Real Time Bird Call Classifier

Iordan M. Tchaparov **Group: 18G05**
1068874 **Project: 18P07**

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

Abstract: This document details the design and implementation of a real time bird call classifier. The classifier is 77% accurate for 10 bird species and can provide results within 30 seconds provided the audio recording is long. The classifier is a Convolutional Neural Network coded in Keras. The system involves both a server and iOS application that communicate. The iOS application allows audio recording as well as audio and data storage on Firebase. The server is a dockerized Ubuntu instance hosted on DigitalOcean that generates Mel-Frequency Cepstral Coefficients from the audio and classifies them, returning the result to the iOS application. The Python language is used in all instances apart from the iOS application which is coded in Swift. The total cost for the system was R294.62

Key words: Convolutional Neural Network, Mel-Frequency Cepstrum Coefficients, Mobile Application, Real Time

1. INTRODUCTION

The ability to autonomously classify sounds through the use of technology greatly minimizes manual processing and storage needs as well as providing instant information to a user. Shazam is an example of this whereby part of a song is recorded and it is then identified and displayed in real time. This document presents the implementation of an application that classifies bird calls in real time on a mobile device. This is enabled through extraction of Mel-Frequency Cepstral Coefficients (MFCCs) from recorded audio which is input into a Convolutional Neural Network (CNN) as an image for classification. The document contains the following sections: background, system overview, implementation, results, analysis & recommendations and conclusion. This project was undertaken by two students and their contributions and a reflection of the group work is present in Appendix A.

2. BACKGROUND

The success criteria of the project is re-examined and the analysis of existing literature as well as research of technologies to implement this system is presented. The project specification and project plan can be found in Appendix B and Appendix C respectively. The requirements and assumptions presented in Appendix C remain unchanged.

2.1 Success Criteria

The system implemented must make use of a CNN with a classification accuracy of 80% and must be able to return results in real time. A limit of 30 seconds is imposed for the system to be considered real time.

2.2 Literature Review

Avian species can produce audio in vocal and non-vocal forms with the vocal category being further subdivided into calls and songs [1]. The real time classifier will be focused on bird calls as they are brief and more commonly encountered in the environment compared to bird songs which are longer and used to indicate readiness for courtship during specific seasons [2].

A method to monitor avian biodiversity in a habitat involves capturing calls on devices that store audio for a period of time [3]. Researchers remove the storage, studying and labeling the data manually. This comes at the cost of storage space and long classification times due to being manual. Challenges have emerged to simplify and decrease the amount of data needed to be classified by recognizing whether an audio recording contains a bird call or not [4].

Further challenges have emerged focused on the classification of the bird calls as opposed to their recognition. The winning submission to the BirdCLEF 2016 Challenge provides good insights as to a CNN architecture for detection of features in bird calls that aid classification [5]. The methods presented in [1, 5–7] for both challenges provide valuable insights in developing a classifier using neural networks.

An assumption is made: all input audio to the system will be a bird call. This assumption is made to simplify the training of the neural network as then it will only need to be trained solely with bird calls and not with other sounds (cars, sirens, human speech) to be able to detect when an input sound is not a bird call. This is left as a future recommendation to expand the system.

A common method for classification is to use images. Time-domain representations of audio signals are not distinct as different signals can look like one another and noise can distort the shape and representation. Filtering audio is a good method to reduce the impact of noise. Bird calls are typically in the range of 20 Hz to 12 000 Hz which makes filtering to allow frequencies between those ranges a good method to clean the data [8]. MFCCs have proven an effective feature extraction method to be used as input to train a neural network [9].

There are two standard types of neural networks: Convolutional and Recurrent. Convolutional Neural Networks work best for image and video classification while Recurrent Neural Networks work best for text and speech analysis [10]. A combination of the two has been used but it was noted that the increase in accuracy was low and will not be considered [1]. A CNN is therefore picked for its lower complexity and ease of implementation.

Many CNN architectures exist with different amounts of accuracies based on image classification such as that of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [11]. The competition is focused on object detection and classification of complex images which requires that the models employ deeper learning. If the input is sized correctly and is less complex in nature, a smaller CNN can still perform adeptly.

After training the CNN model, it will need to be executed on a mobile device to return real time classification results. The two most used operating systems are Android and iOS and both have frameworks that support machine learning. Android has TensorFlow lite and iOS has Core ML.

Research was conducted to see the feasibility of doing the signal processing (MFCC feature extraction) on a mobile device. Python was chosen as the programming language due to its extensive library support for the MFCC extraction, plotting and saving of an image. Android was first targeted using a framework called Kivy which attempted to bundle Python and its libraries as an application. The two libraries needed could not be ported and this approach was abandoned. A mobile IDE, Pydroid3 was then tried and allowed Python libraries to be installed. The problem was that device permissions for the microphone and internal storage could not be enabled, MFCC Plotting took too long and the IDE crashed before it could be completed. After this, iOS was investigated and it was discovered that PySwift existed, a framework that would embed Python into Swift, the language of iOS applications. This framework only allowed simple Python functions to be embedded and the high level libraries could not be embedded.

The real time aspect came from the idea that most people have mobile devices making them the perfect medium to run the classifier on. Since the signal processing could not be done locally on the phone, it was decided that a mobile application that communicates with a server would be the best alternative. While this allows for more processing power due to occurring on the server side, the downside of this is that the device would need an Internet connection to communicate with the server, incurring data charges. iOS was decided to be used as there was access to iPhone mobile devices for testing.

3. SYSTEM OVERVIEW

The different components of the system along with their interactions are shown in Figure 1.

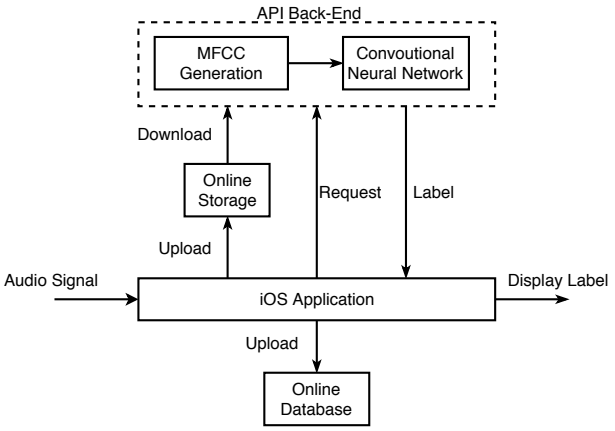

Figure 1 : System block diagram

The input to the system is an audio signal in the form of bird call, recorded using the iPhone's microphone. This audio recording is then uploaded to an online storage. After it has been uploaded, the application makes a request to the API back-end which contains the MFCC generation and the trained CNN model. The API receives the request which contains a link to the sound file that has just been uploaded and starts downloading it on the server. Once downloaded, the MFCCs are generated and plotted as an image which is then passed as input to the CNN model. The model classifies the image and a classification label (text identifying the bird type) is returned to the application. The application now stores that label, the GPS location of where the audio was recorded and the storage link to the audio file to an online database. This last step only happens if the user decides that they want to keep the classification data and audio recording.

4. IMPLEMENTATION

4.1 Data Acquisition

Neural networks require labeled data that they must be trained with to develop a model which can classify new data. A larger dataset results in a more accurate model as there is more for the model to learn as well as more variations to accommodate for. To alleviate the time consuming and expensive task of recording bird calls, a crowd source of bird call data was obtained from Xeno-Canto, an open platform database that birders can contribute to. The database was filtered to avian species in South Africa which were downloaded through the use of a Python script making calls to the API that Xeno-Canto provides. A total of 590 different avian species were obtained but the amount of recordings per species varied. The length of

the recordings were also not constant, ranging from 10 seconds to 2 minutes. Training the CNN with this uneven distribution of recordings would result in a biased model that would most likely classify input sounds as the birds that had more recordings. Processing this amount of data would also have taken a long time due to the hardware limitations. It was decided to use the top 10 birds with the most recordings. This resulted in 661 audio samples of varying lengths and sampling rates (either 44.1 kHz or 48 kHz). This dataset was still unevenly distributed as the birds still differed in the amount of recordings.

To increase the dataset, it was decided to cut these audio recordings based on the bird calls. A peak detection algorithm was considered to cut the audio into bird calls and periods of silence. Due to noise in the recordings, it was not able to accurately cut the bird calls out from the entire recording. An alternative method was used which cut all recordings into segments of 4 seconds, with any end segments below two seconds being completely discarded. Mean analysis was then done which compared the total mean of the full audio signal compared to the mean of the 4 second segments of that same audio recording. Any segment that had a mean below 25% of the total mean was removed and considered noise or silence. This removed noise and periods of silence where no bird calls were present. After this process, one bird species had 356 recordings which was the lowest compared to the other birds. To ensure the CNN remained impartial, the 4 second audio segments were randomly removed from the other nine bird datasets until they all had 356 records each. These were to be used to train the CNN model.

4.2 Data Processing

The steps required to obtain the MFCCs of an audio file are presented as follows [12]:

The audio must be framed. The 4 second audio recordings were framed by taking segments containing 1024 samples which correlates to 23.2 ms and 21.3 ms of the total audio for sampling rates of 44.1 kHz and 48 kHz respectively. These had overlaps of 50% (hop length of 512 samples). A windowing function is also applied to the frame, commonly a Hamming window. Padding is incorporated so that each frame will contain 1024 samples.

The Discrete Fourier Transform (DFT) must now be applied to each frame of the audio signal. The result of this is the power of the audio recording at different frequencies. This is done through equation (1):

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{\frac{-j2\pi kn}{N}} \tag{1}$$

Where $N = 1024$. The frequency domain must now be mapped onto the Mel-scale using equation (2) where f is frequency in hertz.

$$M(f) = 1127 \cdot ln(1 + \frac{f}{700}) \tag{2}$$

Logarithms must be taken of the power of the DFT and then the Discrete Cosine Transform (DCT) in equation (3) must be applied to the result to obtain the MFCCs.

$$MFCC = \sum_{n=0}^{N-1} X[k] \cdot cos\Big[\frac{\pi}{N}(n + \frac{1}{2})k\Big] \tag{3}$$

Only the first 13 coefficients are used for their distinctness. MFCCs above the first 13 included the delta and delta-delta coefficients which are not as distinct per audio signal, requiring a deeper network and more time to learn. Librosa was the chosen Python library to implement the MFCC extraction. These were plotted and saved with dimensions of 1400 by 400. These dimensions were not important as the images could be resized when input in the CNN. The filtering from 20 Hz to 12 000 was also implemented by Librosa. An MFCC plot is present in Figure 2. The y-axis has 13 bands for the 13 coefficients and x-axis is the length of time, usually 4 seconds for the majority of clips unless it was an end segment clip. The MFCCs are not labeled to avoid making the CNN learning extra features such as numbers or labels which do not aid in the classification.

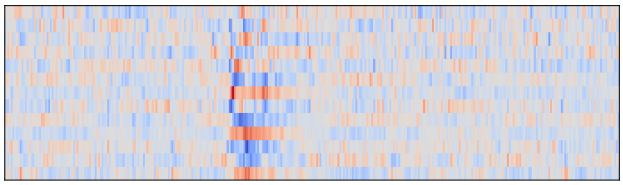

Figure 2 : The MFCC plot of the Andropadus Species

4.3 Machine Learning

Images were resized to a resolution of 120 by 84 with a depth of 3. This depth corresponds to the RGB colour scheme, with each dimension corresponding to a different colour. Normalization between 0 to 1 of the RGB colours is applied to allow the CNN to do calculations faster as the values would be smaller. A split of 80:20 was used to separate the 3560 image dataset into training set and testing set. The splitting of these files is randomly chosen to be fair, however it could still result in more of a specific bird call being in one dataset than another.

The classification labels for the avian species were created using the *LabelBinarizer* function from the scikit-

learn Python library which does one hot encoding of the 10 species. That means that arrays of 10 values are created with a 1 corresponding to the correct species of bird for that specific image and a 0 everywhere else. Data augmentation was incorporated to add a some regularization to the model as well as to increase the dataset. The data was augmented with width and height shifts of 15. The CNN is a supervised model as it is given both the images and their labels for classification.

The different layers of the CNN and their descriptions can be found in Table 1. The training was done on an Nvidia GTX970 graphics card that had 4 GB VRAM.

Table 1 : Layers of the CNN architecture

Layer	Description
Input	84x120x3 Size
2D Convolution	32 Filters
2D Convolution	32 Filters
Max Pooling (3 x 3)	28x40x32 Output size
2D Convolution	64 Filters
2D Convolution	64 Filters
Max Pooling (2 x 2)	14x20x64 Output size
2D Convolution	128 Filters
2D Convolution	128 Filters
Max Pooling (2 x 2)	7x10x128 Output size
Flatten	8960x1 Output size
Dense	1024x1 Output size
Dense	10x1 Output size
Activaton	softmax

Filter sizes of 3 by 3 were chosen as the pattern to detect was relatively smaller compared to the overall image. Overfitting is the phenomena whereby a CNN trains to specifically classify its training data and is not general enough to classify the testing data, or completely new input data. Regularization is a method which aims to prevent overfitting by applying penalties to the learned weights of a CNN. Dropout is a form of regularization and every layer in the network employs a dropout rate of 50% except the Dense 1024 layer which employs a rate of 80%. The Rectified Linear Unit function (Relu) is used as the activation functions of each layer. Adam optimizer was used due to being more efficient than optimizers AdaGrad and RMSProp. The Loss function was chosen as categorical cross-entropy and softmax activation is utilized at the Dense 10 layer to determine the probability of each label for an image. A batch size of 5 and a learning rate of 1e-5 was used as it allowed finer tuning through the use of Adam. Higher learning rates did not allow the CNN model to converge past set points and lower ones did not converge at all. Checkpoints were enabled to save the weights of the model each time a new high testing accuracy value was obtained. All training was done over 3000 epochs. This amount was arbitrarily chosen. The CNN implementation was done using Keras.

4.4 Mobile Application

The mobile application was implemented in the Swift programming language to enable it to run on the iOS mobile operating system. Device permissions such as using the microphone and GPS location were enabled as they were needed to record audio and the location of where the recording was taken. As discussed in Section 2., it was deemed infeasible to do the signal processing and MFCC generation on the mobile phone as it took too much processing power as well as a lack of tools without complete re-implementation which would have taken too long.

The server is a dockerized container running Ubuntu 16.04 with 4 GB RAM and 25 GB storage hosted on DigitalOcean. The container is running a Python script coded with the aid of Flask to act as a RESTful API. This means it acts as an end point that the mobile application sends POST requests with URL links to. The API receives this request and uses the URL, which points to an uploaded audio file on the online storage, to download it. Once downloaded, the MFCC generation from the audio is done and an image is created. The CNN model is then loaded into memory, the image resized and input into the model. The classification label is obtained and sent back to the mobile application where it is displayed to the user. The server approach allows more processing power at the cost of Internet bandwidth to send and receive information. The image is not sent back and classified locally on the phone for two reasons: older devices could struggle to use the Core ML model if it was run locally on the mobile device and the bandwidth consideration of sending a text compared to an image.

Firebase was used to facilitate user login using email accounts. User authentication was required as a history of classification data is available to users that choose to upload their classification results and audio to the database. The Firebase file storage bucket was used for the audios recorded by the device while the database was a NOSQL database which stored the classification label, URL to the audio file, the user ID and the GPS coordinates of where the recording was taken.

5. RESULTS

Each epoch took approximately 14 seconds to process all the images, taking a total of close to 12 hours to train for 3000 epochs. The results of the training of the CNN are present in Figure 3 and Figure 4. The training accuracy is a measure of the model's ability to classify images, when the images and labels have already been shown while the testing accuracy is the model's ability to classify images that are unseen. Accuracy is a measure of the number of correct labels assigned to input images over the total number of images. The testing accuracy is the more useful metric.

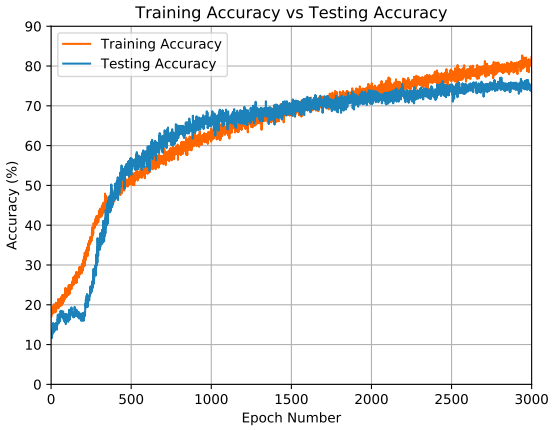

Figure 3 : CNN model's accuracies for the 2 datasets

Figure 3 shows that the model struggles to correctly classify the images in the testing dataset from epoch 0 to 500 compared to the training accuracy. This can be attributed to the augmentation of each batch of images that the CNN is trained on, as they differ from the testing dataset. From epoch 500 to about 1750, the model develops a good classification scheme resulting in high testing accuracies for all the data, even outperforming the training accuracies, indicating good generalization to data it has never seen before. From epoch 1750 upwards, it can be seen that slight overfitting occurs as the training and testing accuracies begin to diverge. The shape does not distinctly level off so it is assumed that with further training that the model accuracies could improve. At the end of 3000 epochs, the highest training accuracy achieved was 83% while the highest testing accuracy was 77%.

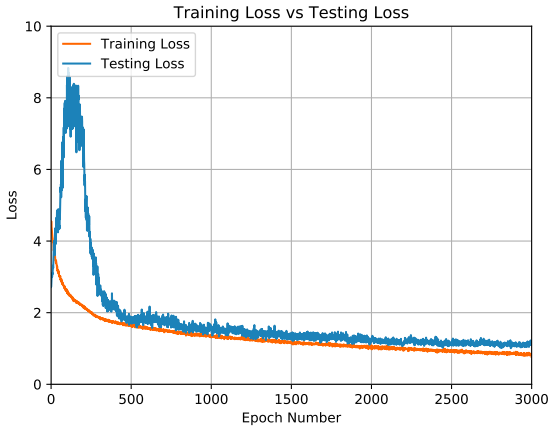

Figure 4 : CNN model's loss for the 2 datasets

The loss is an indication of how far off the classification is compared to the correct label. This is calculated by taking negative logarithms of the classification probability. It can be seen that from 0 to 500 epochs, the classifications are poor resulting in high loss and corresponding to the lower accuracy in the same range in Figure 3. From epoch 500 onwards, the testing loss starts to mimic the training loss. There is an emerging gap between the training and testing loss, indicating overfitting.

The overall system (iOS application with the server) works within 30 seconds depending on two factors: the length of recordings taken by the user and the connection speed of the users phone. As most users will recording only for a single bird call, which is a short duration, it is assumed that the recordings will never be too large and the entire system can produce a result within 30 seconds.

6. ANALYSIS & RECOMMENDATIONS

The desired accuracy of 80% is not obtained but the increasing shape of Figure 3 indicates an upward trend if the training was left for longer. This means that the testing accuracy could converge to 80% or higher. It is recommended to run the training on more power hardware or to split the training up across multiple GPUs to decrease training time.

The entire system was implemented using open source software and therefore there were no costs in terms of implementation. The only costs that exist in this system are those of the mobile data charges for the mobile device and the cost of hosting the Docker container on DigitalOcean costs $20.00 per month (equivalent to R294.62 at the time of writing).

One of the training iterations utilized the same parameters as presented in Section 4., yet the testing accuracy only reached a maximum of 65% while the training accuracy reached a maximum of 80%. The possible cause of this is the data used to train and test the model. If the testing dataset is too different from the training dataset, lower accuracies are obtained. Since the 80-20 split is randomly selected, it indicates that there are some audio recordings that do not accurately relate to the other data. This means that while it has the correct label, the data itself may not be a bird call, instead noise or some other source of sound. Searching manually for this faulty data is infeasible due to the amount of records. An approach to locate the responsible recordings is to retrain the model but using a smaller dataset. Repeatedly train on that dataset, and keep adding back records until the issue occurs again. This will indicate which records are faulty and should be removed as they are drastically different, confusing the training of the model. Further cleaning of data will improve accuracy.

The overfitting to the training data should be reduced so that both training and testing accuracies can be closer together. Either dropout should be increased or alternatively L1 or L2 regularization should be introduced into the CNN.

The system can provide classification results in real time with the mobile application communicating with the server but the requirement of an Internet connection does not make sense as the use case would be in the wild, far away from any communication or Internet. The system should be changed in future to have everything run solely on the mobile device. This means that the MFCC generation needs to be local to the device, requiring implementation of a lot of features from scratch, with minimal external library support.

APIs should be able to handle multiple requests from multiple sources. When testing with two mobile devices, the user who uploaded their audio first would get the classification data while the other user would receive a timeout indication. The reason for this is that when the CNN is loaded into memory, it can only classify one input file at a time before it closes and needs to be loaded again. If another request comes in while the CNN is classifying an image, it is simple rejected and returns a timeout. To avoid this problem, the CNN loading should be parallelized to accommodate multiple simultaneous requests.

The system can be expanded to include more birds through the use of the database as well as user interaction. Implementing a user rating of agree and disagree for classifications allows more data to be labeled. Users disagreeing with the classification can be prompted to provide a manual label. User agreed classification records can be added to the dataset to be used when training the CNN again. Users who disagree and then provide a label will be a source of new data which can be used to classify more species. If the user has sufficient bird call knowledge, they may know exactly which bird species, beyond the current 10, that the call belongs to, which they label manually. This is then added to the dataset the next time the CNN is to be trained. This slowly starts to increase the amount of birds the model can classify. The downside of this approach is its reliance on the knowledge and participation of the users.

7. CONCLUSION

This report has presented the implementation and evaluation of a real time bird call classifier. The system has a classification accuracy of 77% for 10 bird species and works in real time. The system is implemented as a iOS mobile application that communicates with a dockerized Ubuntu instance that acts as the server. Audio is recorded by a mobile device from which Mel-Frequency Cepstral Coefficients are generated and then passed into a trained Convolutional Neural Network to be classified. The system has a low cost of R294.62. While it did not meet the success criteria of 80% accuracy, recommendations are given to reach this desired level.

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] J. Evens. "Whats the difference between bird songs and bird calls?", jul 2001. URL `https://baynature.org/article/whats-the-difference-between-bird-songs-and-bird-calls/`.

[3] A. Digby, M. Towsey, B. D. Bell, and P. D. Teal. "A practical comparison of manual and autonomous methods for acoustic monitoring." *Methods in Ecology and Evolution*, vol. 4, no. 7, pp. 675–683.

[4] 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.

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

[7] 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.

[8] 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)*, pp. 1749–1753. 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] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu. "CNN-RNN: A Unified Framework for Multi-label Image Classification." *CoRR*, vol. abs/1604.04573, 2016.

[11] D. Mishkin, N. Sergievskiy, and J. Matas. "Systematic Evaluation of Convolution Neural Network Advances on the ImageNet." 05 2017.

[12] 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 and division of work

Table 2 : Division of components between partners

Component	Sub Component	K. D. Nair (Partner)	I. M. Tchaparov (Author)
Data Acquisition	Xeno-Canto Download script	✓	
	Cut data into 4 second clips		✓
	Mean analysis		✓
Data Processing	Research	✓	✓
	Filtering		✓
	Mel-Frequency Cepstral Coefficient 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	✓	

Group Reflection

The division of work is present in Table 2. While the work was split, good communication kept both group members up to date with what the other was doing, giving them a strong understanding of the different components of the project. Both members met up early everyday at the university in the computer lab (Dlab) for a brief discussion of the tasks for the day before setting to work on them. The was an attempt to produce work of excellent quality, encouraging one another.

Problems that were encountered by one member were communicated, and thereafter given full attention by both members until they were either solved or a viable alternative was found and implemented. This was done to ensure that the project time-line was not hindered as both members aimed to followed the schedule they had established in the Project Plan (Appendix C).

The members revealed their strengths early in the project, which helped the allocation of project components. This allowed members to work to their strengths, such as Iordan for the data processing and Kavilan for the mobile application.

The machine learning component was completely new to both group members so they shared resources, both research and computational, to establish a theoretical and practical understanding. As this was the biggest learning curve of the project, both members worked on weekends to understand Neural Networks so that they would not fall behind schedule.

The students conducted all meetings and communications with their supervisor in a professional manner. They included one another in email correspondence to make sure that they both were informed of all interactions. When a member could not making a meeting, they would politely excuse themselves ahead of time.

Once components of the project were complete, members presented the work to one another. The component was only considered finished when both members were happy with it. Sometimes disagreements arose as to the method of implementation, but these were discussed respectably until a compromise could be found.

This experience has highlighted the importance of communication, teamwork and planning when undertaking complex projects. Projects become very enjoyable experiences when working with someone who is trustworthy.

B Project Specification

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.

C Project Plan

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

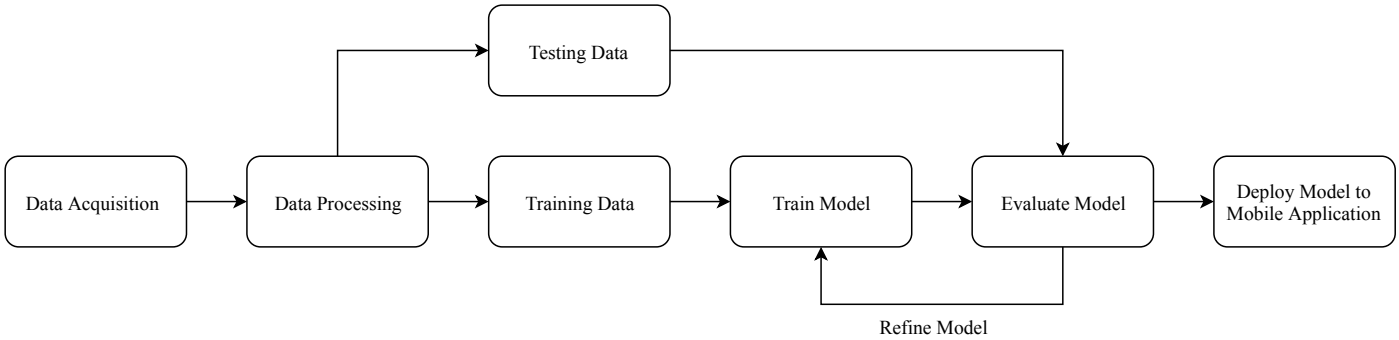

Figure 1: Project process diagram

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.

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,

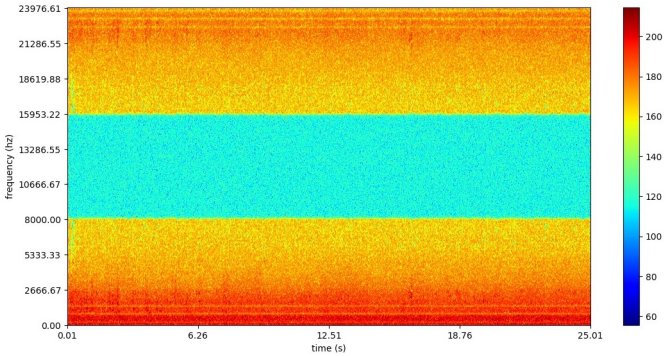

Figure 2: Greater Crested Tern Spectrogram

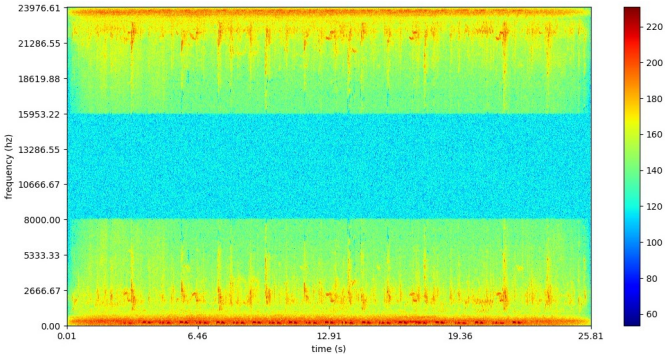

Figure 3: Red Eyed Dove Spectrogram

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.

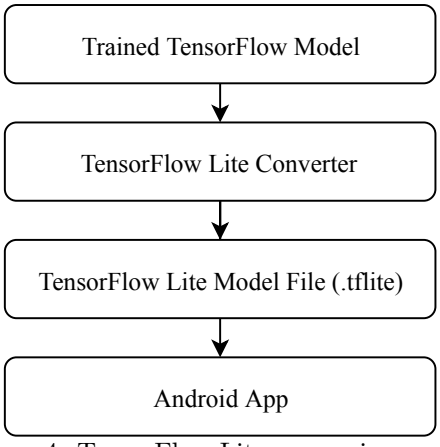

Figure 4: TensorFlow Lite conversion process

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 record-

ing 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

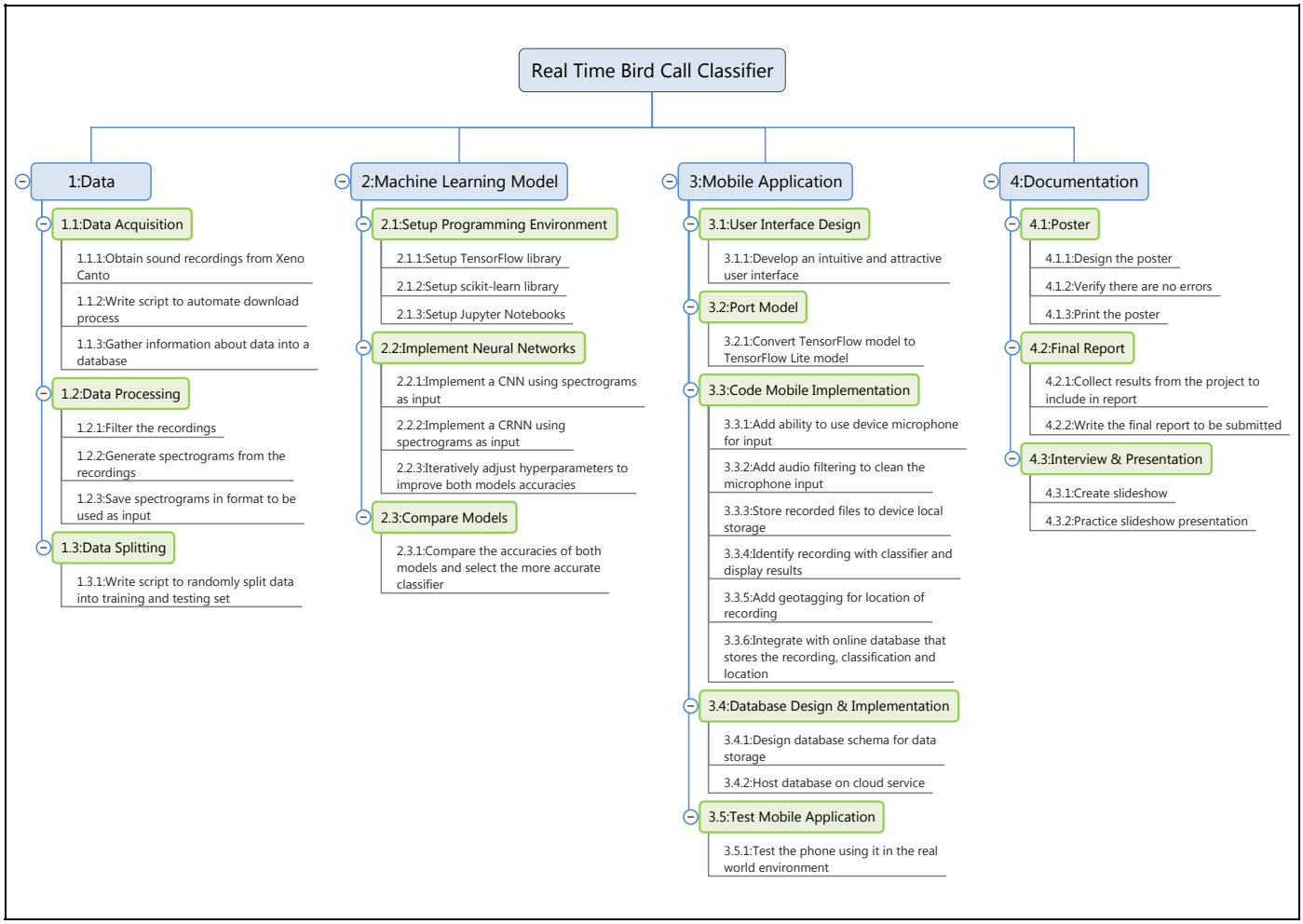

Figure 5: Work Breakdown Structure of the project.

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
