Additionally it was deemed infeasible to implement a plotting library on either of these platforms to achieve the required MFCC output. Hence a decision was made to implement the MFCC generation on a server that the device interacts with over an internet connection.

3. SYSTEM OVERVIEW

The system components and flow of data is illustrated in Figure 1. The iOS application uses the devices built in microphone to record a bird call. The audio is uploaded to a cloud hosted storage bucket. Upon completion of the upload, a request containing the link to the recently uploaded file is sent to the server endpoint. The server then downloads the audio file and generates the MFCC plot. The MFCC plot, in image form is then classified by the trained CNN and the predicted bird and respective probability is returned to the mobile device. The mobile device presents this information to the user, who is prompted to save the information to a database if desired. The predicted bird, probability, link to the audio clip and the GPS coordinates are then sent to a database for permanent storage.

4. IMPLEMENTATION

All source code used in the implementation of this project can be found at the following Github organization: `https://github.com/BAD-Classifier`.

4.1 Data Acquisition

Supervised learning approaches require a vast amount of labelled data to perform successful training and validation of a model. All bird audio data was obtained from Xeno-canto, a website for sharing recordings of wild birds from across the world. It is a community driven initiative where anyone can access, upload and annotate bird sounds under the Creative Commons

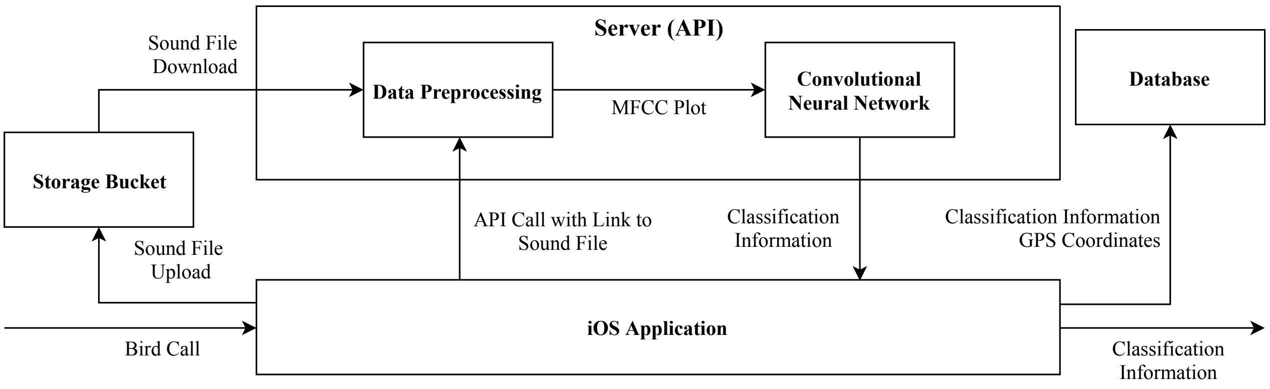
Server (API)
Sound File Download
Data Preprocessing
MFCC Plot
Convolutional Neural Network
Database
Storage Bucket
API Call with Link to Sound File
Classification Information
Classification Information GPS Coordinates
Sound File Upload
iOS Application
Bird Call
Classification Information

Figure 1 : System Overview

license.

The website provides an open Application Programming Interface (API) endpoint which allowed for the programmatic interface with the content hosted by Xeno-canto. By leveraging this API, South African bird recording data can be obtained. In order to automate the process of data acquisition, a Python script was written to download and appropriately label all available South African bird audio recordings. In total, 6123 recordings were obtained, which include 560 different species of birds. The available recordings on Xeno-canto were only downloaded if they were classified as bird calls as opposed to non-vocal bird sounds and bird songs.

4.2 Data Processing

4.2.1 Preprocessing The number of recordings per species of bird varied dramatically, with the most containing 97 recordings and the least containing 1. This uneven distribution in data would result in a biased classification model which favours the bird with more recordings. Furthermore, 6123 recordings is a large amount of data when considering hardware limitations. It was decided that only 10 species of birds be used as there was substantial data for each species whilst the dataset remained manageable for the available hardware. As a result, the dataset consisted of 661 audio recordings that varied in duration and contained different levels of noise. Each recording had either a sampling rate of 44.1 kHz or 48 kHz. Multiple forms of signal preprocessing were performed to transform the audio signals into a more robust and consistent dataset.

Firstly, each audio recording was trimmed into four second clips to obtain a more uniform dataset. One of the consequences of trimming the audio recordings was that some clips had no bird call present for the entirety of the four seconds. These silence filled clips were still labelled as bird calls and could hinder the achievable accuracy of the network. This problem was mitigated by identifying such recordings and removing them from the dataset. The mean values of the trimmed audio clips were compared to the mean value of the original clip and were removed from dataset if the mean was below 25% of the original clip.

Upon completion of the aforementioned data preprocessing, the number of recordings for each species was still dramatically skewed. A species contained 356 recordings, the lowest out of all species. This was taken as the baseline and only 356 recordings for each of the other species was taken to ensure a balanced dataset.

4.2.2 Feature Extraction The audio signals were converted into MFFC plots to represent the information contained within the audio signal. The following steps are required to generate MFCCs [10]:

1. Extract the frequency content by taking the Fourier Transform
2. Map the powers of the spectrum onto the mel scale shown in Equation 1
3. Obtain logarithm of the powers
4. Perform a Discrete Cosine Transform on the powers shown in Equation 2

$$M(f) = 1127 \cdot ln(1 + \frac{f}{700}) \tag{1}$$

$$MFCC = \sum_{n=0}^{N-1} M(f) \cdot cos\Big[\frac{\pi}{N}(n + \frac{1}{2})k\Big] \tag{2}$$

The MFCC generation was implemented using the Librosa library, a Python package for audio analysis. In addition, a software filter was applied to the remove any frequency components outside the frequency range of a typical bird call, 20 Hz to 12 kHz. Only the first 13 MFCCs were plotted as other coefficients did not show distinct differences between other species of birds. The additional more detailed MFCC plots would require a deeper network, more training time and resources to identify the features. An example of a generated MFCC plot is illustrated in Figure 2.

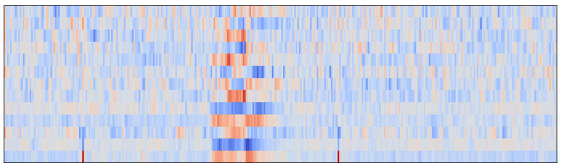

Figure 2 : Sombre Greenbul MFCC plot

4.3 *Machine Learning*

A CNN was implemented to provide image classification on the generated MFCC plots. The MFCC plots were formatted as PNG files with a resolution of 1400x400 pixels and served as the input dataset for the network. The Keras deep learning library was used to implement the CNN described in Table 1.

Table 1 : CNN Architecture

Layer	**Description**
Input	84x120x3 Colour image
Convolutional2D	32 Filters
Convolutional2D	32 Filters
Maxpool	28x40x32 Output size
Convolutional2D	64 Filters
Convolutional2D	64 Filters
Maxpool	14x20x64 Output size
Convolutional2D	128 Filters
Convolutional2D	128 Filters
Maxpool	7x10x128 Output size
Flatten	8960x1 Output size
Dense	1024x1 Output size
Dense	10x1 Output size
Activation	Softmax

The input to the CNN was an MFCC plot that was resized to an image with a resolution of 84x120 pixels and a depth of 3 to represent the RGB values. The image resolution was lowered to reduce both computation time and strain on available memory. Data augmentation was applied to increase the dataset size and provide shifted versions of the MFCCs. Data augmentation in the form of height shifting, width shifting and zoom were incorporated to improve the robustness and accuracy of the model.

The dataset used to train the model consisted of 10 species of birds with 356 MFCC plots each. A training/testing split of 80/20 was applied to train the network. Deep learning is a computationally heavy task and can take many hours to train. The training process can be sped up by utilizing the parallel processing of a GPU. A desktop computer with a Nvidia GTX970 (4GB VRAM) was used to train the model.

The learning rate, batch size and number of epochs comprise some of the networks hyperparameters. A learning rate of 1e-5 was selected as it resulted in the network's accuracy improving at a relatively quick rate and did not result in a divergent loss function. A batch size of 5 was used as this is the maximum that the GPU memory could handle. The network was set to train over 3000 epochs as this gave it enough iterations to achieve a steady validation accuracy. It did not however converge to a single value and could improve if run for longer.

The CNN was trained with the Adam optimizer due to its efficiency. The chosen activation function for the hidden layers of the network was the Rectified Linear Unit (ReLU) which has proven successful in existing CNN models. Batch normalization was included to reduce the training time necessary to achieve a high accuracy and prevent computations from becoming too large. Furthermore, dropout, with a rate of 50% was applied to prevent the neural network from overfitting to the training data. Categorical cross entropy was employed as the loss function. The final layer of the network implemented the softmax activation function which results in probabilities being assigned to each of the classes. These probabilities sum to one across all classes.

4.4 *Mobile Application*

As previously mentioned, the signal processing was chosen not to be implemented on the device. Instead a server was developed to communicate with the mobile application.

4.4.1 iOS Application The Xcode IDE and Swift programming language were used to develop the application for the iOS platform. The application provides the user with the ability to record sounds using the devices built-in microphone and obtains the current location data by utilizing its GPS module. Access to the microphone and location services require the user to provide the necessary permissions.

Google's mobile platform, Firebase was used as a cloud service to handle user authentication, file storage and database needs. User authentication through username and password was implemented so that an individual could view previously recorded sounds and classification information. The file storage bucket was used to store the raw audio files which could be listened to at a later stage.

A database was implemented to permanently store the classification and location data for the user. This was achieved by using the Firebase real time database, an unstructured NOSQL database. The URL to the the stored audio recording, classification information, probability of classification, latitude, longitude and user identification string are stored in each entry to the database. This enabled the ability for a user to view and listen to previously recorded and classified recordings.

4.4.2 Server Both signal processing and feature extraction were implemented on a server. This allows for less of the computation to happen on the mobile device and perform the classification faster.

The server logic was implemented using the Python programming language to plot the MFCCs and provide classification using the trained CNN. In order to reduce the data usage when communicating with the server, it was decided that the classification of the image should take place server side, as a result the image would not need to be sent back to the device.

The server was implemented as a RESTful API using the Flask framework. The audio recording is first uploaded to a storage bucket which returns a unique URL to the file. A POST request with the URL to the file can be made to the endpoint. Upon receipt of a request, the server downloads the audio file and generates the MFCC which is then classified by the CNN and the classification information is then returned to the mobile device.

To achieve a system that works on any internet connection. The Flask server was containerized using Docker and deployed to a cloud computing instance, known as a Droplet hosted by DigitalOcean. The Droplet ran the Ubuntu 16.04 operating system with 4GB RAM and 25GB of storage.

5. RESULTS

The accuracy of the CNN was obtained during the training phase, where the model's training and validation accuracy were recorded after every epoch and is shown in Figure 3. Training accuracy represents how accurate the model is when the training data is classified and the validation accuracy is how accurate the model is when classifying the unseen testing data. The model achieved a maximum training accuracy of 83% and a maximum validation accuracy of 77%.

The loss of the network throughout the training phase was calculated using categorical crossentropy loss function. The loss values for both the training validation are illustrated in Figure 4. The model achieved a minimum loss value of 0.7 and 0.9 for training and validation loss respectively.

The mobile application was implemented and was able to classify an audio clip in under 30 seconds. The time taken to classify a recording depends on two things, the length of the audio recording and the available internet connection speed and latency.

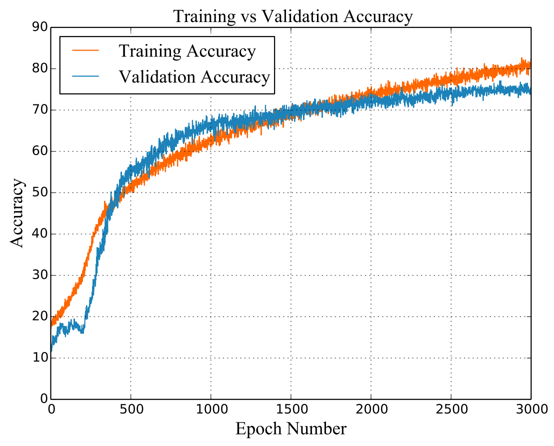

Figure 3 : Training and validation accuracy over 3000 epochs

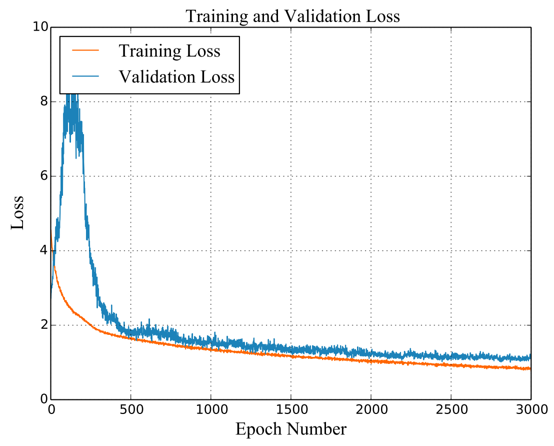

Figure 4 : Training and validation loss over 3000 epochs

6. CRITICAL ANALYSIS AND RECCOMENDATIONS

The model achieved a validation accuracy of 77% which was slightly under the 80% target. The trend of the validation accuracy shown in Figure 3 indicates that if left to train for longer, a higher validation accuracy could have been achieved. This was not done due to the hardware and time constraints. The training and validation values were closely correlated up until the 2000th epoch where the two lines diverged slightly. This is an indication of overfitting the model to the training data.

The utilization of a more powerful GPU with larger memory could allow for a deeper network architecture, larger batch size and quicker run times. These factors could allow for a more accurate model to be achieved. Alternatively a different network architecture such as a CRNN could be implemented to achieve a slightly higher accuracy.

Figure 4 shows a logarithmic decay in the value for loss which is expected when using a cross entropy loss function. An ideal model would have a loss of 0. According to the trend of the graph, a lower loss seems possible if run longer. However, a loss value of 0 does not seem achievable as the line is approaching an asymptote.

The mobile application was successful in classifying bird audio recordings, but is dependent on an internet connection to perform the signal processing on a server. This is not desirable as an internet connection may not be available in the area where the bird audio classification needs to take place.

The current server that runs the API endpoint cannot provide simultaneous classification using the CNN. This could be a result of the memory limitations of the available resources to the application. A queuing system that prevents simultaneous classifications could be implemented to alleviate this issue.

The current implementation of the system cannot classify multiple bird calls in a single audio clip. The identification and separation of multiple bird calls could be added to improve the functionality of application. Furthermore, the application expects a bird call to be present in any given recording which could lead to false classifications if no bird call is present in an audio clip. The addition of another system to detect whether or not a bird call is present in a recording could be implemented to detect the lack of a bird call.

Since the application stores both the audio files and classification information using online storage and a database, there is potential to use these recordings as training data to improve the accuracy of the model.

All the software tools to used to create the machine learning model, iOS application and server logic were free and open source. Hence the cost of development of the system is negligible provided the user has a iPhone and a computer. Firebase services are provided for free until the application starts to use significant resources. The hosting of the server on the DigitalOcean droplet cost US$20 per month.

7. CONCLUSION

The implementation of a real time bird call classifier that is capable of classifying 10 South African bird species on a mobile application was realized. The classification model achieved an accuracy of 77%, three percent less than the desired level of accuracy. This accuracy level could be increased if the model was left to train for a longer duration on hardware with more memory. The application was capable of classifying bird calls within 30 seconds and can be considered to perform in real time. Additionally, the ability to add both classification and location data to a database from the mobile application was implemented.

REFERENCES

[1] E. Cakir, S. Adavanne, G. Parascandolo, K. Drossos, and T. Virtanen. "Convolutional recurrent neural networks for bird audio detection." In *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 1744–1748. Aug 2017.

[2] D. Stowell, M. Wood, Y. Stylianou, and H. Glotin. "Bird detection in audio: A survey and a challenge." In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6. Sept 2016.

[3] H. Goëau, H. Glotin, W.-P. Vellinga, R. Planqué, and A. Joly. "LifeCLEF Bird Identification Task 2016: The arrival of Deep learning." In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum*, pp. 440–449. Evora, Portugal, Sep. 2016. URL `https://hal.archives-ouvertes.fr/hal-01373779`.

[4] K. O'Shea and R. Nash. "An Introduction to Convolutional Neural Networks." 11 2015.

[5] E. Sprengel, M. Jaggi, Y. Kilcher, and T. Hofmann. "Audio Based Bird Species Identification using Deep Learning Techniques." *LifeCLEF 2016*, pp. 547–559, 2016.

[6] T. Grill and J. Schlter. "Two convolutional neural networks for bird detection in audio signals." In *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 1764–1768. Aug 2017.

[7] T. Pellegrini. "Densely connected CNNs for bird audio detection." In *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 1734–1738. Aug 2017.

[8] S. Adavanne, K. Drossos, E. akir, and T. Virtanen. "Stacked convolutional and recurrent neural networks for bird audio detection." In *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 1729–1733. Aug 2017.

[9] J. Cai, D. Ee, B. Pham, P. Roe, and J. Zhang. "Sensor Network for the Monitoring of Ecosystem: Bird Species Recognition." In *2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*, pp. 293–298. Dec 2007.

[10] P. Cryptography. "Mel Frequency Cepstral Coefficient (MFCC) tutorial." URL `http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/`.

Appendix

A Group Reflection

Table 2 : Division of components between partners

Component	Sub Component	K. D. Nair (Author)	I. M. Tchaparov (Partner)
Data Acquisition	Xeno-Canto Download script	✓	
	Cut data into 4 second clips		✓
	Mean analysis		✓
Data Processing	Research	✓	✓
	Filtering		✓
	Mel-Frequency Cepstral Coeffiient Generation		✓
	Image generation of MFCC		✓
Machine Learning	Research	✓	✓
	CNN Implemetation	✓	✓
	CNN Training	✓	✓
	CNN hyperparameter adjustment & retraining	✓	✓
Mobile Application	Research	✓	✓
	Implement UI	✓	
	Integrate with Microphone and GPS	✓	
	Develop Flask API	✓	✓
	Develop Classifier Script	✓	✓
	Dockerize and Deploy to cloud service	✓	
	Integrate App with server	✓	
	Integrate app with Firebase services	✓	

Table 2 indicates the division of work carried out by each of the group members. The work was split where possible but the core aspects of the project were executed by both members.

The work was carried out at the University of the Witwatersrand in the D lab. Both members met up in the morning at Wits in the lab where an informal meeting, discussing current progress and the plan for the day was discussed. Even though the members were working on different components of the project at times, both members were willing to provide assistance and feedback on all aspects of the project. If a major issue was encountered, both students would change their attention to the issue until it was resolved.

Both of the group member's strengths were considered when deciding on the division of work. Iordan has had previous experience in software based signal processing whilst Kavilan has had prior exposure to mobile app development.

Both students put in a great amount of effort to ensure the project was a success. The machine learning component of the project required a lot of time and attention. This was both students' first encounter with the topic. Many additional hours over weekends were also contributed by both members. The work load was evenly distributed between students and the quality of work produced was of a high standard.

Over the duration of the project, the two members disagreed on several things related to the project. This conflict was handled professionally and in an orderly manner. A third opinion from the project supervisor was often asked for to help settle some disagreements. The conflicts were few and far between and did not halt or hinder the progress of the project. Often the disagreement led to more research around the topic resulting in a more insightful decision being made.

The process of engaging and solving a difficult engineering problem in a group presented many challenges along the way. However, the process has proven to be a rewarding one in which the students improved not only their technical skills but their communication and planning skills.

School of Electrical and Information Engineering
University of the Witwatersrand, Johannesburg
ELEN4002/4012: Project Specification Outline

To be completed by supervisor
Assessment:
☐ Unacceptable ☐ Poor
☐ Acceptable ☐ Good ☐ Excellent

Project Title: Real time bird call classifier

Group Number:	18G05	Supervisor Name:	Ellen De Mello Koch
Member 1 Name:	Iordan Tchaparov	Student number 1:	1068874
Member 2 Name:	Kavilan Nair	Student number 2:	1076342

Project Specification:

Many scientist and conservationists track birds through remote monitoring of bird sounds. They use this data to monitor migration patterns and population density of bird species and for other research purposes. Currently Bird Audio Detection (BAD) systems are used to identify bird species, however a majority of the current systems require human intervention to label the data. Machine learning techniques can eliminate the need for humans to actively classify the data as well as improve the accuracy of data classification.

There have been cases where machine learning has been used to identify birds based on audio signals however no real time applications exists. In order to achieve a BAD system that ornithologists can use in the field, a mobile application using machine learning that can record and identify bird sounds is required. A machine learning model that is trained with South African bird sounds will be implemented and transferred to a mobile app to allow for real time classification. A feature to upload the recorded audio signal and its location to an online database would also be desirable.

Audio signals are often converted into spectrograms to identify features contained in the sound. A spectrogram is a visual representation of an audio signal that plots frequency against time and illustrates the amplitude through a heat map (whereby dark blue is the smallest amplitude and bright yellow/white is the largest amplitude). Furthermore, Convolution Neural Networks (CNN) have proven highly accurate in identifying images and the same can be applied for spectrograms. This will be the starting point of the project but the model will be improved and other techniques incorporated to achieve the highest accuracy possible.

The South African bird call data will be obtained from https://www.xeno-canto.org/. The data will have to split up into a training set and testing set to train the machine learning algorithm and to verify the accuracy of the model respectively.

A smartphone application that will run the transferred machine learning identification model is required to be implemented. The Android platform has been chosen as the development environment for this application as it is open-source and there is a plethora of devices with different hardware specifications that will be able to run the application. The recording of the bird sounds will be done through the microphone of the mobile device the application is run on. This means that the quality of the recording is dependent on phone hardware. In addition, many other factors such as noise may affect the audio signal resulting in a lower accuracy of the model. Ambient noise will be filtered out to reduce its effect on the quality and to improve integrity of the bird call audio signal.

Milestones:

1) Obtain data from https://www.xeno-canto.org/ by using a python script as it doesn't allow for batch downloads.

2) Split it into training data and testing data.

3) Research and implement a digital filter to reduce ambient noise to improve audio quality.

4) Create spectrograms from the audio recordings.

5) Implement a machine learning model, most probably a Convolutional Neural Network.

6) Training the model and improving it iteratively, identifying key features. The goal is to achieve an accuracy of 95% for the model.

7) Transfer the optimized trained machine learning algorithm onto a mobile device by developing an Android mobile application.

8) Implement additional features such as the uploading of the audio recording and its location information into an online database.

Preliminary Budget & Resources:

The South African bird call data will be obtained from https://www.xeno-canto.org/. The data is freely available and requires no purchase.

The Python programming language will be used to implement the machine learning model. A signal processing library will be utilized for the noise reduction. Python and all of its libraries are free.

The open-source software library, TensorFlow for machine learning, will potentially be used to implement the machine learning model and to facilitate the transfer of the model to the mobile application. TensorFlow is licensed under the Apache license 2.0, meaning that the students can modify, distribute, patent and commercialise with royalties required to be paid.

The Android application will be implemented using Android Studio, a free integrated development environment, and the Java programming language.

Potentially the additional feature of uploading the audio sound and its location to an online database will be hosted on FireBase with a free account plan.

Both students have access to laptops and only free software will be used, resulting in no budget being required for this project.

Risks / Mitigation:

The website https://www.xeno-canto.org/ does not allow for multiple downloads natively. This can be done manually, but that is not sensible as there are thousands of recordings. A script will need to be written to automate the downloading of data to reduce the time in obtaining the data.

There might not be enough training data available to achieve an accuracy of 95%. It is assumed that the data obtained from https://www.xeno-canto.org/ website has been correctly labelled.

The currently selected machine learning algorithm, a Convolutional Neural Network may not be the optimum choice and may have to be changed or adapted to improve accuracy. A Recurrent Neural Network could be implemented as an alternative.

TensorFlow may not be the correct framework to use for this project. Complications may arise, which will require that it be substituted for a different open-source machine learning library/framework such as scikit-learn.

Transferring the machine learning algorithm onto a smartphone may result in long processing times due to computational requirements. A trade-off between accuracy of the model and computational power will have to be made to ensure that it can run in real time.

Training the machine learning model requires time and can be a bottleneck to the project. Strict adherence to a time schedule will be required to complete this project and therefore any deviations due extra time for the model training will need to be considered from the beginning.

Real Time Bird Call Classifier Project Plan

ELEN4012 - Lab Project
Kavilan Nair (1076342) & Iordan Tchaparov (1068874)
Supervisor: Ellen De Mello Koch

School of Electrical & Information Engineering, University of the Witwatersrand, Private Bag 3, 2050, Johannesburg, South Africa

Abstract: This document details the implementation plan, testing methodology and management of the Real Time Bird Call Classifier Project. The project has a duration of 7 weeks after which it must be submitted along with all relevant documentation. The project is comprised of a data component, machine learning component and mobile application component. The data component involves obtaining and processing the bird sound recording data in terms of filtering and feature extraction. The machine learning component involves implementing a CNN and CRNN and selecting the better performing neural network for the classifier. The mobile application component involves porting this machine learning model to a mobile device to enable real time bird call classification. Python, TensorFlow, scikit-learn, DynamoDB and S3 Storage technologies will be used to realize the project. A successfully implemented model should have a classification accuracy of at least 80%.

1. INTRODUCTION

Audio classification and identification is important in the study and monitoring of bird wildlife [1]. The audio from birds comes in vocal and non-vocal forms, with the former being subdivided into bird calls and bird songs [2]. Just as human speech, bird calls and songs have patterns and features that can be identified and extracted to allow classification of bird wildlife. These processes are usually manual and require human intervention [1]. Through the incorporation of machine learning techniques and methods, this classification process can be automated to a certain degree of accuracy. This document details the implementation of a system that will be able to classify bird calls and songs through the use of machine learning. Details of the testing methodologies to be used for evaluation of the project are given. The management of this project is also included by giving time estimates for all project tasks in the form of a Work Breakdown Structure (WBS).

2. PROJECT OVERVIEW

The project is first contextualized in terms of its requirements, success criteria, assumptions and constraints. This allows for a summarized overview for the main tasks that must be accomplished. In this document bird sounds will encompass both bird songs and bird calls.

2.1 Requirements

Machine learning techniques must be utilized to develop a model to classify bird sounds. This model must be ported to an application that can be run on a mobile device. The mobile application should allow a user to classify an audio sample, recorded by the user through their device in real-time. Additionally, while not a requirement, it is desirable that any recordings that a user takes using the mobile application will be stored in an online database along with the location data of where the recording was taken.

2.2 Success Criteria

The project will be deemed successful if it can meet all the requirements listed and successfully identify and classify input bird sounds with a minimum accuracy of 80%.

2.3 Assumptions and Constraints

It is assumed that all the data that will be obtained is accurately labelled and that there is enough data to train the machine learning model. There exists a constraint in terms of a budget and therefore the purchase of high end equipment for both sound recording and compute power is infeasible. It is assumed that the mobile application will work with the same accuracy regardless of the operating system the mobile device is using (Android or iOS). Due to time constraints, the mobile application may be only developed for one of these operating systems.

2.4 Project Components

From the requirements, success criteria, assumptions and constraints, the project can be broken down into three main components: data, machine learning and the mobile application. Figure 1 shows the project process diagram.

2.4.1 Data - This component of the project involves both data acquisition and data processing. Data acquisition refers to obtaining samples of bird sounds. As these will be used to train the machine learning model for classification, numerous recordings are required. South African bird sounds will be the main focus of this project. The recordings of the bird sounds are subject to distortion and noise from both the external environment and the quality of the recording device. For better accuracy, it is desirable for the recordings to have as little noise as possible. A method to remove or lower noise and its impact on the recording, would be to listen to the each of the recordings and manually remove noisy sections. This is a tedious and impractical process as a large number of recordings are required. The recordings can be filtered digitally with a bandpass filter with a bandwidth of 20 Hz - 12 000 Hz as most South African bird sounds are within this range [3]. Once they have been filtered, features will be extracted from the recordings and will be used as input to the model. This will require the data to be partitioned into two sets, a training set and a testing set.

2.4.2 Machine Learning Model - This component involves utilizing machine learning techniques which can include, but not limited to, an Artificial Neural Network (ANN) for classification purposes. The training data will

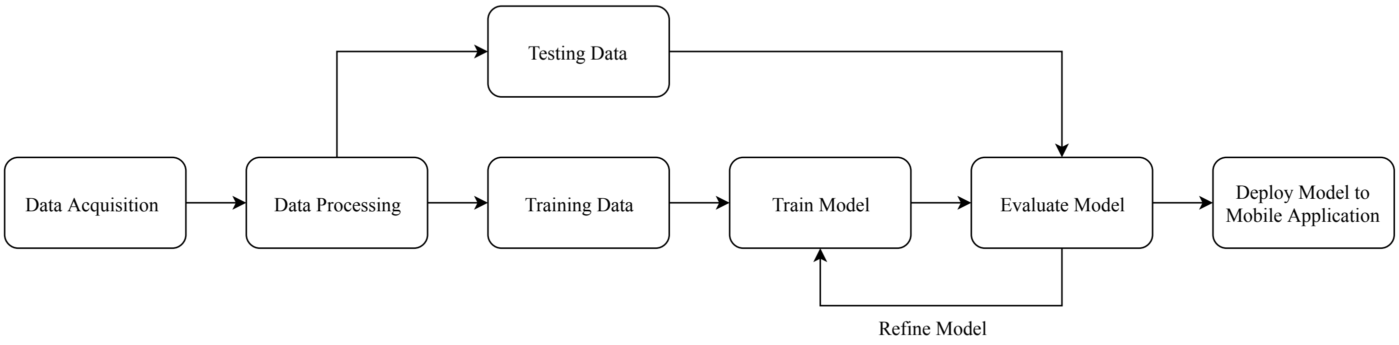

Figure 1: Project process diagram

be used to develop the ANN model and subsequently the testing data will be used to refine and optimise the model. This is an iterative process, hence the model will need to be refined multiple times, making it the most time consuming task in the project. Additionally, careful consideration will need to be taken to not over-fit the model to the training data as this will lead to a lower classification accuracy. An optimum model needs to be found whereby it can accurately classify based on the training data but remains generalized enough so that newly applied testing data can still be classified.

2.4.3 Mobile Application - This component involves the integration of the ANN model with an application that can run on a mobile device. This is desirable as it can be used in the field by ornithologists and birdwatchers to identify birds where there is often lack of an internet connection. The application will allow the user to record a bird sound with the built-in microphone on their device. This sound will then be classified by the model and the associated bird will be identified to the user. The application then can be used to upload the recording, its classification and location to an online database whereby this information could be used for future wild-life classification applications.

3. PROJECT IMPLEMENTATION

This section details the different components of the project with more depth and methods of implementation. For any scripting in the project, the Python language will be used.

3.1 Data Acquisition

A crucial step in project, due to the supervised machine learning process, is to collect useful data that can be used to train the classification model. This requires a large number of species of birds which reside in South Africa to be obtained. There exists a website, `https://www.xeno-canto.org/` which is dedicated to sharing bird sounds from all over the world. It is an open platform which is typically used by scientists, birders and researchers to access and contribute to the growing database of bird sounds. The website allows a user to access bird sounds according to country, allowing filtration of non-South African Birds. Xeno-Canto currently has 6123 recordings from 560 different species from within South Africa. Xeno-Canto is a a community initiative and by nature, different qualities of recordings are uploaded to the site resulting in the need to filter recordings to remove potential noise and unwanted signals. While there would be a benefit in using state of the art equipment to record bird sounds to obtain recordings of better quality than those on the website, this would make the data acquisition drastically longer. This is impractical for the project due to the time span in which it must be completed and this is reason that the recordings of Xeno-Canto will be used.

Xeno-Canto does not offer a bulk download service for all these recordings and it is impractical to manually download over 6000 recordings. Xeno-Canto does however offer an open RESTful API to the website and a script can be written to automate the data collection process by utilising the API. The API documentation can be found at `https://www.xeno-canto.org/article/153` and while limited, it is good enough to achieve this goal. The API is able to respond with country specific bird data depending on the API endpoint. The following endpoint responds with all the bird data collected in South Africa with a GET command: `http://www.xeno-canto.org/api/2/recordings?query=cnt:south_africa`

The JSON response includes the URL of the file to download, the genus, species and name which is all the necessary information to download the file and name it appropriately. A portion of the response to the GET request is shown in Appendix A. All files on Xeno-Canto are in MP3 format. A rough estimate for the total size of this data is about 3.5 GB given the current state of the Xeno-Canto library.

It is vital to understand the data that has been collected. A script that iterates through all the sound files will generate insights into how many species of birds the total dataset contains, number of recordings per species and the length of each of the recordings. These insights are useful for splitting the data into training and testing data in a more meaningful manner.

3.2 Data Processing

Sound recordings have a number of patterns and features that can be extracted including volume, energy and bandwidth. This requires signal manipulation techniques, namely conversion to and from the frequency domain to extract such features. Graphing the frequency domain against the time domain results in a spectrogram, with the colour scheme of the plot indicating the magnitude of power of the signal. The Mel Frequency Cepstrum (MFC) is a method of extracting and representing features in a signal and will be utilized [4]. The MFC has found success in being used for speech recognition as the Mel scale is linear at low ranges and logarithmic at higher ranges [4]. This corresponds to the way in which the human ear works and allows certain ranges of hearing for humans, with this range encompassing the vocalization of birds. Mel Frequency Cepstral Coefficients (MFCC) are components of the MFC and have proven to be a very robust and effective way to segment audio recordings to extract features [5]. The method in which to obtain MFCCs is as follows:

- Split the recording into multiple groups of frames through the use of a window. These groupings should not be too large, with sizes of 20 ms to 50 ms frames.
- Take the Fourier transform of each obtained windowed part of the recording.
- Transform the powers of the spectrum into the Mel scale using triangular overlapping windows.
- Take the logarithm of the powers at each of the Mel frequencies.
- Apply the Discrete Cosine Transform on the Mel log powers. The result of this is the MFCCs.

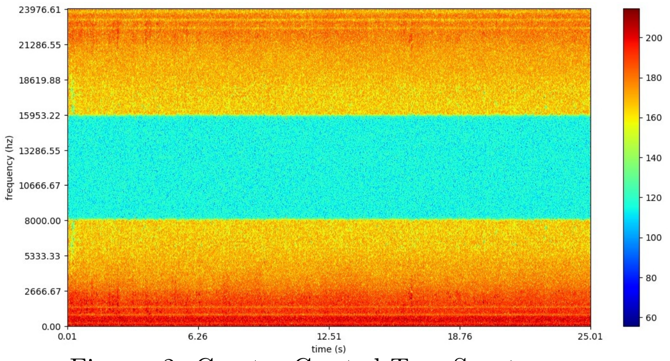

Figure 2: Greater Crested Tern Spectrogram

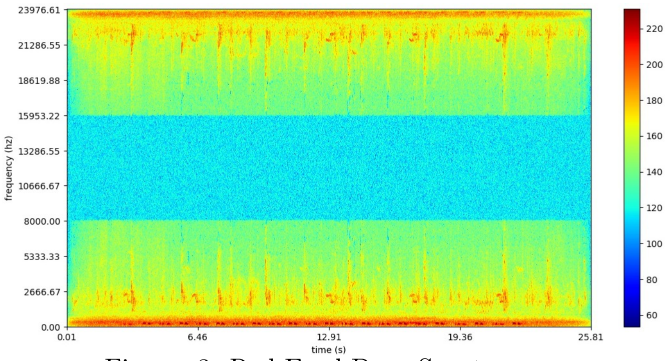

Figure 3: Red Eyed Dove Spectrogram

The spectrograms for two South African birds, the Greater Crested Tern and the Red Eyed Dove, were generated and are present in Figure 2 and Figure 3 respectively. Neither sound recording was filtered. Applying filtering techniques (namely bandpass filters) can result in even clearer recordings and isolate the desired frequency range. Obtaining the MFCCs of the recordings will allow better features to be extracted, resulting in clearer and distinct spectrograms. These will be used to represent the input data to the machine learning model.

3.3 *Splitting of Data*

Since the data available from Xeno-Canto is identified and labelled, a supervised machine learning model is suitable to train an accurate classifier. The existing bird sound data should be split into a training dataset and testing dataset. The training data will be used to develop and teach the model and the testing data will be used to evaluate the performance of the model.

The training and test data can be split manually depending on the number of species and recordings associated with it. However, due to the amount of recordings, this proves unreasonable. The simplest way of splitting the data is to write a script to randomly select recording samples and classify them as either training or testing data. A typical split between the two data sets usually ranges between 60% - 80% for the training data and 20% - 40% for the testing data [6]. This method for splitting the data might not be desirable as it may lead to a skewed model depending on the distribution of the data. If the training data is heavily skewed to certain species of birds, the model will tend to reflect this skew and is more likely to predict a sound as that species. In order to mitigate this, the data should be split as evenly as possible with respect to the number of recordings per species.

3.4 *Machine Learning*

The bird sound recordings will have features extracted and compiled into MFCC spectrograms, visual representations, that allow pattern identification and classification. By inputting a large number of these visual representations, the machine learning model will be able to find patterns between sets of different spectrograms and use this to classify them. This type of problem falls under image classification, where the spectrogram image will be the input to the model and will be classified from a list of known birds. Convolutional Neural Networks (CNN) have shown to be a novel approach to accurately classify images and thus will be implemented [7]. CNNs usually work by segmenting an image into small subsections, processing them, learning something about that particular pattern and then moving onto another subsection [7].

CNNs are made up of multiple hidden layers in addition to the input and output layers [8]. The hidden layers can include convolutional, pooling, and fully connected layers. Layers are comprised of numerous neurons - nodes which are interconnected with one another. Each of these nodes can either have a weight or activation function attached to it, which processes its input into a different form to be passed onto the next layer. The number of neurons that each of these layers contain can be adjusted in an attempt to improve the classification accuracy of the model. As the output from one node is passed to another, the model begins to learn what certain bird sounds look like based on their labels. The more input data of a certain type, the better the probability of the model to classify that data accurately when it is input to the model.

Recurrent Neural Networks (RNNs) are machine learning models that use their output from one epoch as the input to the next epoch in addition to their training data. This is particularly useful for speech recognition as words depend on the words said prior [9]. As the bird sounds are a form of vocalization, it would be worthwhile to investigate these ANNs as they may allow the classification of patterns in a better way. However, it is decided that a visual representation of the recordings is also a very good manner of representing patterns and so a hybrid ANN will also be implemented, a Convolutional Recurrent Neural Network (CRNN) [2]. A CRNN makes use of features from both CNN and RNN in an attempt to provide better accuracy [2].

The implementation of the machine learning model and training process will not be done from first principles. The Python programming language will be used to implement the models as it has extensive supporting libraries for data science applications. The TensorFlow library and the scikit-learn library are suitable tools for the machine learning process. Additionally Jupyter Notebooks, which uses Python, will be used to help with the development of the code as it allows isolation of sections of a codebase, making it easier to adjust hyperparameters and other features without affecting the entire codebase.

3.5 Evaluate model

Once the neural networks have been implemented and trained, they will be given the testing data to classify. There will be an associated accuracy based on the results of this testing data classification, which in turn will be used to determine whether a model is working correctly or not. Manual verification will also be done alongside the model evaluation to determine if the results of the model are what they are should be. If the accuracy of a model is low, it will have some of its hyperparameters adjusted and trained again in an attempt to develop a better model. This is an iterative process and it is expected that the first few iterations will result in relatively low accuracy of correct classifications. Each iteration will better the models until they converge to a static accuracy amount. It is important to make sure that the number of iterations are balanced so that the models are not over-fit to the data otherwise new input data will result in low accuracies. The models have to be general enough to classify the testing data with good accuracy but to also be able to classify new input testing data that was not in the original training-testing set. Care must be taken to never train on the testing set. This would be incorrect as the model is then classifying data that it has been trained on prior, resulting in a high accuracy as the model was exposed to the data before. Once this evaluation process is done, the more accurate machine learning model between the CNN and the CRNN will be selected to be used in the next stage of the project, the mobile application component.

3.6 Mobile Application

A mobile application that allows a user to record a bird sound in the real world and be classified is a major requirement of this project. This means that the application has to make use of the mobile device's built-in microphone. An Android application will be built using Android Studio and the Java programming language. There will be more of a focus on the Android operating system over iOS as it is a more accessible platform to develop for and there is more support for it. If time permits, an iOS application may also be developed. The following components need to be completed to successfully implement the real time bird call classifier as a mobile application.

3.6.1 User Interface Design The User Interface (UI), is required to be designed to have an intuitive and attractive user interface. Certain native UI components are available on the Android Mobile platform and should be taken into consideration when designing the interface. UI mockups are a useful tool to quickly prototype the look and feel of the application before committing to implementing it in code.

3.6.2 TensorFlow Lite Porting The trained TensorFlow bird call classifier model cannot be used directly in the mobile application. Only TensorFlow Lite models are currently compatible with the Android mobile application platform. The TensorFlow Lite converter can be used to convert the model so that it can be ported to an Android application [10]. TensorFlow Lite makes use of the Android Neural Networks API to optimize the performance of the classifier. The conversion process is outlined in Figure 4.

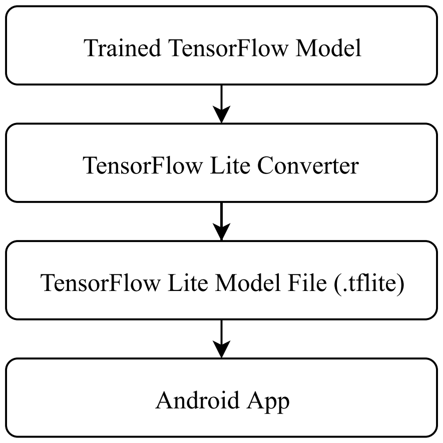
Trained TensorFlow Model
TensorFlow Lite Converter
TensorFlow Lite Model File (.tflite)
Android App

Figure 4: TensorFlow Lite conversion process

3.6.3 Code implementation The mobile application needs to make use of the built-in microphone present on all modern Android mobile devices. The application should record the bird sound for a user defined duration and save the sound clip to the local storage of the device.

It will often be the case that there is ambient and other unwanted noise when a user is recording a bird sound for identification. In order to increase the accuracy of the classifier in non-ideal conditions typically found when recording wildlife in the field, a digital signal processing and filtering stage will need to be implemented. The sound clip will need to be trimmed and filtered, which can be achieved by making use of the built in functionality contained in the Superpowered audio processing library [11].

Once the sound is filtered and trimmed, it should then be processed by the identifier model to classify the bird. The classifier should then display the genus, species and name of the identified bird. In addition to this information, the Google Maps API should be leveraged to also use the geo-location information of where the recording was taken and save this location data to the local storage of the device [12].

In order for the community to benefit from all the data that could be potentially collected whilst using this application. The locally stored identified bird calls alongside the relevant geo-location data should be uploaded to an online database when the phone has an internet connection. This will allow the information to be accessible.

A database that stores the the identified bird calls and associated information is required to be designed so that it can integrate seamlessly with the mobile application and support multiple users. The database should be hosted online using a cloud computing service company such as Amazon Web Services. A combination of DynamoDB and Amazon S3 is suitable for the storage of the actual bird sound recording and the classifying information associated with the recording [13].

4. TESTING METHODOLOGY

By its nature, machine learning includes a testing phase when training and developing the model. The model's classification accuracy can be determined by calculating the number of correctly classified bird sounds compared to the total number of sounds in the testing data set, expressed as a percentage. This is heavily dependent on both the model and the data used for testing. The TensorFlow Lite library has less functionality and the model may not perform as well as the model present on the desktop TensorFlow version [10]. Additionally the accuracy of the model may also vary depending on the quality of the microphone of the mobile device, with older phones and hence older microphones recording bird sounds with a worse quality. This degraded quality recording might impact the model's classification accuracy. The quality of the recording is dependent on the user and is out of the project's control. As a result of these factors, the accuracy of the desktop TensorFlow model will be the focus and it will be assumed that the mobile variant performs with similar accuracy.

5. PROJECT MANAGEMENT

The Work Breakdown Structure (WBS) for the project is present in Appendix B in Figure 5. It breaks up project tasks under four main components such as Data, Machine Learning Model, Mobile Application and Documentation. The project takes place over seven weeks, beginning on the 16th of July 2018 and ending on the 3rd of September 2018. The interviews and presentations are conducted on the 13th of September. The project will be hosted on Github as this allows easy access and version control. Weekly progress meetings with a supervisor are recommended as this provides iterative feedback and can help guide the development of the project. Both students will keep engineering notebooks for the duration of the project.

Table 1 shows the durations for the different tasks in the project. It is evident that the machine learning component will take the most time due to to the fact that the model training is an iterative process and relies on numerous input data and defined hyperparameters. Some tasks may be run in parallel by dividing the work among the students as indicated by the letter sub-index. The other tasks will be done by both students as these are potentially harder and critical for the project's success. Each task has been given a "worst case" time to account for any difficulties that may arise and add onto its duration. The documentation component runs in parallel with the entire project. Reasoning for this is that students will record important discoveries in their engineering notebooks which will then be used in the final report. Additionally, the poster must be ready before Open Day (28-08-2018) so it is advisable to do that part of the documentation a week prior, in parallel with the task assigned to that duration. The interviews and presentation slide-show can be developed and prepared for after the seven weeks.

Table 1: Time schedule for tasks of the project

Component	**Order**	**Task**	**Time**
Data (1 Week)	1	Data Acquisition	1 Day
	2	Data Processing	3 Days
	3A	Data Splitting	1 Day
Machine Learning Model (4 Weeks)	3B	Setup programming environment	1 Day
	4	Implement Neural Network models	18 Days
	5	Model comparison	2 Days
Mobile Application (2 Weeks)	6A	User interface design	1 Day
	6B	Port model to mobile	3 Days
	7	Code the mobile implementation	4 Days
	8	Database design and implementation	1 Day
	9	Test application	2 Days

6. CONCLUSION

This document has explored the possible development of a real time bird call classifier. By utilizing the power of machine learning, a classifier can be trained on numerous South African bird sounds and then ported to a mobile device that can be used in the field. Additionally, this application will allow for uploading of identified bird sounds recorded by a user to an online database with classification information and location data. The implementation plan was detailed and the testing methodology was described. A successfully implemented model should have a classification accuracy of at least 80%. The estimated time management of the project was also presented as the project must be completed within a span of seven weeks.

REFERENCES

[1] M. Hodon, P. Sarafin, and P. Sevcik. Monitoring and recognition of bird population in protected bird territory. In *2015 IEEE Symposium on Computers and Communication (ISCC)*, pages 198–203, July 2015.

[2] E. Cakir, S. Adavanne, G. Parascandolo, K. Drossos, and T. Virtanen. Convolutional recurrent neural networks for bird audio detection. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1744–1748, Aug 2017.

[3] Q. Kong, Y. Xu, and M. D. Plumbley. Joint detection and classification convolutional neural network on weakly labelled bird audio detection. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1749–1753, Aug 2017.

[4] Wouter Gevaert, Georgi Tsenov, and Valeri Mladenov. Neural networks used for speech recognition. 20, 01 2010.

[5] Parwinder Pal Singh. An approach to extract feature using mfcc. 4:21–25, 08 2014.

[6] Adi Bronshtein. Train/test split and cross validation in python. https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6. Accessed 15/07/2018.

[7] Ksenia Sorokina. Image classification with convolutional neural networks. https://medium.com/@ksusorokina/image-classification-with-convolutional-neural-networks-496815db12a8. Accessed 15/07/2018.

[8] Adit Deshpande. A beginner's guide to understanding convolutional neural networks. https://adeshpande3.github.io/A-Beginner Accessed 15/07/2018.

[9] A. Graves, A. r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, May 2013.

[10] TensorFlow. Introduction to tensorflow lite. https://www.tensorflow.org/mobile/tflite/. Accessed 15/07/2018.

[11] Superpowered. About superpowered. https://superpowered.com/about-superpowered. Accessed 15/07/2018.

[12] Google. Maps sdk for android. https://developers.google.com/maps/documentation/android-sdk/intro. Accessed 15/07/2018.

[13] Amazon. Amazon s3. https://aws.amazon.com/s3/. Accessed 15/07/2018.

Appendix

A JSON Response from Xeno-Canto API

```
{
    "numRecordings": "6123",
    "numSpecies": "560",
    "page": 1,
    "numPages": 13,
    "recordings": [
        {
            "id": "46725",
            "gen": "Struthio",
            "sp": "camelus",
            "ssp": "",
            "en": "Common Ostrich",
            "rec": "Derek Solomon",
            "cnt": "South Africa",
            "loc": "Hoedspruit",
            "lat": "-24.3834",
            "lng": "30.9334",
            "type": "Call",
            "file": "//www.xeno-canto.org/46725/download",
            "lic": "//creativecommons.org/licenses/by-nc-nd/2.5/",
            "url": "https://www.xeno-canto.org/46725",
            "q": "B",
            "time": "07:00",
            "date": "2010-02-09"
        },
        {
            "id": "208128",
            "gen": "Struthio",
            "sp": "camelus",
            "ssp": "",
            "en": "Common Ostrich",
            "rec": "Jeremy Hegge",
            "cnt": "South Africa",
            "loc": "Mmabolela Reserve, Limpopo",
            "lat": "-22.6085",
            "lng": "28.2996",
            "type": "call",
            "file": "//www.xeno-canto.org/208128/download",
            "lic": "//creativecommons.org/licenses/by-nc-sa/4.0/",
            "url": "https://www.xeno-canto.org/208128",
            "q": "C",
            "time": "06:00",
            "date": "2014-11-21"
        },
        .
        .
        .
}
```

B Work Breakdown Structure

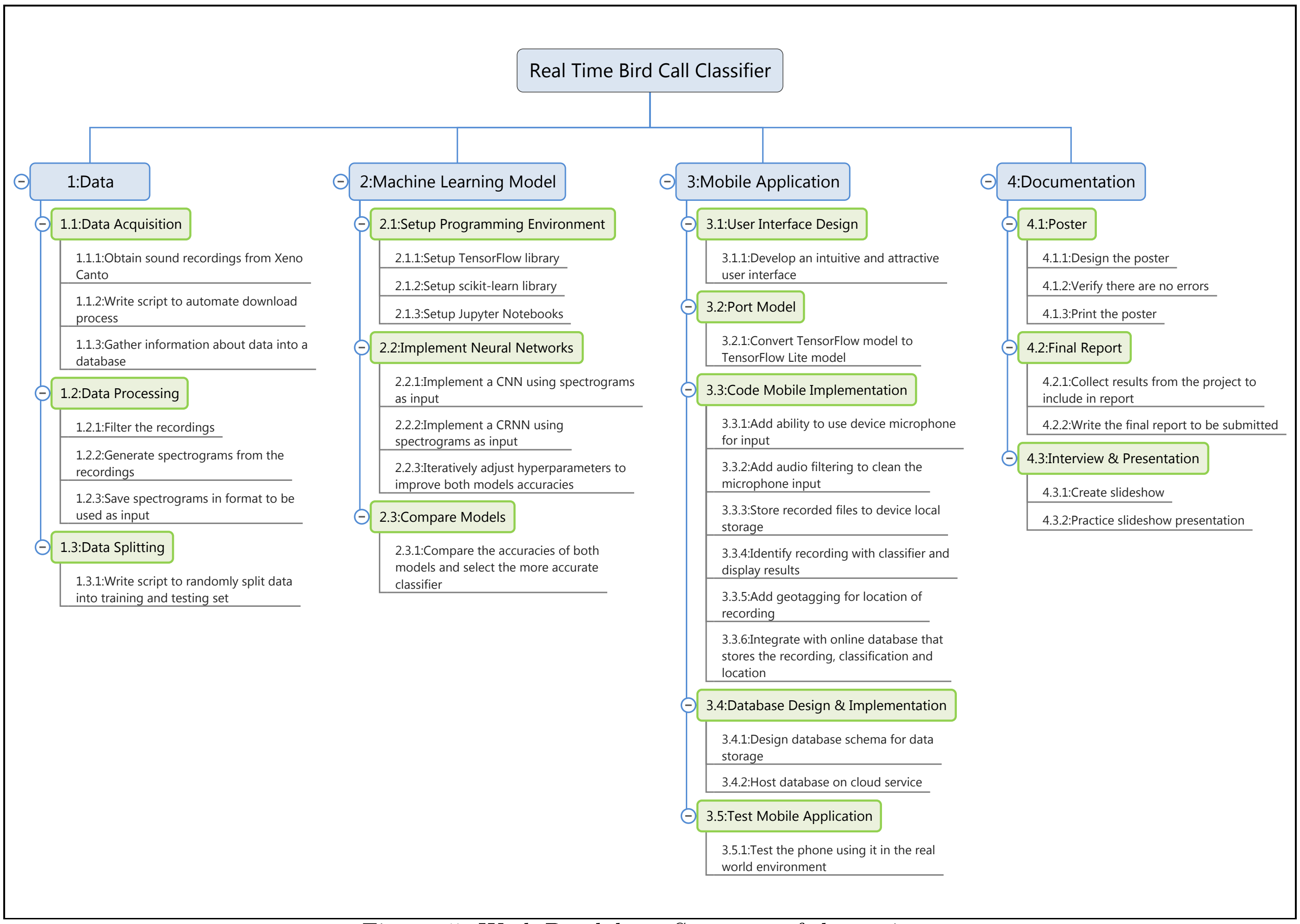
Real Time Bird Call Classifier
1:Data
1.1:Data Acquisition
1.1.1:Obtain sound recordings from Xeno Canto
1.1.2:Write script to automate download process
1.1.3:Gather information about data into a database
1.2:Data Processing
1.2.1:Filter the recordings
1.2.2:Generate spectrograms from the recordings
1.2.3:Save spectrograms in format to be used as input
1.3:Data Splitting
1.3.1:Write script to randomly split data into training and testing set
2:Machine Learning Model
2.1:Setup Programming Environment
2.1.1:Setup TensorFlow library
2.1.2:Setup scikit-learn library
2.1.3:Setup Jupyter Notebooks
2.2:Implement Neural Networks
2.2.1:Implement a CNN using spectrograms as input
2.2.2:Implement a CRNN using spectrograms as input
2.2.3:Iteratively adjust hyperparameters to improve both models accuracies
2.3:Compare Models
2.3.1:Compare the accuracies of both models and select the more accurate classifier
3:Mobile Application
3.1:User Interface Design
3.1.1:Develop an intuitive and attractive user interface
3.2:Port Model
3.2.1:Convert TensorFlow model to TensorFlow Lite model
3.3:Code Mobile Implementation
3.3.1:Add ability to use device microphone for input
3.3.2:Add audio filtering to clean the microphone input
3.3.3:Store recorded files to device local storage
3.3.4:Identify recording with classifier and display results
3.3.5:Add geotagging for location of recording
3.3.6:Integrate with online database that stores the recording, classification and location
3.4:Database Design & Implementation
3.4.1:Design database schema for data storage
3.4.2:Host database on cloud service
3.5:Test Mobile Application
3.5.1:Test the phone using it in the real world environment
4:Documentation
4.1:Poster
4.1.1:Design the poster
4.1.2:Verify there are no errors
4.1.3:Print the poster
4.2:Final Report
4.2.1:Collect results from the project to include in report
4.2.2:Write the final report to be submitted
4.3:Interview & Presentation
4.3.1:Create slideshow
4.3.2:Practice slideshow presentation

Figure 5: Work Breakdown Structure of the project.

D Meeting Minutes

The group meetings were carried out in the presence of everyone in the Telecommunication Research Group for the first week, but thereafter was broken up into 2 subgroups due to the size. These minutes contain the relevant information to the Real Time Bird Call Classifier Project. The meetings had the agenda that students would present the progress of their project and any problems encountered. The rest of the groups and supervisors would then ask questions or recommend solutions. The meetings were chaired and minutes scribed by a different group each week.

Meeting:	**Telecoms Lab Project Group Meeting**
Date:	Thursday 19th July 2018, 12:30 to 13:30
Location:	Jackson Room, Wits School of Electrical and Information Engineering
Chairperson:	Professor Fambirai Takawira
Secretary:	Lara Timm

- How many bird species will you try and classify? 560 species.
- Recommendation to scale down to 10 or less classifications to simplify the neural network
- Why neural network over other classification techniques? The project description suggests a neural network solution. Also, there is lots of existing research in this field.

Meeting:	**Telecoms Lab Project Group B Meeting**
Date:	Thursday 26th July 2018, 12:30 – 13:18pm
Location:	Jackson Room, Wits School of Electrical and Information Engineering
Chairperson:	Amprayil Joel Oommen
Secretary:	Leantha Naicker

- Decreased number of species in training set from 560 to 10.
- Began machine learning; making use of existing libraries to do MFCC and may not apply filter to samples as literature shows that it has no known benefits.
- How many samples are there per species (Ms De Mello Koch)? Lowest of the 10 species has 40 samples.
- What is the speed of the data training (Ms De Mello Koch)? 754 seconds per epoch. This is probably due to the network speed and size of the test data. Train on desktop for better performance but current time seems reasonable.
- Look at MFCC coefficients change to improve accuracy (Prof Takawira).
- Can training sets have samples of equal length (Dane Slattery)? It is not feasible to break current samples into equal length tracks as the data is not at consistent time intervals.

Meeting: **Telecoms Lab Project Group B Meeting**
Date: Thursday 2nd August 2018, 12:30 – 13:10pm
Location: Jackson Room, Wits School of Electrical and Information Engineering
Chairperson: Sara Sasha Berkowitz
Secretary: Arunima Pathania

- Improved the spectrogram issues by applying a filter and limiting the number of MFC coefficients on the y-axis.
- Implementing VGG which is a version of CNN.
- Started the training on two birds and got an accuracy of 80%. On increasing the number of species the accuracy went down to 65%.
- The issue in the process is that splitting the data in 4 second intervals now has certain clips with only noise. Currently working on getting the SNR and eliminating the noise from the database.

Meeting: Telecoms Lab Project Group B Meeting
Date: Wednesday 8th August 2018, **12:15 – 13:05**
Location: Convergence Lab Wits School of Electrical and Information Engineering
Chairperson: Iordan Tchaparov
Secretary: Kavilan Nair

It was decided in the previous meeting that the following 2 meetings would be practical in that we would go to the work stations as a research group and 3 groups would present their projects. The groups assigned this week were Sasha and Arunima, Thembelani and Matsobane and Leantha and Joel. The real bird bird classifier project was not discussed and was scheduled to be presented for the next week.

Meeting: Telecoms Lab Project Group B Meeting
Date: Wednesday the 16 August 2018, **12:15 – 13:05**
Location: Control Lab Wits School of Electrical and Information Engineering
Chairperson: Matsobane Maruma
Secretary: Thembelani Bheza

- Iordan demonstrated the MFCs and the classifier on data that was already trained.
- Demonstrated the training and validation accuracy after the transformations were performed. The accuracy increased as the algorithm trained, but there was overfitting. The training data could be classified really well, but the testing data did not produce good results.
- Had to apply different techniques to reduce how well it classifies training data so that the training and validation accuracy curves are closer together.
- The training accuracy was about 90% and the validation accuracy, which is what we really want was about 68% which is not good. We will look into techniques to try and fix that.

- Demonstrated the downsides of the algorithm and discussed how they were going to refine the training in order to improve results.
- Kavilan demonstrated the mobile app which was the second part of the project.
- Build a Python API which took the URL to the sound, generates the MFCC, does the classification and produces the output. For this to work the device needs an internet connection.
- Kavilan demonstrated the mobile app, showing how you can record the sound. The app uploads the clip to a bucket and then the classification is triggered.
- Still working on optimisation by doing the machine learning and classification onto the device.
- We want to add geolocation, user login and link to the sound clip.

Questions?
60% is not is not good enough for examination. It can be used for open day, but you need to improve it because you have to demonstrate that you were able to meet the specification which is 80%. If you cannot get it to 80% then what you might try and do is classify to a group of birds instead of classifying to the actual bird. (Professor Takawira)

Meeting:	**Telecoms Lab Project Group B Meeting**
Date:	Thursday 23rd August 2018, **12:15 – 13:05**
Location:	Jackson Room
Chairperson:	Nick Raal
Secretary:	Dane Slattery

- IOS app which runs the Neural Network locally and connects to cloud hosted API implemented successfully
- Achieved 75% accuracy on test data set.
- Able to get more accuracy with more training, but time is running out.
- Poster is under construction.
- Still using 10 birds.

Questions
- Sasha - “You could train on Prof. Otoo’s cluster, which has many CUDA cores.”
    - Okay, although we do have our own GPU enabled PC
