A Mobile-based Image Recognition System for Identifying Bird Species in Kenya

Gideon Mwangi Nyaga

Master of Science in Computer-Based Information Systems

2019

A Mobile-based Image Recognition System for Identifying Bird Species in Kenya

Gideon Mwangi Nyaga

Submitted in partial fulfillment of the requirements for the Degree of

Master of Science in Computer-Based Information Systems at Strathmore University

Faculty of Information Technology

Strathmore University

Nairobi, Kenya

June, 2019

This dissertation is available for Library use on the understanding that it is copyright material and that no quotation from the dissertation may be published without proper acknowledgement.

DECLARATION

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the dissertation contains no material previously published or written by another person except where due reference is made in the dissertation itself.

© No part of this proposal may be reproduced without the permission of the author and Strathmore University

Gideon Mwangi Nyaga

……

10th June, 2019

Approval

The dissertation of Gideon Mwangi Nyaga was reviewed and approved by the following:

Dr. Vincent Omwenga,
Senior Lecturer, Faculty of Information Technology,
Strathmore University

Dr. Joseph Orero,
Dean, Faculty of Information Technology,
Strathmore University

Prof. Ruth Kiraka,
Dean, School of Graduate Studies,
Strathmore University

Abstract

Kenya is world-renown for its wildlife tourism, which attracts many foreigners and is a leading foreign exchange earner. The last few years has seen significant growth in the number of people, called birders, who observe birds for recreation or for citizen science. Kenya in particular is one of the leading birding destinations in the World with about 11% of the World's bird species. A big challenge birders face is correctly identifying the bird species observed. Fine-grained visual classification, and in this case the ability to identify Kenyan bird species, is a challenging task for both humans and machines mainly because of subtle differences between different bird species and strong variety within same species. This research study solved this challenge by developing a mobile-based image recognition system that can identify Kenyan bird species from images.

A deep neural network called a Convolutional Neural Network (CNN), inspired by the human visual cortex, is well suited for image recognition because it is able to handle shifts and distortions in an image well, has fewer trainable parameters and thus uses less memory and training time than a standard artificial neural network (ANN). Further a depth-wise separable CNN is suited for resource-constrained mobile devices as it has lower computational cost as compared to standard CNNs.

This research developed a depth-wise separable CNN model, which was trained using 39,031 labelled bird images via supervised transfer learning. The model achieved a final test accuracy of 97.3%. Consequently, an Android mobile application was developed to consume the resulting model. The model was embedded into the mobile application. Therefore, the user did not need internet to make an inference. The mobile application was able to process an input image containing a bird and identify the bird species. The mobile application was also able to give more details of the identified bird species.

Keywords: fine-grained classification, bird species identification, image recognition, depth-wise separable convolutional neural networks.

Table of Contents

List of Figures

List of Tables

List of Abbreviations/Acronyms

AMT	-	Amazon's Mechanical Turk
ANN	-	Artificial Neural Network
API	-	Application Programming Interface
BaaS	-	Backend-as-a-Service
BRIEF	-	Binary Robust Independent Elementary Features
CNN	-	Convolutional Neural Network
CPU	-	Central Processing Unit
DNN	-	Deep Neural Network
FGVC	-	Fine-grained Visual Categorization
GPU	-	Graphics Processing Unit
PC	-	Personal Computer
PIXEL	-	Picture Element
POOFs	-	Part-based one-vs-one Features
RAD	-	Rapid Application Development
RAM	-	Random Access Memory
ReLU	-	Rectified Linear Unit
SFIT	-	Scale-Invariant Feature Transform
SURF	-	Speeded-Up Robust Features
SVM	-	Support Vector Machine
UI	-	User Interface
UML	-	Unified Modelling Language

Acknowledgements

I thank the Lord God Almighty for giving me the strength, time, resources and good health as I pursed this Masters course. It is by His grace that I have been able to complete this research project. I thank my immediate supervisor, Dr. Vincent Omwenga, for his guidance and support, encouragement and honest critique throughout the dissertation process. I also want to thank Dr. Bernard Shibwabo who guided me in the early stages of my research work. I would like to acknowledge and thank my classmates, MSIS class of 2019 and MSIT class of 2019 for the encouragement, knowledge sharing and even the light moments we shared. I acknowledge and appreciate you all.

Dedication

I dedicate this research project to my wife Amy Kabura and daughters Nissi and Neema Mwangi for their support, encouragement, perseverance and understanding as I was undertaking this research. God bless you all.

Chapter 1
Introduction

1.1 Background

Bird watching, or birding, is a fast growing outdoor activity especially in the Western world of Western Europe and the United States of America (USA). The USA for example has over 45 million birders and they contributed about USD 76 Billion to the US economy in 2016. Birding is popular because it is a convenient, inexpensive and uncomplicated hobby, which can be done right at home. What you need is some interest, time and a good resource to identify the bird species you have seen. Moreover, birding is also attractive because birds are a visual and appealing subject and they occur in virtually all habitats (Sinclair, 1990; U.S. Fish & Wildlife Service, 2016; Why is Bird Watching So Popular?, 2018). One key reason for growth in bird watching numbers has been the impressive growth of wildlife photography. Therefore, there are plenty of bird images available (Marketing Analysis of Bird-Based Tourism, 2011).

The Kenya Bird Map, a joint initiative of the National Museums of Kenya, A Rocha Kenya and the Tropical Biology Association, and implemented in conjunction with Nature Kenya as a project of the Bird Committee of the East Africa Natural History Society, aims to map the current distribution of all Kenya's bird species with the help of volunteer members of the public called citizen scientists, who are generally birders who are keen to contribute their observations to the project (Ornithology Section: National Museums of Kenya, 2018). A species' distribution is the most essential information needed in order to conserve it. Therefore, by pooling the efforts of citizen scientists sharing their bird sighting around Kenya, the Kenya Bird Map shows bird species distribution and number and thus is a powerful tool in conserving Kenya's diverse bird life, bird habitats and identifying bird species under threat (Wachira, Jackson, & Njoroge, 2015).

Smartphone use worldwide and including Kenya has seen incredible growth with mobile penetration at 51% worldwide and 83% in Kenya. Almost all smartphones come equipped with a camera (Zab, 2015). Image Recognition i.e. the process of identifying and detecting an object or feature in a digital image or video has dramatically improved (Brynjolfsson & Mcafee, The Business of Artificial Intelligence, 2017; Image Recognition: MathWorks, 2018). The Merlin Bird ID application by Cornell University, whereas it may identify common birds found in most

habitats worldwide like the glossy ibis, it does not have bird packs for Africa, it only has bird packs for the Americas and Western Europe and employs human-in-the-loop identification methods (Branson, Horn, Wah, Perona, & Belongie, 2014; Merlin Bird ID, 2018).

1.2 Problem Statement

Fine-grained visual categorization of bird species is a challenging task for both humans and machines mainly because of subtle differences between different bird species and strong variety within same species. One central element of birding or bird species conservation is being able to identify a bird correctly. Some distinguishing features are so subtle to distinguish in a quick flash or in poor lighting or may get deformed or simply have wear and tear that only detailed examination of captured specimen can a positive identification be made. Therefore, it is difficult to make correct identification in the field. Nevertheless, most birders will take video or photographs of the bird in order to study the bird later on (How to Identify Birds, 2018; How to Identify Birds by Sounds, 2018; Identification of Birds, 2016; Shaoli & Tao, 2016).

Image recognition, powered by advances in deep neural networks, are now performing many vision related tasks faster and better than humans mainly due to convolutional neural networks (Devikar, 2016). Moreover, there has been growing interest in building small and efficient neural networks called MobileNets that work well in resource-constrained mobile devices (Howard, et al., 2017). In machine learning, having abundant and correct data is more important than having correct algorithms. Therefore, this research focuses on solving the fine-grained visual categorization of bird species by creating an image dataset of Kenyan bird species. The dataset created is used to train a convolutional neural network via supervised transfer learning. The resulting model is embedded into a mobile application, which a user can use to make inferences.

1.3 Specific Objectives

(i) To evaluate the current methods used in bird species identification.

(ii) To analyze machine learning techniques that support image recognition.

(iii) To develop a mobile-based image recognition system that can identify bird species.

(iv) To validate the image recognition system.

1.4 Research Questions

(i) What are the current bird identification methods?

(ii) What are the machine learning techniques that support image recognition?

(iii) How can a mobile-based image recognition system that identifies bird species be developed?

(iv) How can an image recognition system be validated?

1.5 Justification

Birdwatching is a popular hobby and birder numbers are on the increase. The ability to identify a bird is crucial to the success and enjoyment of the birding hobby. Therefore, this research will add to the repertoire of tools that a birder has, making it easy for even a beginner birder to correctly identify a bird.

Bird identification is also very important to any member of the public who wishes to contribute to noble projects like the Kenya Bird Map because the Kenya Bird Map collects bird observation data from volunteer citizen scientists. Therefore, accurate identification of bird species is crucial to the integrity of the data the Kenya Bird Map has on bird distribution and bird numbers because this information is used in conservation efforts of bird species and their habitats. Conservation of bird habitat is important for Kenya's tourism sector, which is Kenya's third largest foreign exchange earner (Wachira, Jackson, & Njoroge, 2015; Okello, 2014). The research seeks to use machine learning to not only aid the birder identify the bird but also eliminate chances of misidentification by training the inference model with a large image dataset so as to increase its accuracy.

Thanks to smartphones and high internet penetration, there are many bird images to be found online. For example the Internet Bird Collection (The Internet Bird Collection, 2018). Therefore, there is a large image dataset available, which is required for effectively training a deep neural network.

This research will create an application programming interface (API) that can be beneficial to other application developers who may wish to add image recognition to their applications or to future researchers in computer vision who can build on this research.

1.6 Scope and Limitation

The research is limited to identifying Kenyan bird species and provides the common name of the birds in English and their scientific name. The research will further use a subset of 50 bird species, which includes the 13 endemic bird species in Kenya, for the proof of concept. The mobile-based image recognition system will be developed using JavaScript and other web technologies but will be compiled and built for the Android mobile operating system only.

Chapter 2
Literature Review

2.1 Introduction

This chapter briefly introduced birdwatching or birding and common methods of bird identification currently in use. The chapter then compared traditional computer vision with deep machine learning and delved into details of convolutional neural networks (CNNs) which are well suited for image processing. Then the TensorFlow and Keras machine learning frameworks were introduced. Finally, the researcher presented a conceptual framework.

2.2 Birding

Birding is the observation of birds in their natural habitats as a hobby or as a citizen scientist (Birdwatching tourism from Europe, 2017; Ornithology Section: National Museums of Kenya, 2018). Birdwatchers can be grouped into three main categories: Hardcore birders or twitchers, enthusiastic birders and casual birders.

Hardcore birders or twitchers are a small group of highly dedicated and competitive birders. They are generally impatient with less-skilled birders and crowds. They travel long distances to see new or rare birds to add to their "life list" of species. They carry their own equipment for example telescopes, binoculars or cameras and are not interested in other activities. They represent about 10% of birders.

Enthusiastic birders are broad-based and knowledgeable nature lovers and thus not just focused on birds. They are interested in other activities but also desire to get a large and diverse bird slit. They represent about 50% of birders.

Casual birders are travellers interested in outdoor and nature-based activities and can be persuaded to include birding as an additional activity. They value accessible places with less effort and more comfort. They represent about 30% of birders (Birdwatching tourism from Europe, 2017; Marketing Analysis of Bird-Based Tourism, 2011).

2.2.1 Bird Identification Methods

The most common method of bird identification is from memory. Birders must familiarize themselves with different species by going through the various birding books,

magazines and guides so that they can recognize a species the first time they see it in the wild. This requires the birder to use any spare time they can get to comb through books. Typically, in the wild a bird is not clearly seen and is often seen for a short period as it darts from one place to another or a feature on the bird may be exaggerated leading to misidentification. Therefore, the birder should aim for sustained and better viewing to properly identify the bird. There are four main keys to identifying a bird visually. They are the bird's size and shape, the colour pattern, behaviour and habitat (Building Skills: The 4 Keys to Bird Identification, 2009; How to Identify Birds, 2018).

When trying to identify a bird the birder should first judge its size and try to work out its general shape, colour pattern, length of peak or tail, shape of wings and every other conspicuous detail. More often than not it takes a combination of features to make a positive identification. It takes time and practise to become proficient at identifying a bird visually (Building Skills: The 4 Keys to Bird Identification, 2009; Sinclair, 1990).

Another method of identifying a bird is by its sound and is referred to as birding by ear. Birding by ear is difficult but it can be rewarding. Usually a birder will hear a bird before seeing it and species that are similar physically can be differentiated by their voice. However, just like identifying a bird visually, birding by ear takes years of learning and experience to master (How to Identify Birds by Sounds, 2018).

The quickest way for a birder to identify a bird is to ask an experienced birder, but the experienced birder must be nearby or be readily available. The next best way is to photograph the bird and then consult books, magazine or guides or consult an expert. Similarly, the birder can write notes and possibly sketch the bird and then consult an expert later on (How to Identify Birds, 2018; Identification of Birds, 2016).

2.2.2 Current Bird Identification Mobile Applications

This research study reviewed Merlin Bird ID and BirdSnap, which are two common and publicly available applications for bird species identification. Merlin Bird ID was built with the help of experts, Mechanical Turkers from Amazon (AMT) and over 500 citizen scientists, who annotated their dataset. They created a dataset with part annotations and bounding boxes. In addition, the application utilizes human-in-the-loop methods to return an identification result to the user. Part annotation is expensive both in terms of time and labour costs and it does not

guarantee an error free dataset. The less experienced or knowledgeable annotators can make mistakes. However, in contrast, it can generate a more accurate dataset if the human annotators are sufficiently knowledgeable in the particular field. Human-in-the-loop methods reduce the user experience since they require the user to perform extra steps in order to get a result (Branson, Horn, Wah, Perona, & Belongie, 2014; Horn, et al., 2015). Figure 2.1 below shows an example of human-in-the-loop methods employed by Merlin Bird ID application.

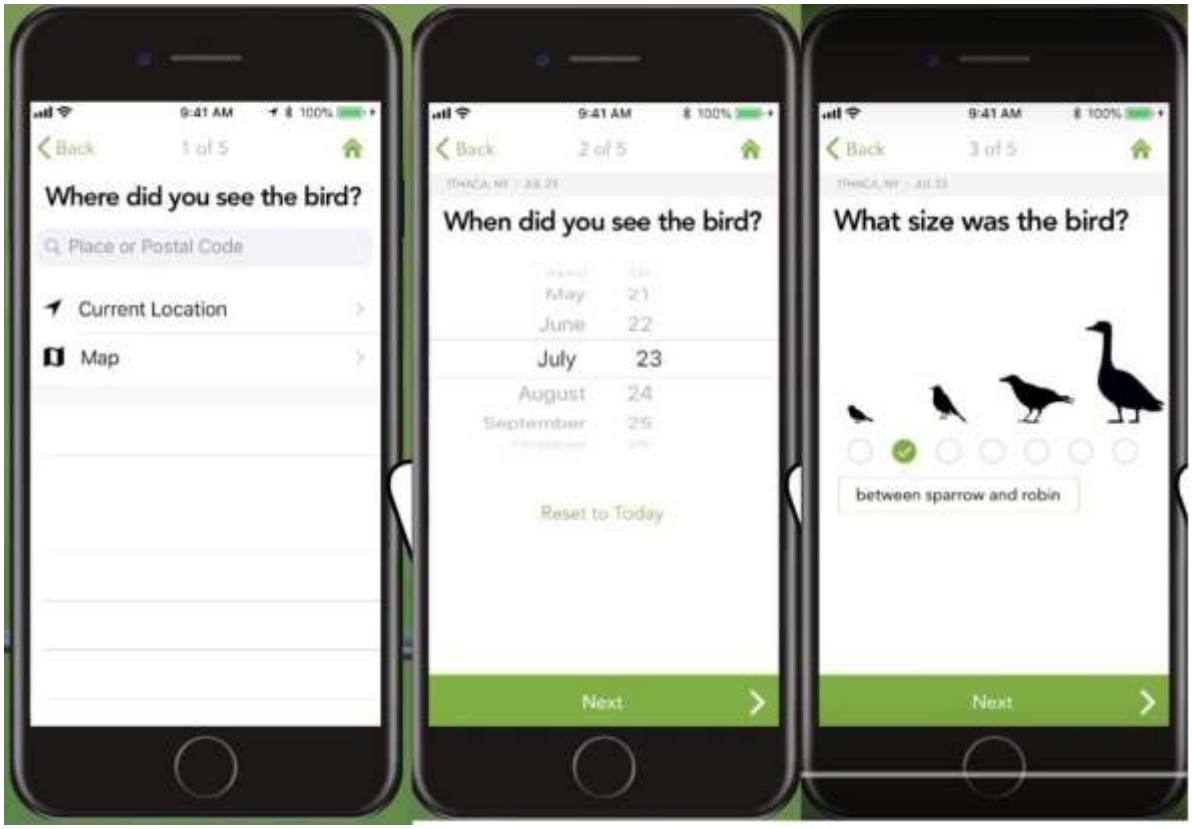


Figure 2.1: Sample of Human-in-the-loop Methods adapted from (Merlin Bird ID, 2018)

BirdSnap introduces a one-vs-most classifier instead of the common one-vs-all classifier. A one-vs-most classifier improves accuracy by comparing the input image with the most likely species instead of all the species. BirdSnap employs Part-based one-vs-one Features (POOFs) Support Vector Machines (SVMs). However, CNNs have been proven to be better in image classification than SVMs (Lee, 2016). Like, Merlin Bird ID, BirdSnap also uses a dataset with bounding boxes and part annotations. Both these applications have image datasets for American and/or European birds. There may be some Kenyan bird species found in these datasets but these

datasets are not necessarily composed of Kenyan birds (Berg, et al., 2014; Wah, Branson, Welinder, Perona, & Belongie, 2011).

2.3 Machine Learning

Machine learning can be defined as a set of methods that can automatically detect patterns in data and then use the uncovered patterns to predict future data or perform some kind of decision making under uncertainty (Robert, 2014). Machine learning technology can be used to identify objects in images, transcribe speech into text, select relevant results and many more aspects of modern society (LeCun, Bengio, & Hinton, 2015). Machine learning has grown mainly due to three factors; increased data, improved algorithms and more powerful computer hardware (Brynjolfsson & Mcafee, What's Driving the Machine Learning Explosion?, 2017).

2.3.1 Traditional Computer Vision

Features are small "interesting", descriptive or informative patches in images. In the past 10-20 years, feature descriptors like Scale-Invariant Feature Transform (SFIT), Speeded-Up Robust Features (SURF), and Binary Robust Independent Elementary Features (BRIEF) among others were used for object detection and used in conjunction with machine learning algorithms like Support Vector Machine (SVM), K-Nearest Neighbour for prediction to solve image recognition problems. These techniques are based on extracting features like edges, corners and colour deemed relevant. It takes a relatively short time to train traditional vision algorithms but they are limited in accuracy and number of objects they can detect. Deep neural networks (DNNs) discovered in 2006, on the other hand, learn features that help differentiate one image from another via an automated procedure using non-linear statistical models. Deep neural networks are algorithms modelled after the human brain. The DNN learns denser and denser, that is more abstract, representation of the training image as you proceed up the architecture. Therefore a DNN requires more computation power, training time and a very large dataset. However, a DNN results in higher accuracy and can detect more objects. Figure 2.2 below illustrates the differences between the two techniques (Lee, 2016).

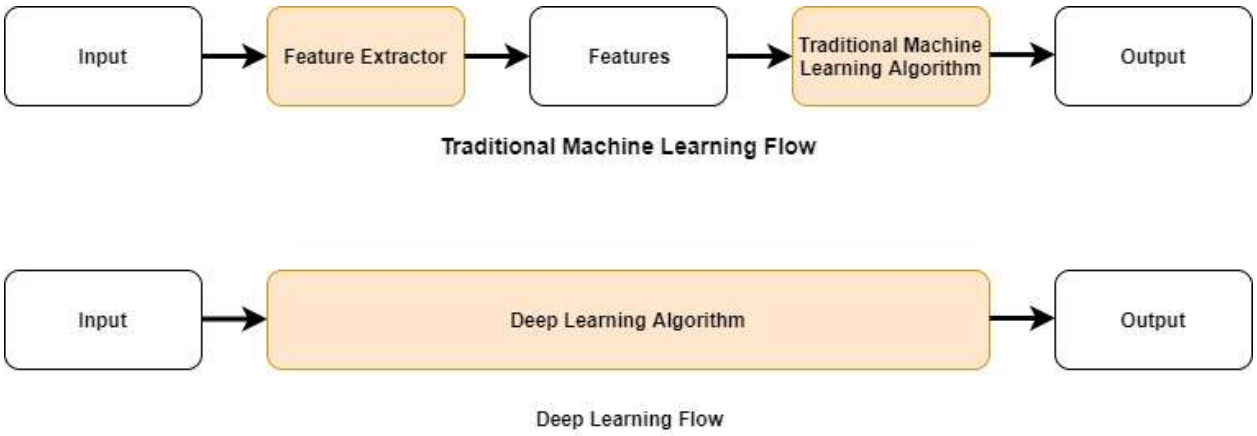


Figure 2.2: Traditional Machine Learning compared with Deep Learning adapted from (Lee, 2016)

Conventional machine-learning techniques are, therefore, limited in their ability to process natural data in their raw form like pixel values of an image. Representation learning is a set of methods that allow machines to automatically discover representations needed for detection or classification from raw data. Deep learning methods are, therefore, representation-learning methods with multiple (deep) levels of representation. By using deep learning, very complex functions can be learned. One essential component leading to breakthrough results in image recognition is a special neural network called a convolutional neural network (CNN). CNNs eliminate the need for manual feature extraction, as shown in Figure 2.2 above. The features are learned directly by the CNN (LeCun, Bengio, & Hinton, 2015; Olah, 2014). Therefore, this research study focused on deep learning methods to achieve image recognition instead of traditional machine learning methods.

2.3.2 Artificial Neural Networks (ANNs)

Deep learning uses a layered structure of algorithms, modelled on the biological neural network of the human brain, called artificial neural networks (ANNs). The neuron is the basic computational unit of an ANN. Neurons are combined into groups called layers, which are connected to each other. In the classical artificial neural network, the neurons in one layer are all connected to the neurons in the next layer as show in Figure 2.3 below. Each neuron and its connection to the next layer neurons have associated learnable weights. These weights change during the learning process. These weights form input to an activation function that gives an

output if a certain threshold is met. A process called forward-pass is used to obtain predictions, whereas a process called backpropagation is used to train the model (Filip, 2018; LeCun, Bengio, & Hinton, 2015).

However, the classical ANN is not suitable for image processing because these networks easily experience over-fitting i.e. the network fails to generalize well from the training data to correctly classify unseen data. Moreover, the fact that each neuron in one layer is connected to every other neuron in the next layer means that ANNs have many trainable parameters which in turn require more powerful processors to process (IIango & Kumar, 2017). This is not ideal in a resource-constrained device like a mobile phone.

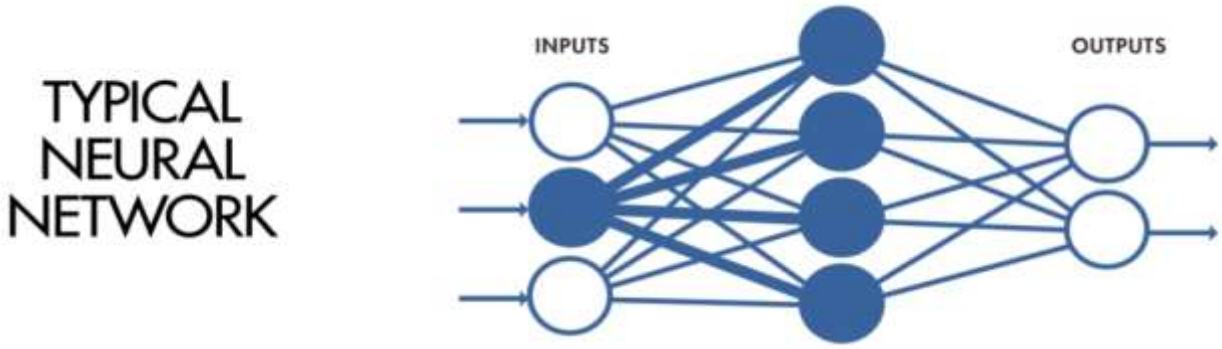


Figure 2.3: Artificial Neural Network Structure (Image Recognition: MathWorks, 2018)

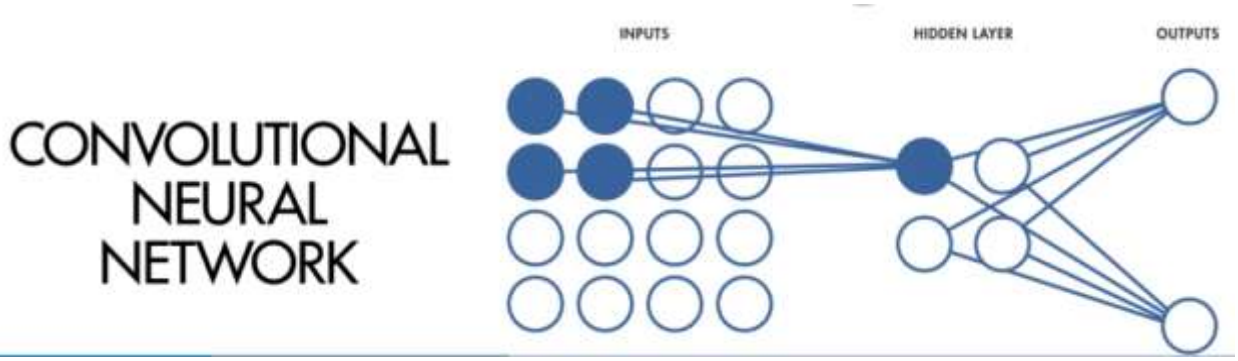


Figure 2.4: Convolutional Neural Network Structure (Image Recognition: MathWorks, 2018)

2.3.3 Convolutional Neural Networks (CNNs)

CNNs are multi-layered neural networks particularly designed for use with two-dimensional data like images and videos. The main difference between the classical ANN and a CNN is that only the last layer of a CNN is fully connected whereas in the classical ANN, each neuron is connected to every other neuron as illustrated in Figure 2.4 and Figure 2.3 above (Arel, Rose, & Karnowski, 2010; IIango & Kumar, 2017). A CNN uses many identical copies of the same neuron. This allows the network to have many neurons and express computationally large models while keeping the number of actual parameters that need to be learned reasonably small. In programming, a function is written once and then used in many places. Similarly, a CNN can learn a neuron once and then use it in many places (Olah, 2014). CNNs have proven to be efficient in image classification. However, the major drawbacks are CNNs require a large amount of images as the training dataset and take considerable time for training for the CNN to achieve meaningful accuracy. To overcome this, transfer learning is used to use a pre-trained model to perform classification on image datasets that may be outside the domain of the pre-trained model (Devikar, 2016).

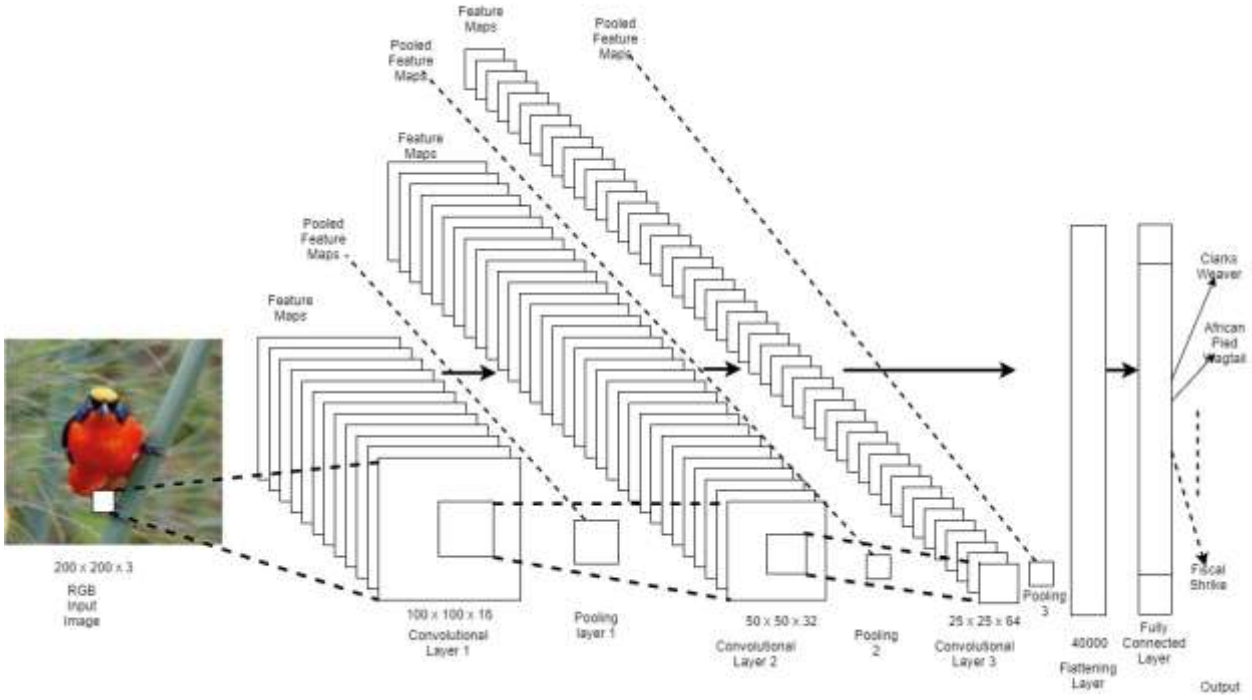


Figure 2.5: Training process of a CNN adapted from (Devikar, 2016)

Figure 2.5 above shows a sample training process of a CNN and is explained below. A CNN consists of different layers, which include convolutional, an activation function usually rectified linear unit (ReLU), pooling and the fully connected layers.

Convolution layer – This layer extracts features from the input image. Filters are used to extract these features. For example, assume a 32 x 32 image. The computer sees a 32 x 32 x 3 array of numbers, where the 3 represent the red, green and blue values for each picture element (pixel). Assume a 5 x 5 x 3 filter. The filter takes the first position at the top left corner of the image. It then slides, or convolves, across the input image multiplying the values in the filter with the original pixel values of the image. These multiplications are summed up to create one figure that represents where the filter is. The filter moves one unit or stride to the left and the process repeats. After sliding over the whole image, a 28 x 28 x 1 array of numbers called the feature map is created. If 6 5 x 5 filters are used then a 28 x 28 x 6 array will be created (Nielsen, 2015).

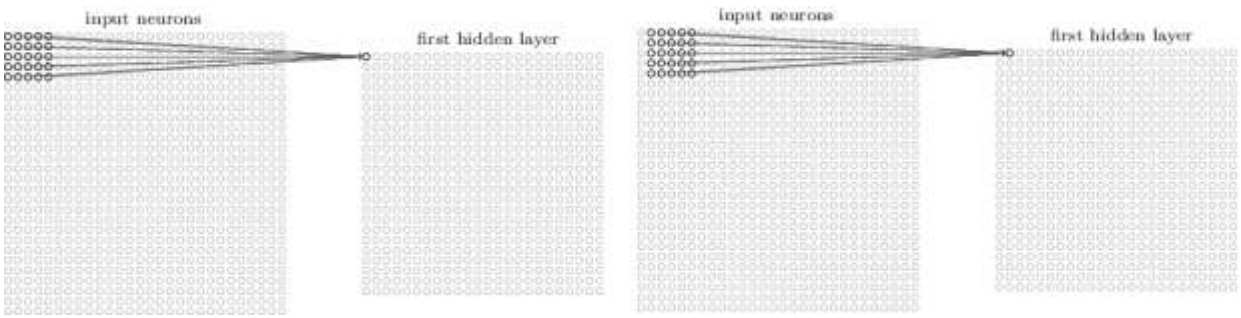


Figure 2.6: Convolutional Layer (Nielsen, 2015)

After each convolution the output reduces in size and it becomes the input of the next layer.

Activation function – Rectified Linear Unit (ReLU) – ReLU is an operation that replaces all negative pixel values in the feature map by zero. This is to introduce non-linearity since most of the real world data is non-linear (Devikar, 2016).

Pooling Layer – Pooling layers usually come after the convolution layer and its role is to simplify the information in the output from the convolutional layer. It reduces the dimensionality of each feature map but retains the most important information. A common pooling procedure is max pooling, where a pooling unit outputs the maximum value (Nielsen, 2015).

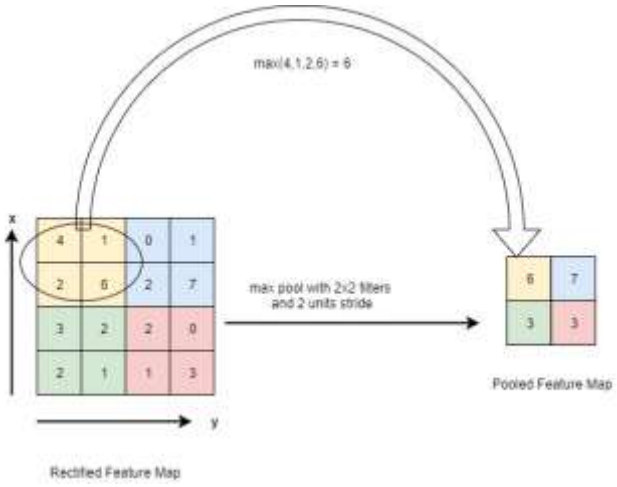


Figure 2.7: Max Pooling process adapted from (Devikar, 2016)

Fully connected Layer – After several convolutional and pooling layers a fully connected layer takes all the neurons in the previous layer and connects it to every neuron in the layer to produce fully connected output. This is the layer that does the actual classification work.

2.3.4 Depth-wise Separable Convolutional Neural Networks

As we have seen in section 2.3.2 above, convolutional neural networks are well suited for computer vision tasks. Most work in CNNs generally has been to increase accuracy by making deeper and more complicated networks. However, increased accuracy comes at a cost of speed and network size. This becomes a problem when deploying such CNNs in resource constrained mobile devices. For example, many real-world problems like self-driving cars require recognition tasks to be done quickly on a computationally limited platform. To address this, research has been done on different approaches in creating high accuracy but small and low latency neural networks. One such network is the MobileNet, which uses depth-wise separable convolutions to drastically reduce computation required and network size (Howard, et al., 2017).

Figure 2.8 below shows the differences between standard convolution and depth-wise convolution. Assume an input image of **Df x Df x M** where Df x Df is the image size and M is the number of channels. If there are **N** filters of size **Dk x Dk x M**, then the output of a standard convolution will be **Dp x Dp x N**. Therefore, standard convolutions have the computational cost of **Dk x Dk x M x Dp x Dp x N**.

Depth-wise separable convolution consists of two layers: depth-wise convolution and pointwise convolution. In the depth-wise operation, convolution is applied to a single channel

instead of all the M channels. So the filter will be of size **Dk x Dk x 1** as shown below in Figure 2.8. Since there are M input channels the output will be of size **Dp x Dp x M**. Therefore, depth-wise convolutions have the computational cost of: **Dk x Dk x Dp x Dp x M**. In the pointwise operation, a 1 x 1 convolution is applied to all M channels in order to combine and generate new features. So the filter will be of size **1 x 1 x M** and since we originally had N filters the output will be of **Dp x Dp x N**. The computational cost of the pointwise operation will therefore be **M x Dp x Dp x N**. This added to the depth-wise computational cost to give a total cost of **(Dk x Dk x Dp x Dp x M) + (M x Dp x Dp x N)**.

Ratio (R) = depth-wise separable convolution computation cost / standard convolution's

We get ratio $R = (Dk^2 \times Dp^2 \times M) + (M \times Dp^2 \times N) / Dk^2 \times M \times Dp^2 \times N = \mathbf{1/N + 1/Dk^2}$

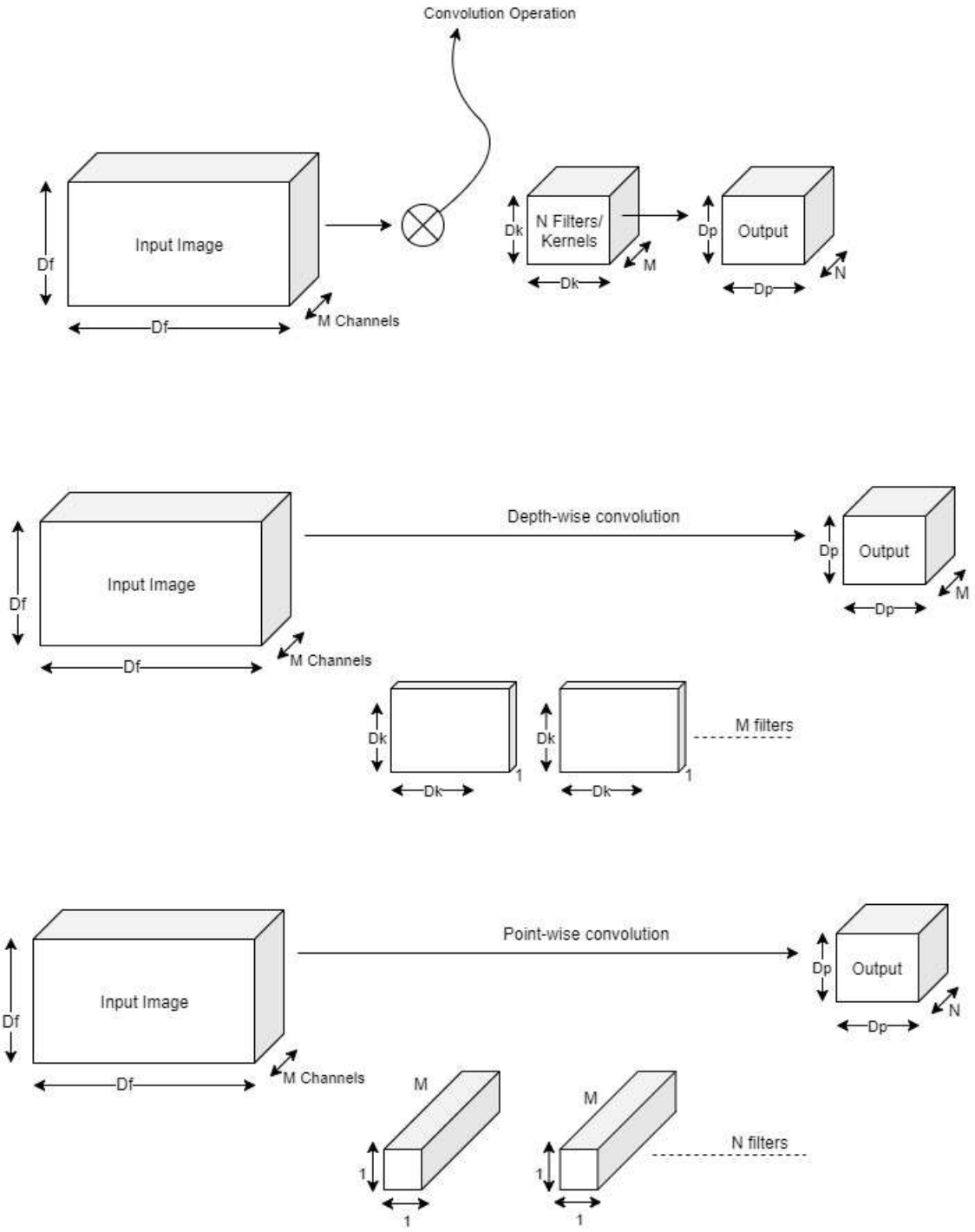


Figure 2.8: Standard versus Depth-Wise Convolution adapted from (Howard, et al., 2017)

MobileNet uses a 3 x 3 depth-wise separable convolutions. This uses eight to nine times less computational than a standard convolution with only a small reduction in accuracy. This means we can deploy small and fast CNNs without losing much in terms of accuracy (Howard, et al., 2017). Version two of the MobileNet improves accuracy and performance by using inverted residuals and linear bottlenecks. MobileNetV2 are faster for the same accuracy and need about 30% fewer parameters than the first version. Therefore, MobileNetV2 is an effective feature extractor for object detection, segmentation and fine-grained categorization tasks (Sandler & Howard, MobileNetV2: The Next Generation of On-Device Computer Vision Networks, 2018; Sandler, Howard, Zhu, Zhmoginov, & Chen, 2019).

2.3.5 TensorFlow Framework

TensorFlow is a high performance numerical computation software library originally developed by researchers and engineers at Google. It is now open source and has strong support for machine learning and deep learning. At Google, for example, the framework powers many of their products like Gmail and Google translate. This study used the convolutional neural network libraries and tensorboard, the visualization toolkit, of TensorFlow (TensorFlow, 2018).

2.4 Conceptual Framework

The following conceptual framework links the reviewed literature with the research problem and objectives. The first part of the proposed image recognition system is training a depth-wise separable convolutional neural network with images of bird species using transfer learning. The next stage is testing the trained model with a training dataset and finally the last stage is embedding the trained model in the mobile application that will query the resulting model and get a result from the model.

A two-fold conceptual model is proposed for the image recognition system's mobile application. The developed image recognition system should be able to determine whether or not an image contains a bird. If the image contains a bird, then the system should perform recognition of the bird in the image based on multiple learnt features of birds from the training dataset and determines its species. Finally, the mobile application user will have a chance to add to the image dataset via a feedback link so as to improve the consequent results of the image recognition system. The conceptual framework is shown in Figure 2.9 below.

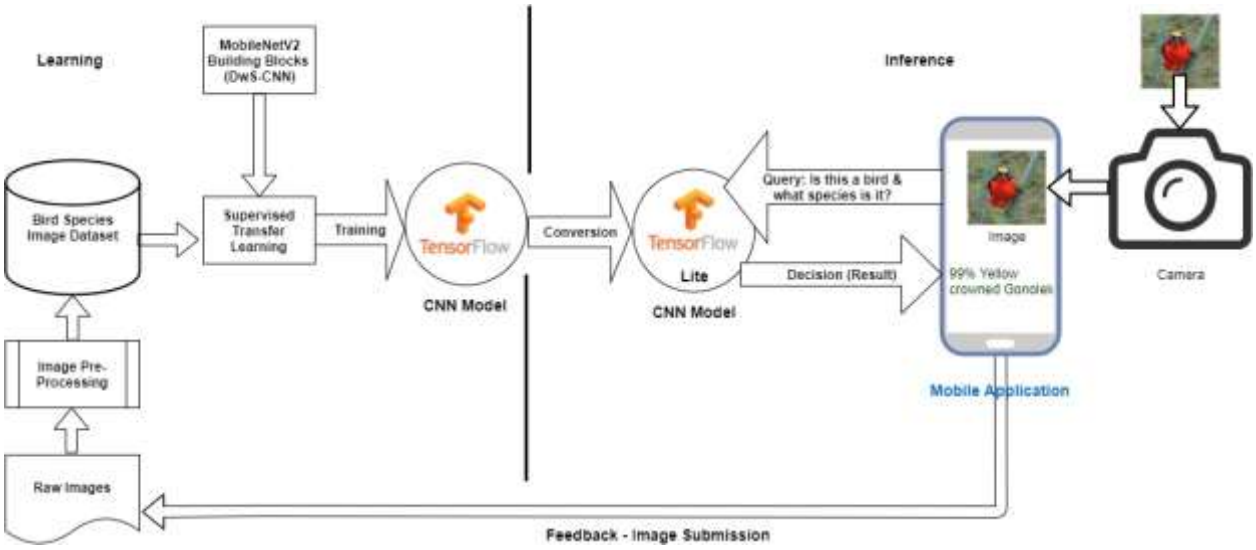


Figure 2.9: Conceptual Framework of the Image Recognition System

Chapter 3
Research Methodology

3.1 Introduction

The Research Methodology chapter explored the research design, data collection and the development of the convolutional neural network model. Additionally, the chapter covered the methodology that would be used in system development and the proposed CNN system architecture. Finally, the chapter made a statement on the research quality and the ethical considerations for the research.

3.2 Research Design

This research was applied research as it aimed at solving a real-world problem facing the birding community of bird identification and its utility was limited to bird identification (Basic Research Designs: CIRT, 2018). The research built a mobile application based on the popular and open Android mobile operating system that took an image as input, processed the image, and then inferred the bird species in the image.

3.3 CNN Model Development

The convolutional neural network (CNN) would be developed using the following steps:

i. Data Collection – image dataset
ii. Data pre-processing – augment, standardize and split the images to a training and testing dataset
iii. Training the CNN model – using the training dataset
iv. Testing the CNN model – using the testing dataset

3.3.1 Data Collection

Thousands of images are required to train the CNN effectively thus the size of the training dataset will be crucial because one of the characteristics of deep learning is its ability to perform well when trained with large data sets (Brynjolfsson & Mcafee, What's Driving the Machine Learning Explosion?, 2017). The bird images were obtained from public sources like Google's Open Image, ImageNet, the Internet Bird Collection, social media websites like Flickr and the researcher's own images. The image files collected were in JPG or PNG image format.

3.3.2 Data Pre-processing

3.3.2.1 Augmentation

Image augmentation artificially expands an image dataset. Image augmentation operations will include flipping, rotation, shear, zoom, cropping, deforming, adjust hue, adjust saturation or adjust brightness and contrast (Custom Image Augmentation: Towards Data Science, 2017). The augmentation process expanded the image dataset by three or four times and helped the model cope with all distortions that occur in real-life images.

3.3.2.2 Standardization

The images were transformed to the same size i.e. 224px by 224px and converted to the same image format i.e. JPG.

3.3.2.3 Folder Structure

Approximately 80% of the images formed the training dataset and the remaining 20% formed the validation dataset. The images collected were put into the respective dataset folder and subfolders corresponding to the species name as shown in Figure 3.1 below. The subfolder name defined the label that would be applied to each image.

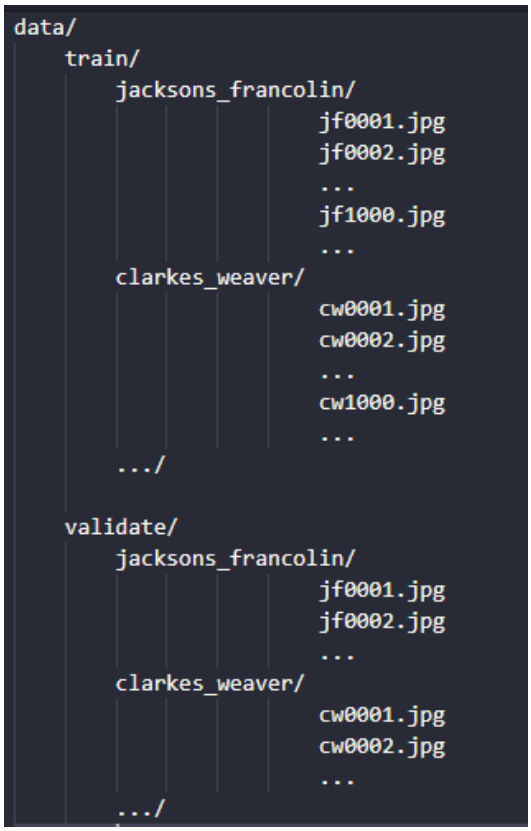


Figure 3.1: CNN Data Folder Structure

3.3.3 Training the Model

Transfer learning was used to train the network. Google's MobileNetV2 was used as the pre-trained model for feature extraction. MobileNetV2 offers better performance in terms of speed and model size and a higher accuracy than the previous MobileNetV1 model (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2019). TensorFlow was used to load the pre-trained MobileNetV2 model (Figure 3.2) and retrain the next-to-last layer with the training dataset. TensorFlow provides a Python script called retrain.py that can be used to retrain a model. Similarly, the validating dataset was then used to validate the new retrained model. Example python function call is as below:

```bash
python tensorflow/image_retraining/retrain.py \
    --image_dir ~/data/train --architecture mobilenet_v2/feature_vector
```

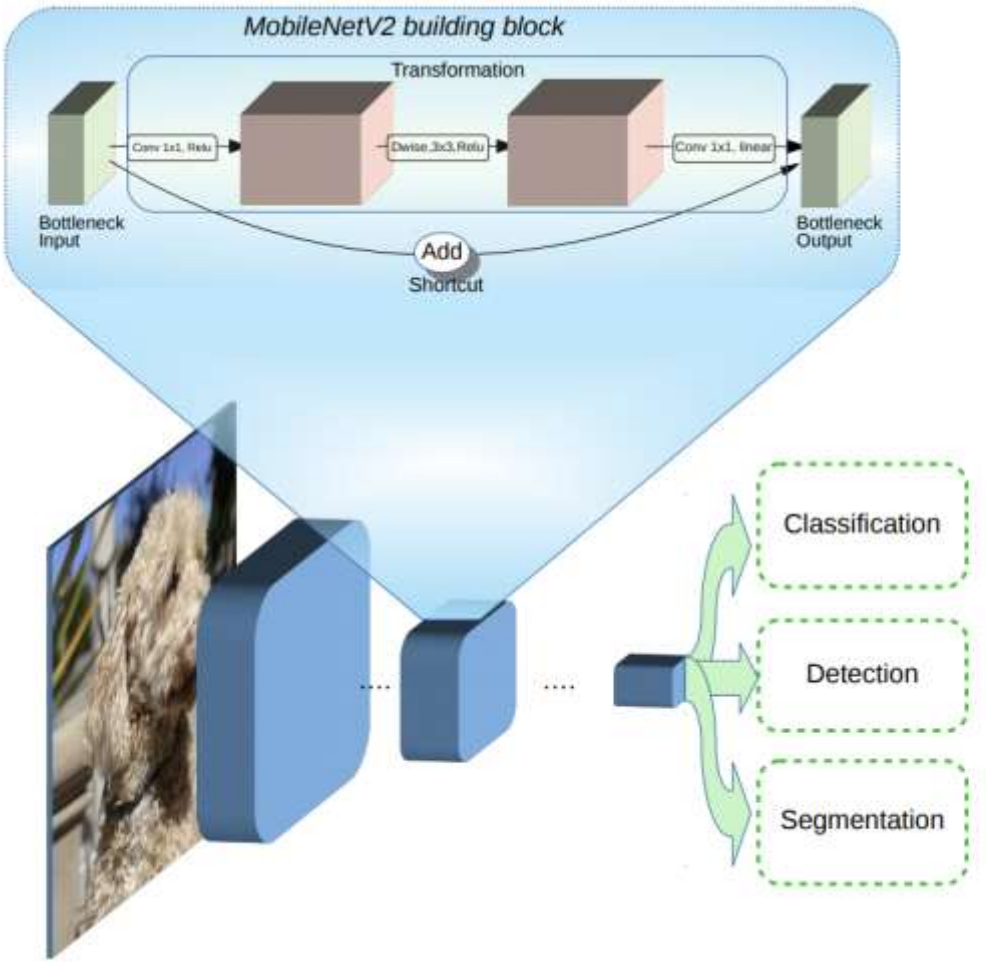


Figure 3.2: MobileNetV2 Architecture (Sandler & Howard, MobileNetV2: The Next Generation of On-Device Computer Vision Networks, 2018)

3.3.4 Testing the Model

50% of the testing image dataset i.e. 10% of the whole dataset was used as the sample size to test the resulting model. Tensorflow provided a script called label_image.py that was used to test the model. An example usage of the script is as below:

```
python tensorflow/image_retraining/label_image.py --
graph=c:\kenbim\retrained_graph.pb --labels=c:\kenbim\retrained_labels.txt --
input_layer=Placeholder --output_layer=final_result --input_height=224 --
input_width=224 --image=C:\images\test_image1.jpg
```

3.4 System Development Methodology

The system development methodology for this research was a prototyping-based methodology and evolutionary prototyping in particular. Analysis, design and the implementation phases were done concurrently, and all the three phases were repeated until the system was completed as shown in figure 3.3 below. Using prototyping methodology quickly produces a system that users can interact with and thus there was visible progress throughout the research project (Dennis, Wixom, & Tegarden, 2009).

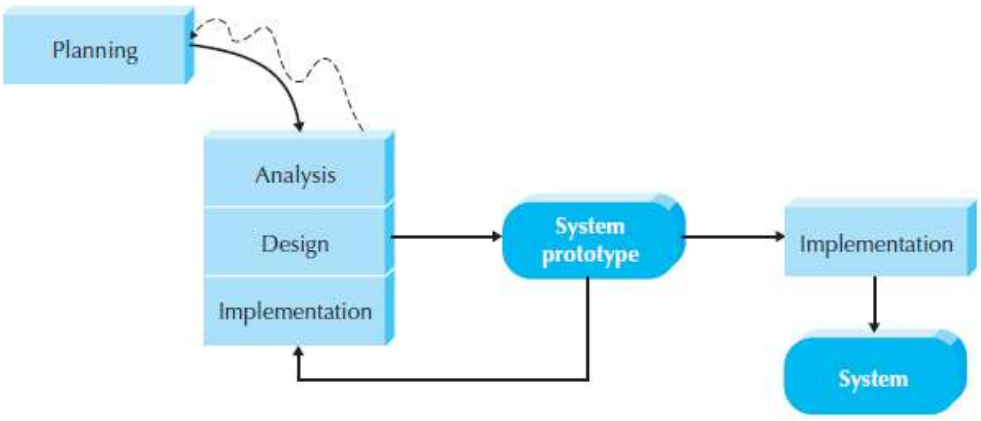


Figure 3.3: Prototyping-based methodologies (Dennis, Wixom, & Tegarden, 2009)

In evolutionary prototyping, a prototype is built which is then refined based on feedback from stakeholders. Features are added or removed progressively and there is real-time development via experimentation. The source code, style, design patterns and test plans are production-ready unlike in throw-away prototyping. Evolutionary prototyping is about repetitive iterations and is, therefore, closer to agile. However, unlike agile methodologies, there are no rules that bind iterations to regular sprints (Why we need Throw-away Prototyping, 2017).

Both the convolutional neural network and the mobile application were developed using evolutionary prototyping. Prototyping was essential in building the CNN because some hyper parameters, for example, number of training steps, were experimental. Figure 3.4 below shows how the mobile application prototype evolved.

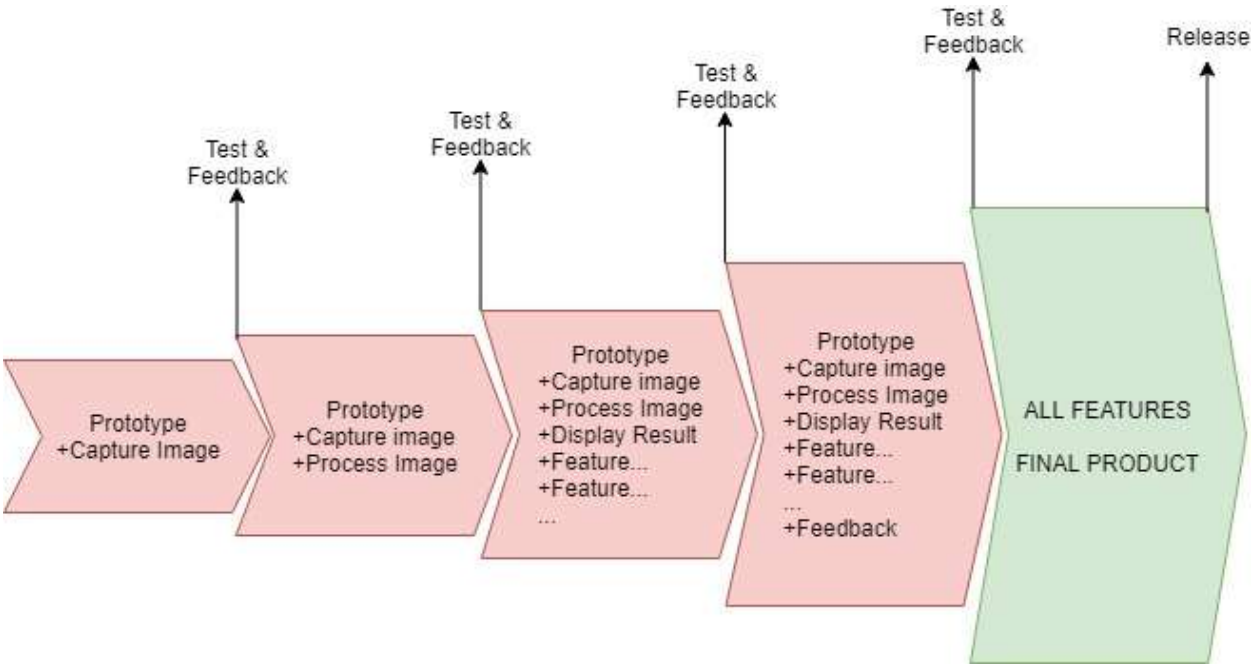


Figure 3.4: Evolution of the Mobile Application adapted from (Why we need Throw-away Prototyping, 2017)

3.4.1 System Analysis

This phase provided the answers to the questions of who would use the image recognition system, what would the system do and where and when it would be used. This phase mainly entailed requirements gathering. System analysis performed involved the steps below.

3.4.1.1 Requirements Specification

The requirements for the image recognition system was developed using content analysis methodology. After analysis, functional and non-functional requirements were documented.

3.4.1.2 Hardware Analysis

Training a deep neural network like a CNN requires a lot of computational power. A graphics processing unit (GPU) is recommended (LeCun, Bengio, & Hinton, 2015). However, by using transfer learning, the researcher was able to train the model using CPU only in reasonable time.

3.4.1.3 Software Analysis

This research used Tensorflow machine learning framework and React-Native JavaScript framework to create the CNN model and the mobile application respectively. Tensorflow framework was chosen because it is open-source, flexible and has all the tools needed in this research to train and validate the model (TensorFlow, 2018). React-Native framework was chosen because it is a hybrid mobile application framework. A single codebase was created that could be compiled for iOS or Android mobile operating system. Moreover, React-Native create native components unlike other hybrid frameworks that use WebViews. Thus, it has close-to-native performance (Singh, 2018).

3.4.2 System Design

The research's system design approach was rapid application development (RAD) using evolutionary prototyping strategy and object-oriented programming. The output from this phase was the system architecture diagram, the use case diagram, the system sequence diagrams, the mobile application wireframes and the database schema diagram.

3.4.3 System Implementation

The image recognition system was built in this phase. System implementation usually has three steps i.e. construction, installation and maintenance or support (Dennis, Wixom, & Tegarden, 2009).

i. System construction – Windows 10 64-bit was the development environment. The following are the steps taken to construct the image recognition system:
    a. Install Python and machine learning libraries like NumPy, Pandas, MatplotLib etcetera.
    b. Install TensorFlow and Keras libraries and related dependencies.
    c. Collect and save image dataset in the folder structure as shown in section 3.3.2.3.
    d. Download and re-train the pre-trained model using TensorFlow and libraries.
    e. Validate the new model.

f. Convert the model to TensorFlow Lite format to create a smaller "lite" model that is better suited for a mobile phone.
g. Construct the mobile application user interface.
h. Embed the "lite" model in the mobile application
i. Build the mobile application for the Android mobile operating system.
j. Test the whole system.

ii. System installation – The system was installed on a demo Android smartphone and simulated on an Android simulator.

iii. System support – There was formal and informal post installation review to identify any major or minor changes needed for the next prototype.

3.5 Research Quality

The research data must have both validity and reliability. Validity is the extent to which the data accurately measures what they were intended to measure, whereas reliability is the extent to which the data collection method will yield the same findings if replicated by others. This research ensured validity by using automated model testing and using a confusion matrix. Furthermore, tensorboard was used to visually track the convolution neural network model's loss or learning rate and this made it easier to debug and optimize the model. Proper documentation and following best practises in mobile application development was done to ensure the research's reliability.

3.6 Ethical Considerations

Ethics are standards or codes of conduct that help distinguish between what is acceptable and what is not (Ethical Considerations: CIRT, 2018). Images used in this study were obtained from public sources online with permission to use for educational research. The researcher also used his own images which he has the copyright. All previous works were cited appropriately and due acknowledgement given to the respective authors.

Chapter 4
System Design and Architecture

4.1 Introduction

This chapter outlines the design architecture of the mobile-based image recognition system that can identify bird species in Kenya based on the conceptual model presented in Figure 2.5. This chapter will explain the components of the developed system, the interactions between the different components of the developed system and the interactions between the users and the developed system. Unified modelling language (UML) modelled these interactions and illustrated using the use case diagram, system sequence diagram, database schema diagram and the mobile application mock-up wireframes.

4.2 Requirements Analysis

This section describes the system requirements based on the research objectives. The requirements that were gathered, through content analysis methodology done and following the objectives of this research study, can be categorized into two sections; functional and non-functional requirements.

4.2.1 Functional Requirements

i. The system should allow the user to choose an image from their mobile device's Photo Library and enable the user take a photo if their mobile device has a camera. The chosen image is then used as the input to the image recognition system.
ii. The system should have the ability to infer the bird species in the input image and determine or get location information of the input image.
iii. The system should allow first time users to register for an account and login.
iv. The system should have a database to save user information and save the user's classification and location results to create a bird distribution map.
v. In addition to the above, the system should show the user their saved classification results.

4.2.2 Non-functional Requirements

i. Can run on a mobile device – The system should be able to run on the Android mobile operating system.
ii. User friendly – The mobile application should have a user-friendly user interface, which conforms to common user interface design patterns. The mobile application user interface should be responsive; that is adapting to the various mobile phone screen sizes
iii. Performance – The system should be able to infer and return a classification result in less than a second so as to increase the user experience.
iv. Data security – Data between the user's mobile device and the cloud database should be encrypted to protect the user's data.
v. Availability and stability – The system should be available whenever the user wants to use it and it should be stable and not crash when in use.
vi. Maintainability – The system should be easy to maintain and support.

4.3 System Architecture

The system architecture for the image recognition system is shown below in Figure 4.1. Raw images are pre-processed to create training and testing datasets. Subsequently, the convolutional neural network is trained and validated and converted to a lite format for embedding into the mobile application. The main input for the mobile application is the user selected image, which is fed into the embedded CNN model for inference. The result obtained is showed to the user. The user can choose to save the result into a cloud database called Firebase. Therefore, the User does not need internet to make any inference and can use the application while offline. Internet would be only required if the User needs to save the results.

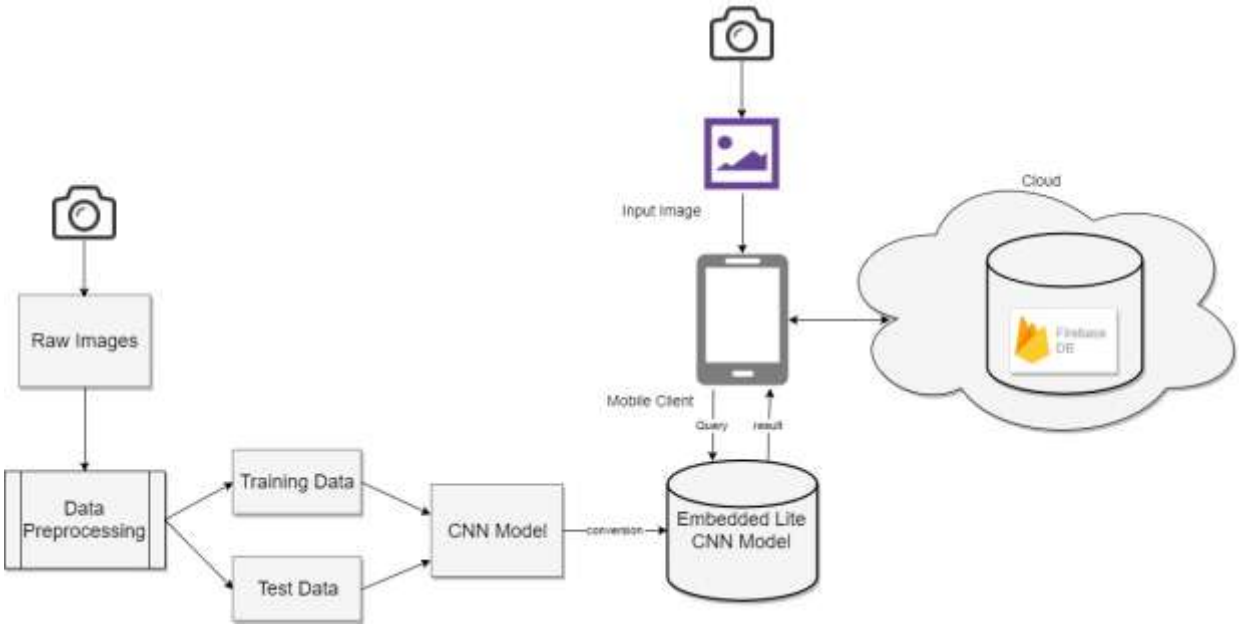


Figure 4.1: System Architecture

4.4 Use-Case Diagram and Description

A use case is a list of actions and event steps that define the interactions between an actor and a system to achieve a goal. The actors in this system are the general users and the system administrator. The administrator collates and pre-processes the image dataset and trains the deep neural network as explained in chapter three. The administrator then creates a lite model that is embedded into the mobile application. This embedded model makes the image classification. The bird observer, who is a sub-entity of the user, provides the system with an image to classify. The observer can choose to generate a bird map of their observation by saving to the bird map database. Figure 4.2 below shows the system use case diagram.

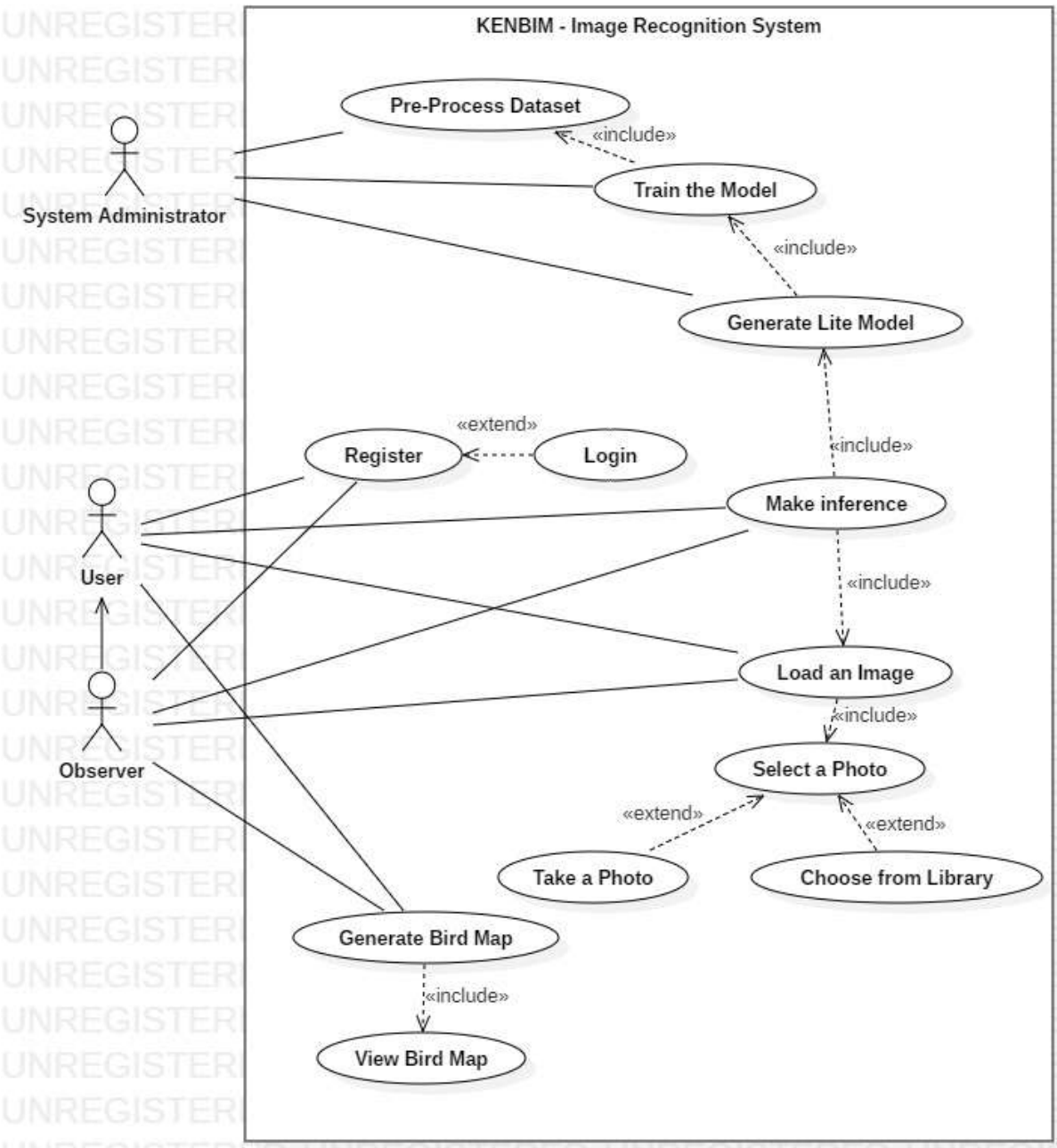


Figure 4.2: Use Case Diagram

Table 4.1 Data Pre-processing and Model Training Description

Use Case	**Data Pre-processing**
Primary Actors	The System Administrator
Brief Description	This use case describes how the System Administrator will pre-process the image dataset
Pre-condition	Sufficient bird images available, training system available
Post-condition	Pre-processed Image Dataset
Major Steps Performed	
Actor	**System**
Administrator sorts images into their respective folders	
Administrator sets augmentation parameters	
Administrator runs augmentation	
Administrator feeds pre-processed images into system	
	System performs feature extraction from the dataset
	System saves extracted features

Table 4.2 Get Image and Make Inference Description

Use Case	**Model Training**
Primary Actors	The System Administrator
Brief Description	This use case describes how the System Administrator will train the deep neural network
Pre-condition	Pre-process dataset, training system available
Post-condition	Trained model that can recognize and categorize bird images
Major Steps Performed	
Actor	**System**
Administrator selects pre-processed data	
Administrator selects training parameters, that is, model architecture, learning rate, number of epochs, training set percentage & testing set percentage	
Administrator selects output files	
Administrator runs training command	
	System splits dataset into training sets as per Administrators command.
	System downloads architecture selected e.g. MobileNet
	System inputs training dataset to retrain the model
	System uses test data to validate model during the training process
	System generates new model and saves it
Administrator runs conversion to lite command	
	System converts trained model into a lite model and saves it

Table 4.3 Get Image and Make Inference Description

Use Case	**Load an Image and Make Inference**
Primary Actors	The User
Brief Description	This use case describes how the User will load an image into the system for classification.
Pre-condition	Image to classify is available, Trained CNN model
Post-condition	Classified Image
Major Steps Performed	
Actor	**System**
The User selects image from the photo library or takes a photo	
	System loads image as input to embedded model
	System performs classification of image
	System returns classification result
User views returned species name and accuracy level	

Table 4.3 Generate and View Bird Map Description

Use Case	**Generate and View Bird Map**
Primary Actors	The User
Brief Description	This use case describes how the User will generate and view their bird map
Pre-condition	Classified image
Post-condition	Saved Bird Map
Major Steps Performed	
Actor	**System**
User verifies classification result	
User verifies or enters extra information about the observation e.g. location	
User commands system to save	System saves the classification results alongside any extra information to the database
User retrieves saved result	
	System retrieves classification data from database
	System displays information in a map
User views the bird map	

Table 4.4 Register and Login Description

Use Case	**Register and Login**
Primary Actors	The User
Brief Description	This use case describes how the User will register and login
Pre-condition	User wants to save classification result, Registration details
Post-condition	Registered user, Authenticated user
Major Steps Performed	
Actor	**System**
User enters registration details	
	System verifies and saves registration details
User enters login details	
	System verifies and authenticates the user

4.5 System Sequence Diagram

The system sequence diagram shows the interactions between the main entities in the system. Figure 4.3 and Figure 4.4 below shows the flow of activities in sequence. The User selects or takes a picture that they want classified. The User then chooses the image classifier function to classify the uploaded image. The image classifier queries the embedded neural network to obtain a result. The result is displayed to the User. The User can then save the result into the bird-map database and retrieve them later on.

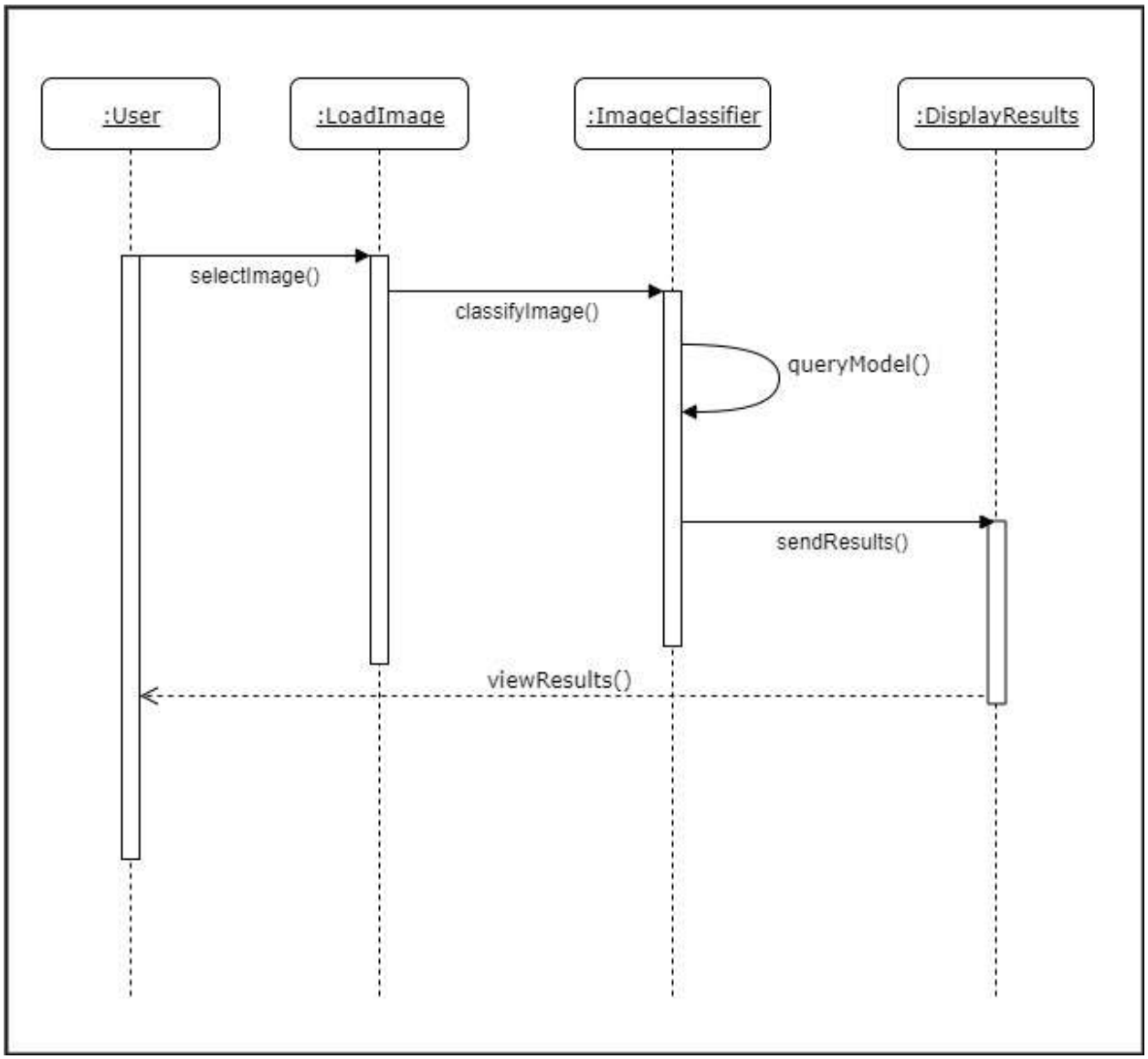


Figure 4.3: Image Classification Sequence Diagram

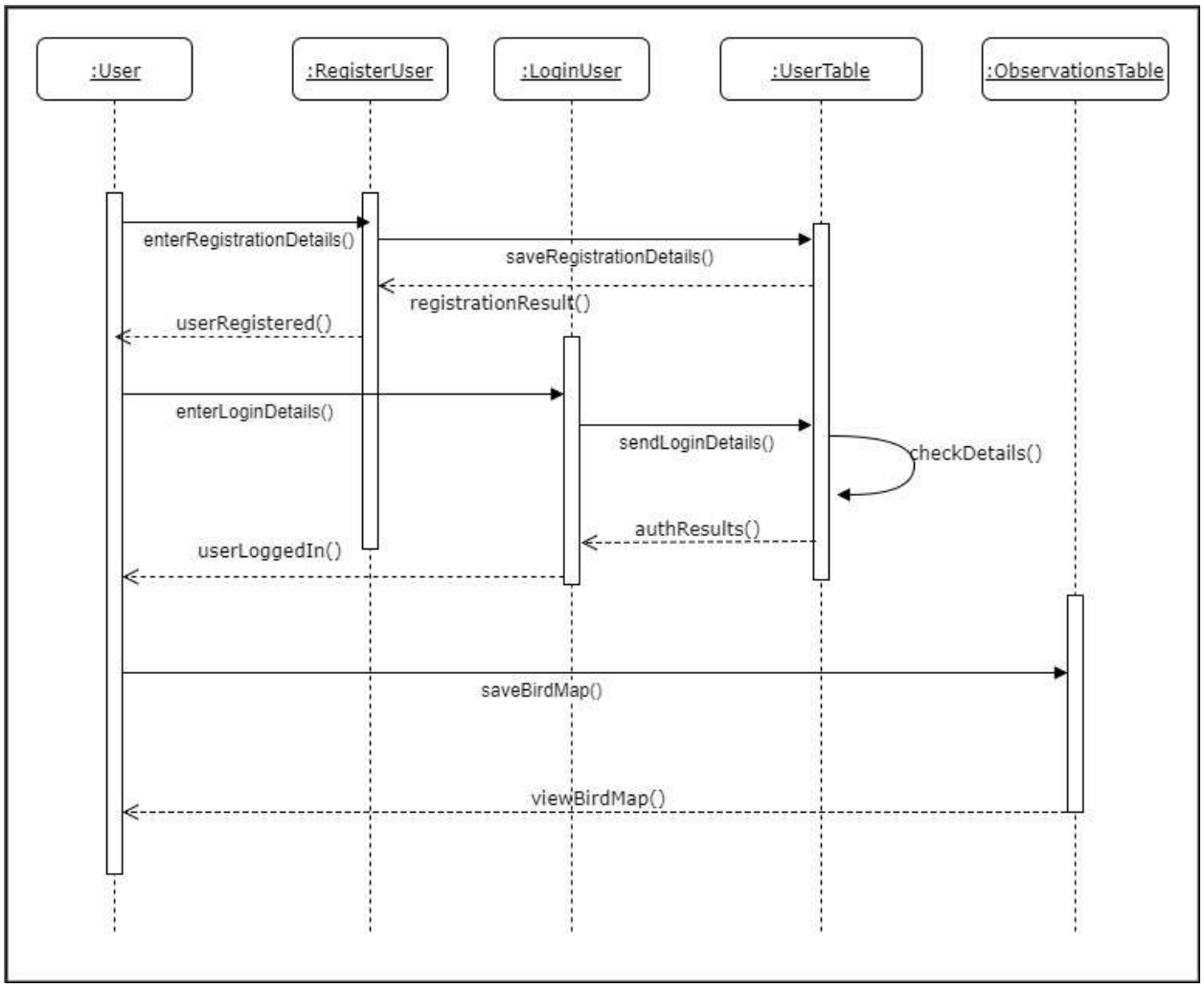


Figure 4.4: Bird Map Sequence Diagram

4.6 Database Schema

The system used the cloud Backend-as-a-Service (BaaS) database provided by Google called Firebase, which is a platform that allows for rapid application development. The database contained four tables as described in the database schema shown in Figure 4.5 below. The User table is automatically generated by the Firebase service and contains user data. The Observer table extends the User table and adds additional user information. The Bird table contains information about birds found in Kenya. The Observations table contains information about bird observations that the observer has made.

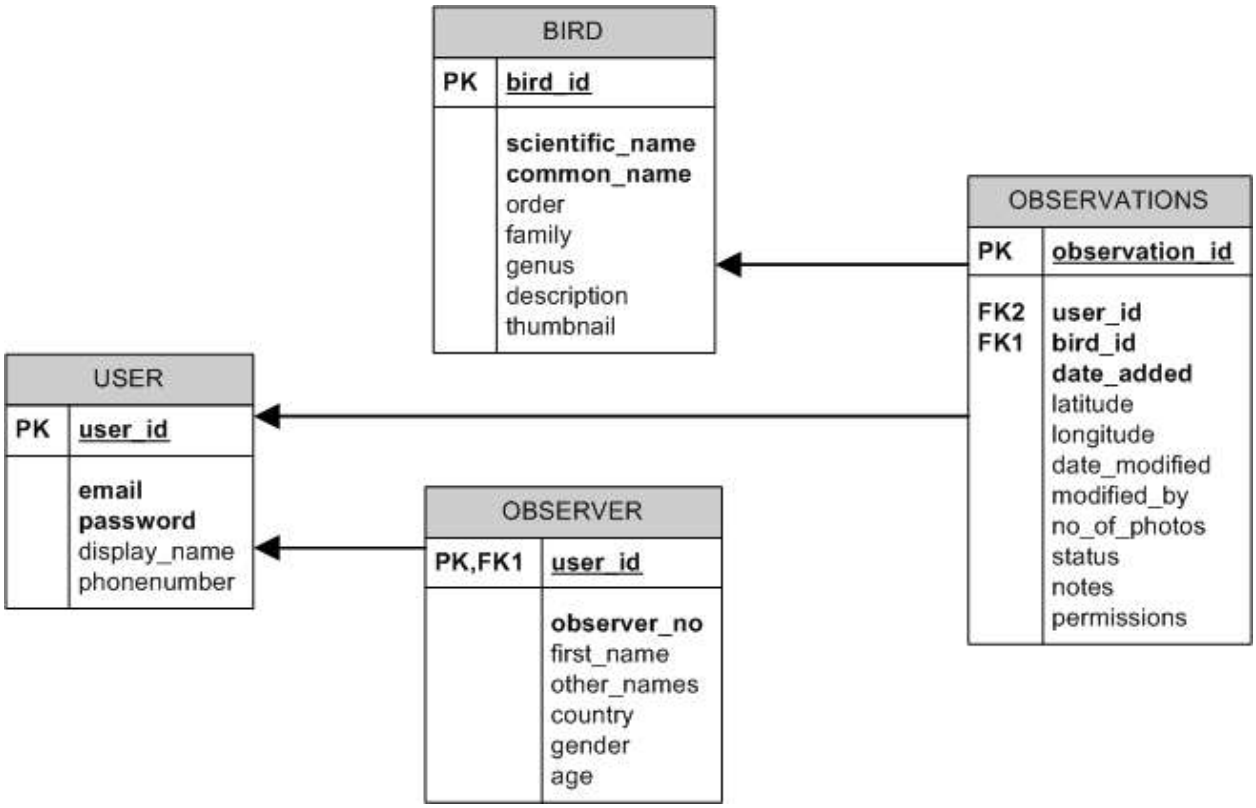


Figure 4.5: Database Schema

4.7 Wireframes of the Mobile Application

Figure 4.6 and 4.7 below shows the mobile application's wireframes. The user interface is designed to be simple and easy to use. The user does not have to be logged in to make inference or get more bird details, but has to register and log in to save and retrieve their bird map.

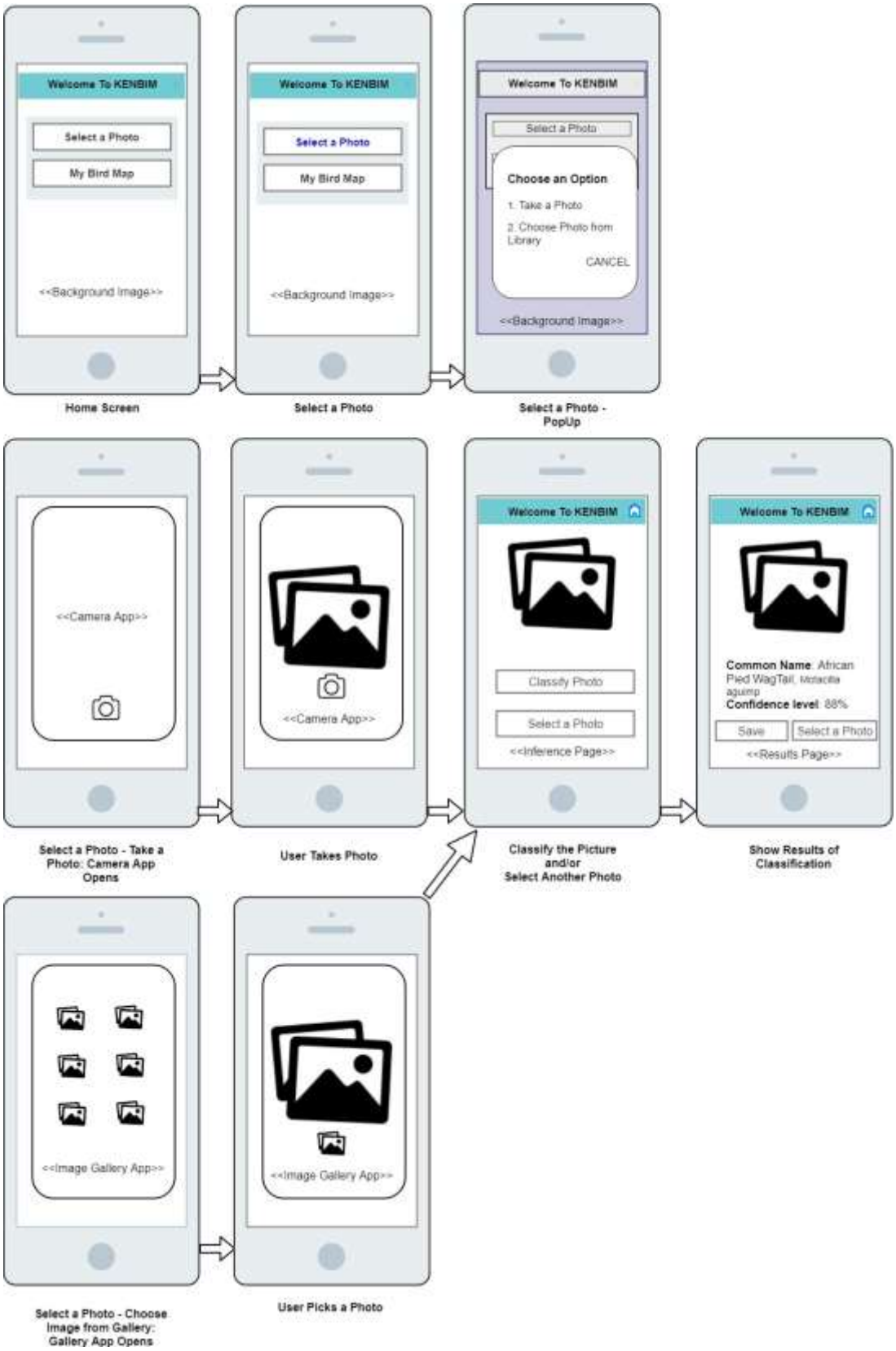


Figure 4.6: Wireframes showing the inference user journey

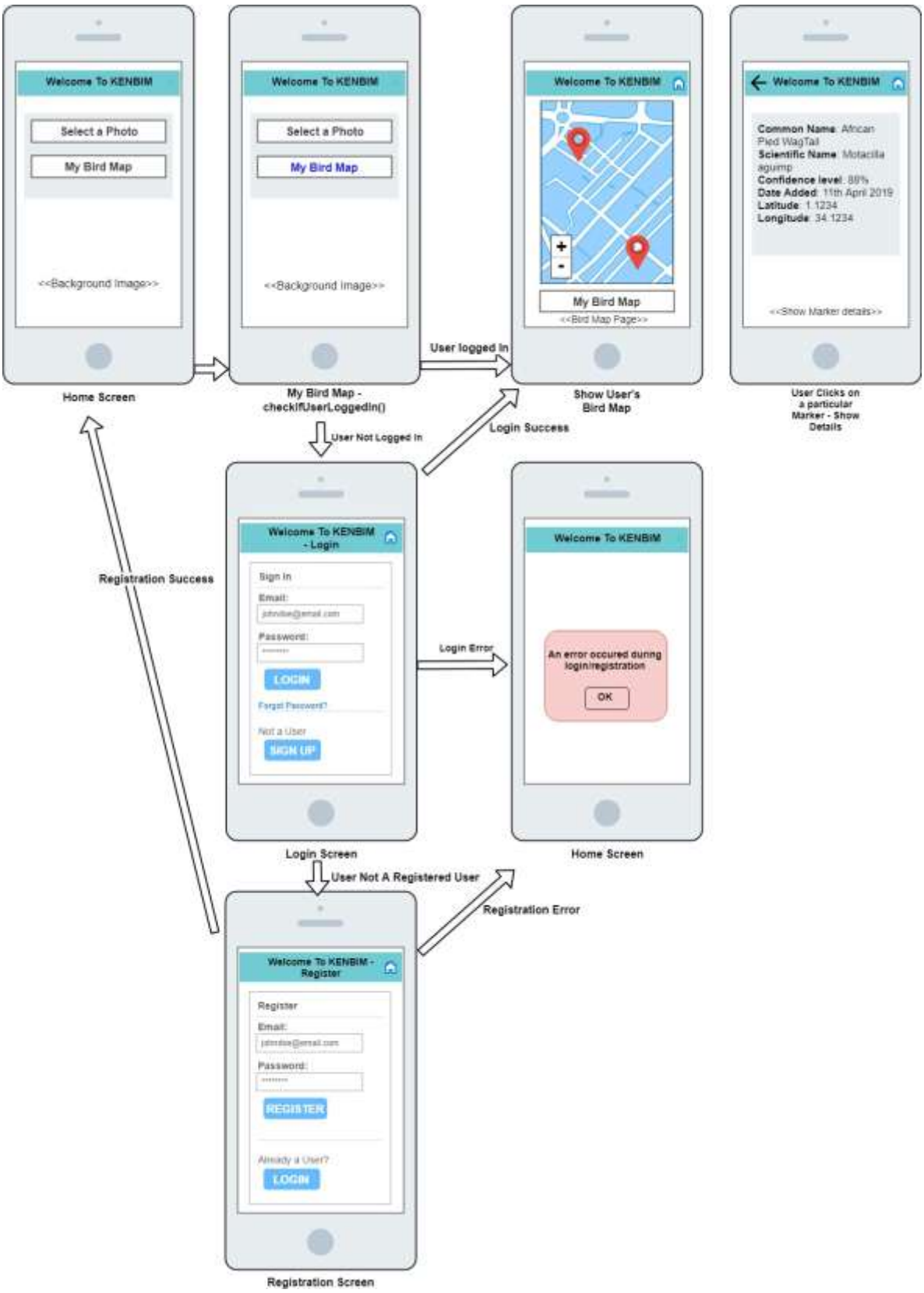


Figure 4.7: Wireframes showing the bird map user journey

Chapter 5
System Implementation and Testing

5.1 Introduction

The research resulted into an Android application that can classify a bird in an image. Thus, this chapter discusses the implementation and testing of the different modules developed to create a mobile-based image recognition system that can identify Kenyan bird species. The image recognition system prototype was developed by retraining a deep neural network with Kenyan bird image dataset and embedding the resulting model into an Android mobile application. Testing included functional and usability tests to check if the developed system accomplished the objectives set out at the beginning of the research project.

5.2 Convolutional Neural Network Components

The CNN was created using transfer learning. Features extracted from the activation of the MobileNet version 2 deep neural network, which has been trained on a large, fixed set of object recognition tasks, were re-purposed specifically for this project. As reviewed in Chapter two, MobileNet was chosen instead of bigger networks like Inception-ResNet-v3 because the MobileNet is specifically designed for on-device vision tasks like image classification. They create smaller, fast models yet do not lose much in accuracy (Sandler & Howard, MobileNetV2: The Next Generation of On-Device Computer Vision Networks, 2018). The CNN model created had several components as summarised in Figure 5.1 below. Note that the Figure 5.1 is obtained by running the tensorboard visualization tool that is "tensorboard --logdir C:\kenbim\training_summaries" then accessing http://localhost:6006 in a web browser.

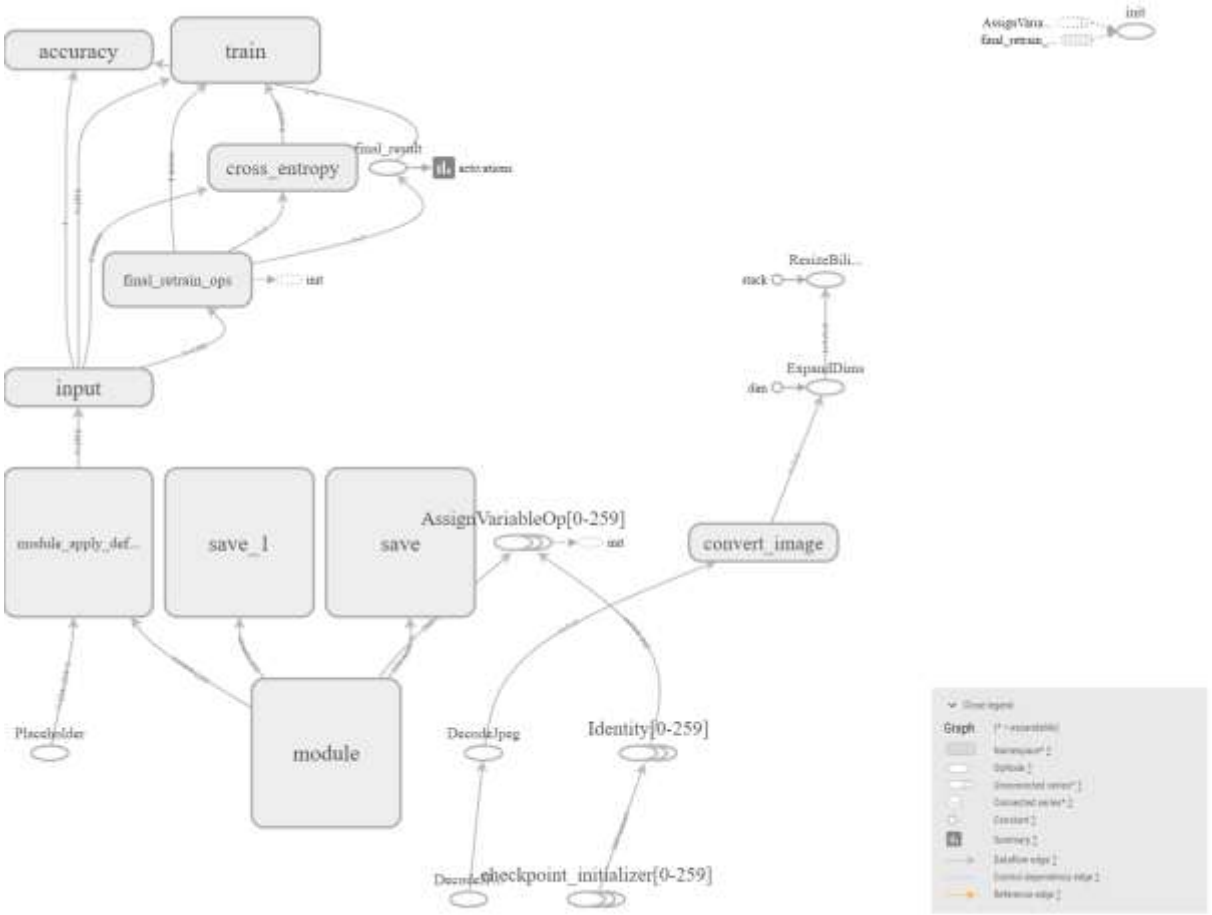


Figure 5.1: The CNN Model Structure

5.2.1 Input Layer

The first layer in the CNN is the input layer. The input nodes, Placeholder and Module, form the input layer and provide input images to the network. No computation is done to the input layers; they just pass information to the hidden layers. The input layers are highlighted in Figure 5.2 below.

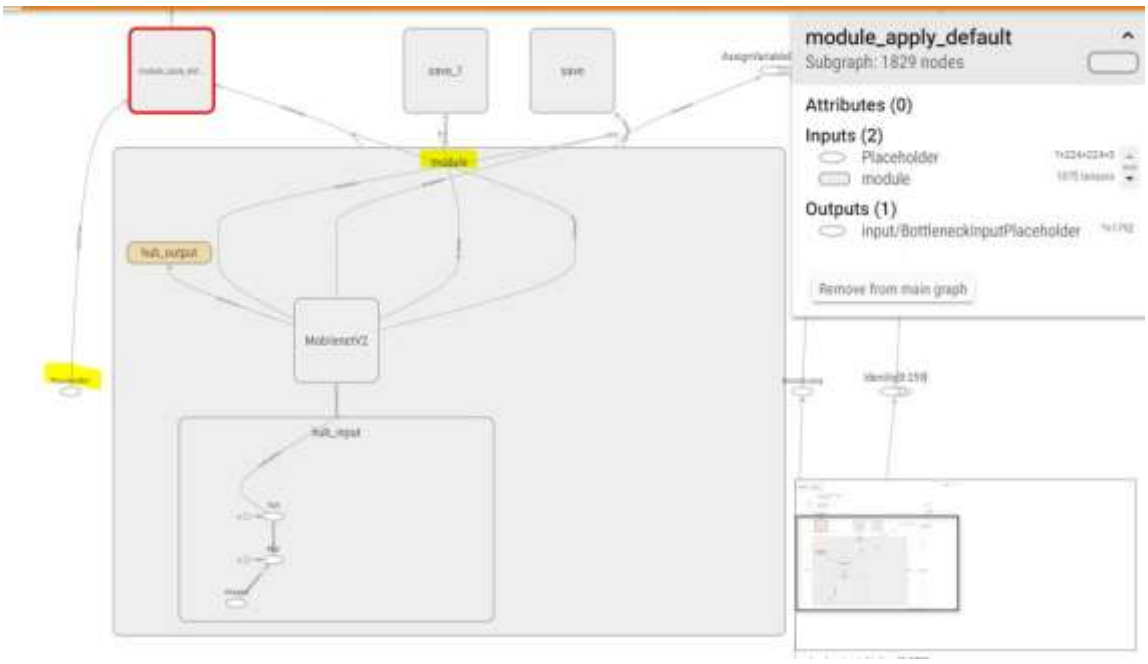


Figure 5.2: Expanded Input Layers

5.2.2 Hidden Layer

The hidden layer is a series of network nodes that are hidden from the outside world. They perform the necessary computations that perform feature extractions from the images. The hidden layer passes data from the input layer to the final output layer. The hidden layers in the CNN model perform convolution, pooling and activation. Figure 5.3 below shows an example of the hidden layer.

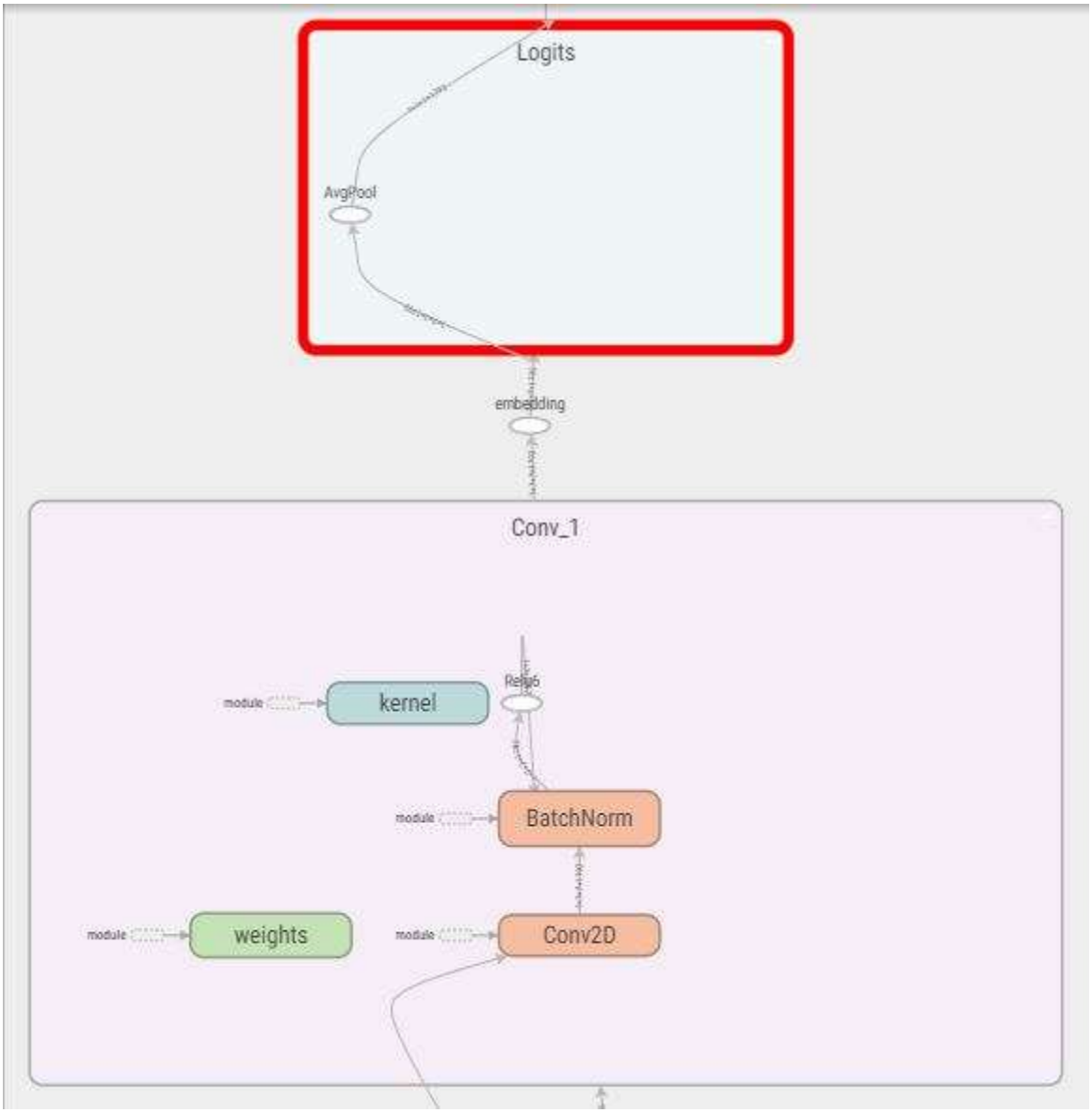


Figure 5.3: Example of Hidden Layer, Showing Convolution, ReLU and Average Pooling

5.2.3 Output (Classification) Layer

In the CNN model, the output layer performs the actual classification, giving the final output of the neural network. The output is a probability value between zero and one. The output layer is the layer that transfer training replaces, so that the retrained network can make predictions on the new labels. Softmax was used as the activation function for classifying the image data. The loss function used was cross entropy loss, which measured how close the predictions were to the actual labels.

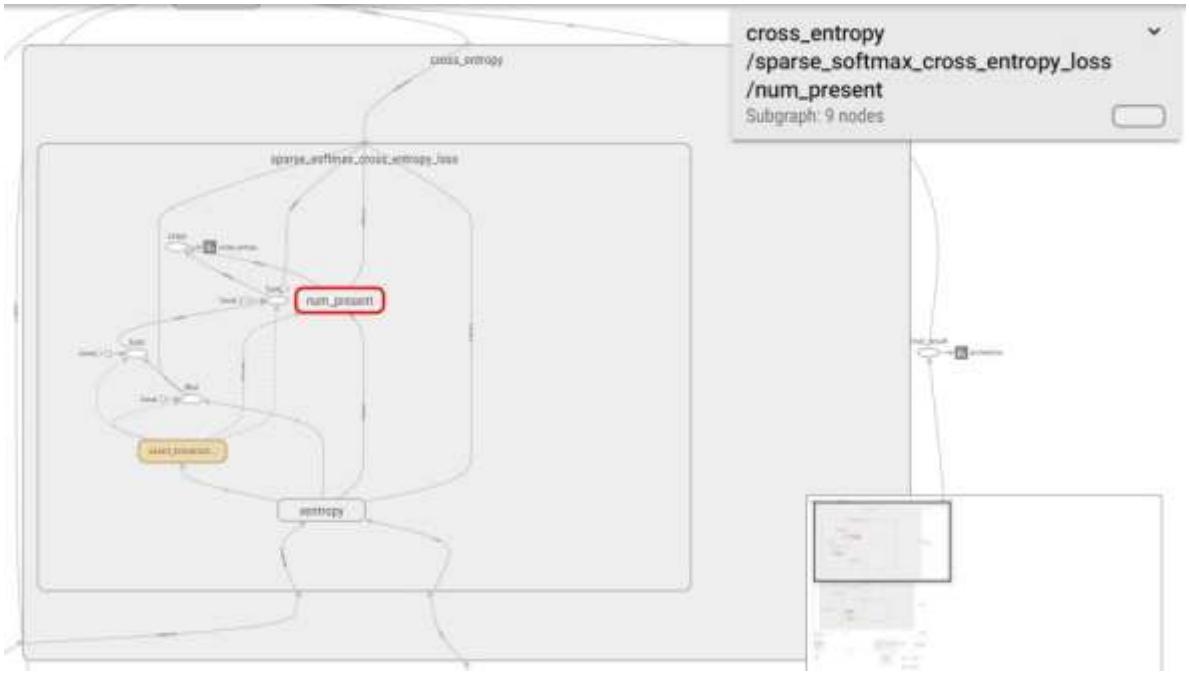


Figure 5.4: Cross Entropy Layer

5.3 Mobile Application Components

The frontend mobile application was developed using React Native, which is a JavaScript framework for creating natively rendered mobile applications. React Native is open and modular in nature. Various independent React Native components available can be re-used and this makes React Native a powerful platform to use when developing mobile applications. These components are discussed further below. Figure 5.5 below shows a screenshot of the landing or home screen that the user sees when they first open the application.

Figure 5.5: The Home Screen

5.3.1 Choosing an Image

React-native-image-picker, a React Native module that enables you to use mobile specific native user interface to choose media from the mobile device library or from the device's camera, was used to enable the user choose an image. The module returns the chosen image path or image URL, which is passed to the image classifier for classification. Figure 5.6 below shows a screenshot of the image picker in action.

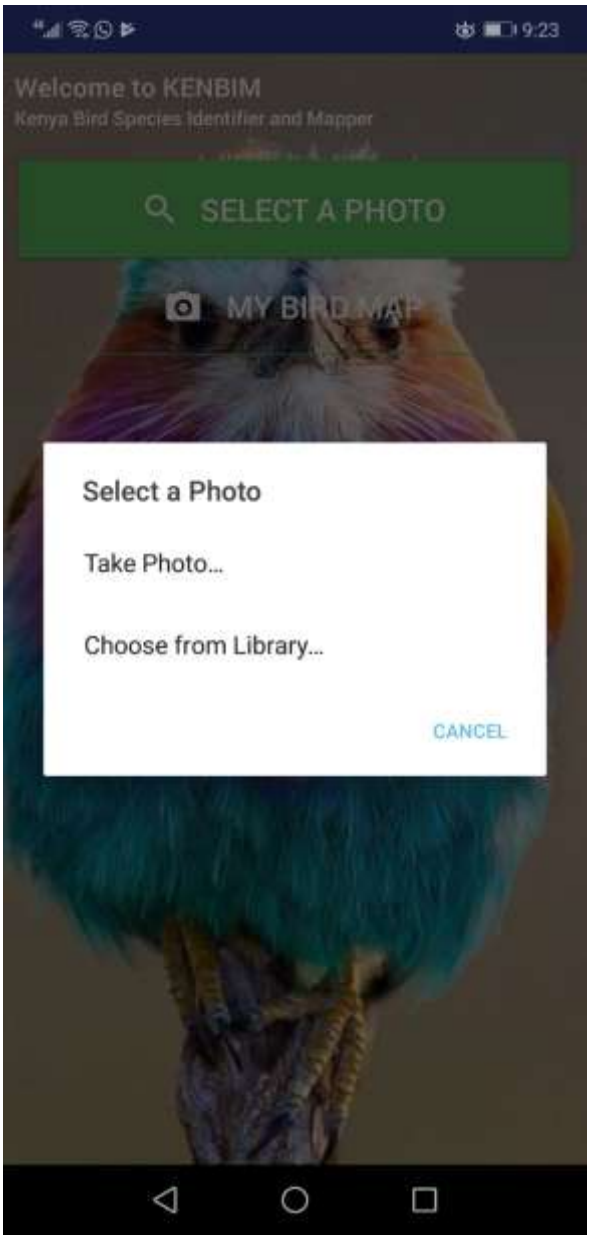


Figure 5.6: Choosing an Image

5.3.2 Image Classification

The CNN model developed was embedded into the mobile application using a React Native module called react-native-tensorflow-lite. The module was initiated with the lite model and its corresponding label file. The image path of the image chosen by the user was then used as an input to the module. The output obtained is stored in a results object that contains the bird species name, the confidence value and the inference time.

5.3.3 Login and Registration

The user must have created an account and logged into the system for the user to save or retrieve their bird map. The user does not need to login to classify an image though. The user requires internet to register or login. During registration, the user provides a valid email address and a password. The email address must be unique. The email and password is sent to the cloud database for verification. If the email is valid and unique and the password is more than six characters then the system stores the user's registration data. The system creates a unique user ID for each registered user. The password is hashed before being saved to increase system security. Figure 5.7 below shows screenshots of the registration and login screens.

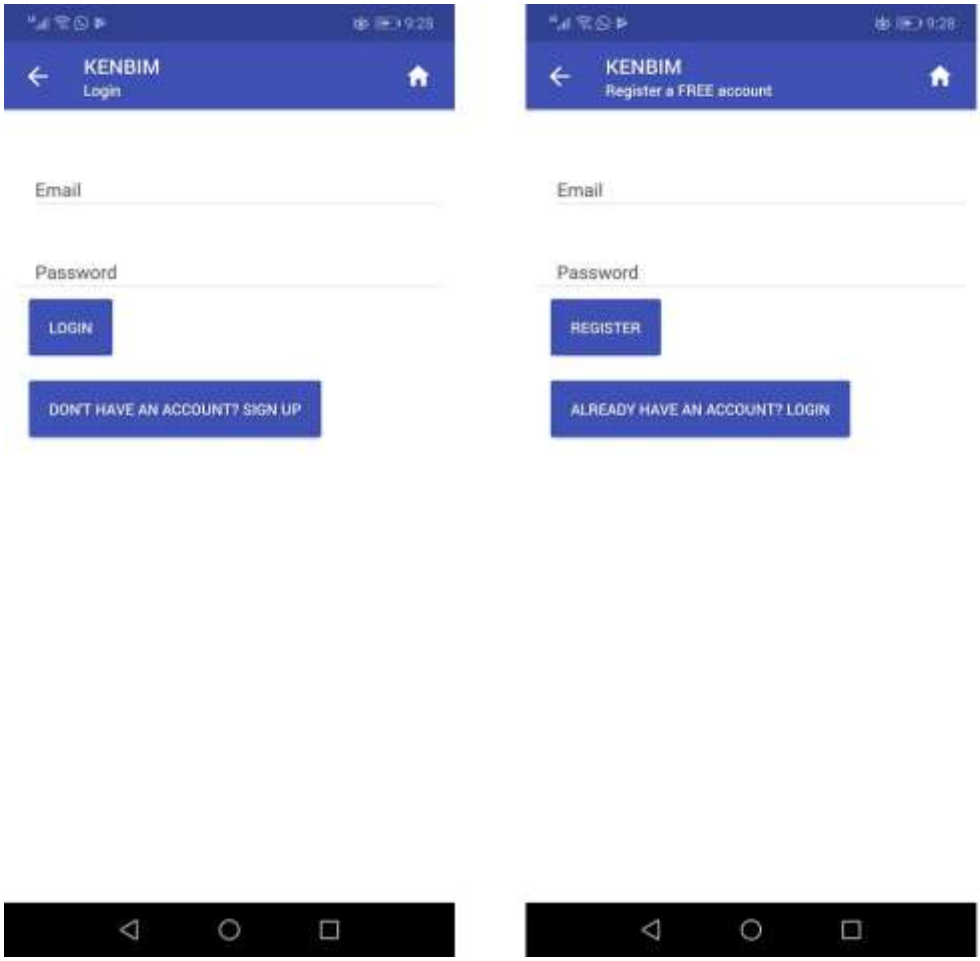


Figure 5.7: Login and Register Screens

5.4 System Implementation

Evolutional prototyping methodology was used to design and develop the system as discussed in chapter three. The system, named Kenyan Bird Identifier and Mapper (KENBIM), has a front-end and a back-end. The front-end is a mobile application developed using React Native and JavaScript and built for Android mobile devices. The back-end uses Google's Backend-as-a-Service Firebase cloud platform. The first prototype created could classify five species as a proof-of-concept. Subsequent prototypes had more species added. Saving classification results and drawing a bird map features were added to the last prototype.

5.4.1 Development Environment

The system was developed in the following hardware and software environment.

i. Lenovo ThinkPad L460 x64-based PC
ii. Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz, 2401 Mhz, 2 Core(s), 4 Logical Processor(s)
iii. 8GB System RAM
iv. 500GB Hard Disk
v. Microsoft Windows 10 Enterprise Operating System
vi. Microsoft Visual Studio Code
vii. Python 3.7
viii. Tensorflow 1.13.1
ix. Tensorboard 1.13.1
x. Numpy 1.16.2
xi. React 16.8.6
xii. React Native 0.59.3
xiii. Firebase 5.9.2
xiv. Native Base 2.12.1

5.4.2 Image Dataset Collection

Images were collected from Google Open Images, social media sites like www.flickr.com and the researcher's own images. A Python script was used to automatically download images from Google Images using specified keywords. However, the researcher had to manually collect

images from the other image sources. An example of the commands run is, “googleimagesdownload -k "White-bellied Bustard" -o "F:\Pictures\birds\White-bellied Bustard" -t photo -n -nn -cd "C:\\Users\\user\\Documents\\Soft\\chromedriver\\chromedriver.exe" -l 500”

The images were downloaded or put into corresponding folders whose folder name would be the image label. The folder name naming convention was “common_name;scientific_name”. This enabled the researcher to associate the common name with the scientific name without requiring a database call. Appendix B shows an example of the image download command and the resulting folder structure that was created.

5.4.3 Image Dataset Pre-Processing

A Python script, shown in Appendix C below, was used to automatically augment images according to the parameters given. The augmentation operations were chosen experimentally to see the best fit. The final augmentation operations performed were, randomly change 20% of the images to greyscale, flip 50% of the images horizontally and vertically, rotate by maximum 10degrees 70% of the images, skew and tilt 40% of the images, zoom 50% of the images and distort 40% of the images.

Augmentation enabled the researcher to create thousands of images from the original images, while retaining label data. The command to run the augmentation is “python /path/to/augmentor.py /path/to/image/folder”. The first prototype was created to identify six bird species. After augmentation, the initial image dataset consisted of 15,607 images of these six bird species and another set of 6,300 pictures that were not birds. The “Not a bird” images would train the network to recognize pictures that did not contain any birds. Therefore, the initial image dataset contained a total of 21,907 images. Figure 5.8 below shows an example of the running augmentation command and Figure 5.9 below shows samples of the resulting augmentations.

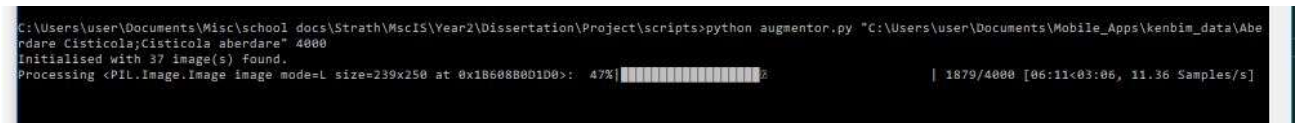


Figure 5.8: Image Augmentation Example

Figure 5.9: Sample Image Augmentations

5.4.4 Re-Training the CNN Model

The MobileNet version two neural network was retrained using Tensorflow's retraining script to perform bird species classification. The command actually used is shown in the Figure 5.10 below. This results in a Tensorflow model file and a labels text file.

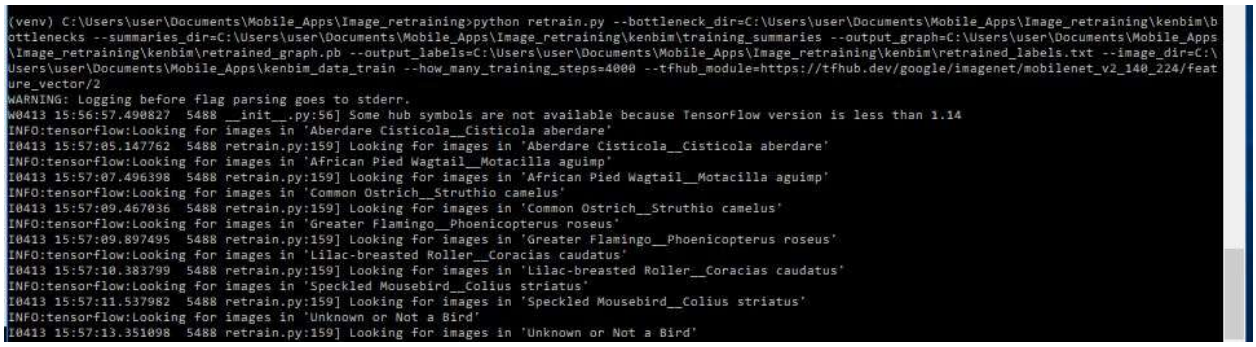


Figure 5.10: Retraining Command

The initial dataset of 21,907 images for six species took about 1 hour to train on the researcher's laptop PC. The next prototype had a dataset of 39,031 images for twelve species, which took 3 hours to train on the researcher's laptop PC.

5.4.5 Converting Tensorflow Model to Tensorflow Lite

The generated Tensorflow model was converted to a Tensorflow lite model using Tensorflow's Lite converter tool, tflite_convert. The command below was used to convert the generated model into a Tensorflow lite model. "tflite_convert --graph_def_file=C:\kenbim\retrained_graph.pb --output_file=C:\kenbim\optimized_graph.lite --input_format=TENSORFLOW_GRAPHDEF --output_format=TFLITE --input_shape=1,224,224,3 --input_array=Placeholder --output_array=final_result --inference_type=FLOAT --input_data_type=FLOAT"

Input graph is the Tensorflow model and the output file is the desired Tensorflow lite file. The input and output array are as per the neural network model architecture showed in Figure 5.1 above.

5.4.6 Building the Android Mobile Application

React Native was installed and initiated. The react native dependencies packages were installed, as per the packages.json file shown in Appendix D, and then linked to the initiated React Native. To build for android, the following command was run; "react-native bundle --platform android --dev false --entry-file index.js --bundle-output android/app/src/main/assets/index.android.bundle --assets-dest android/app/src/main/res && react-native run-android"

The resulting android APK file was then installed in a test mobile device.

5.5 System Testing

Tests performed on both the neural network model and the mobile application are described in this section.

5.5.1 CNN Model Validation and Testing

Cross entropy, the loss function used, measured the classification performance of the CNN model. A cross entropy of zero is the most desired value. On the 177^{th} training step the cross entropy was 0.3600 for the training dataset and 0.3834 for the validation training set. The final cross entropy figures at the end of the training steps were 0.04665 for the training dataset and 0.1101 for the validation dataset. The cross entropy dropped as the number of training steps increased, showing that the network was learning. The Tensorflow training and validation accuracy and the cross entropy values are shown in Figure 5.11 below. Figure 5.12 shows the cross entropy graph.

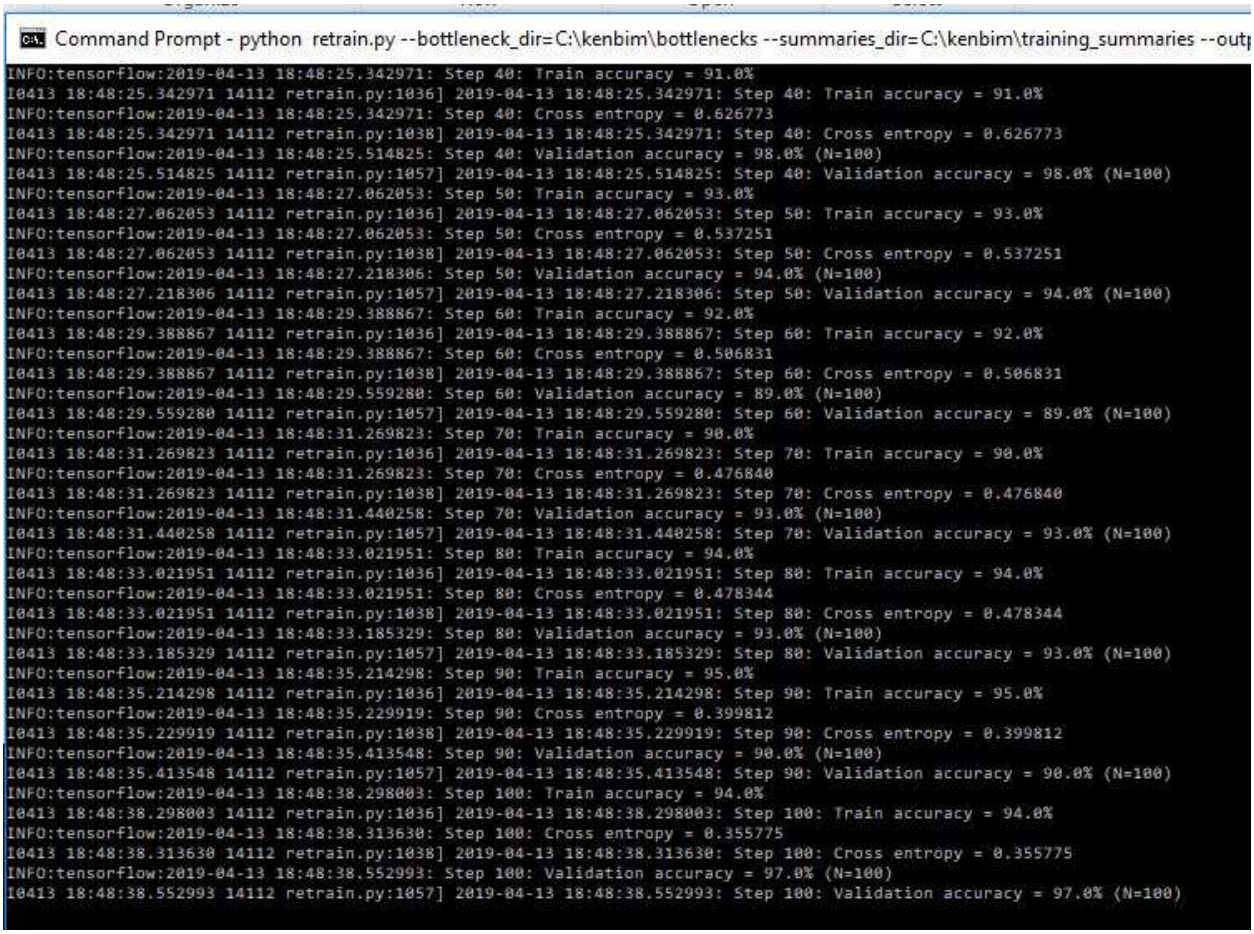


Figure 5.11: Training and Validation Accuracy

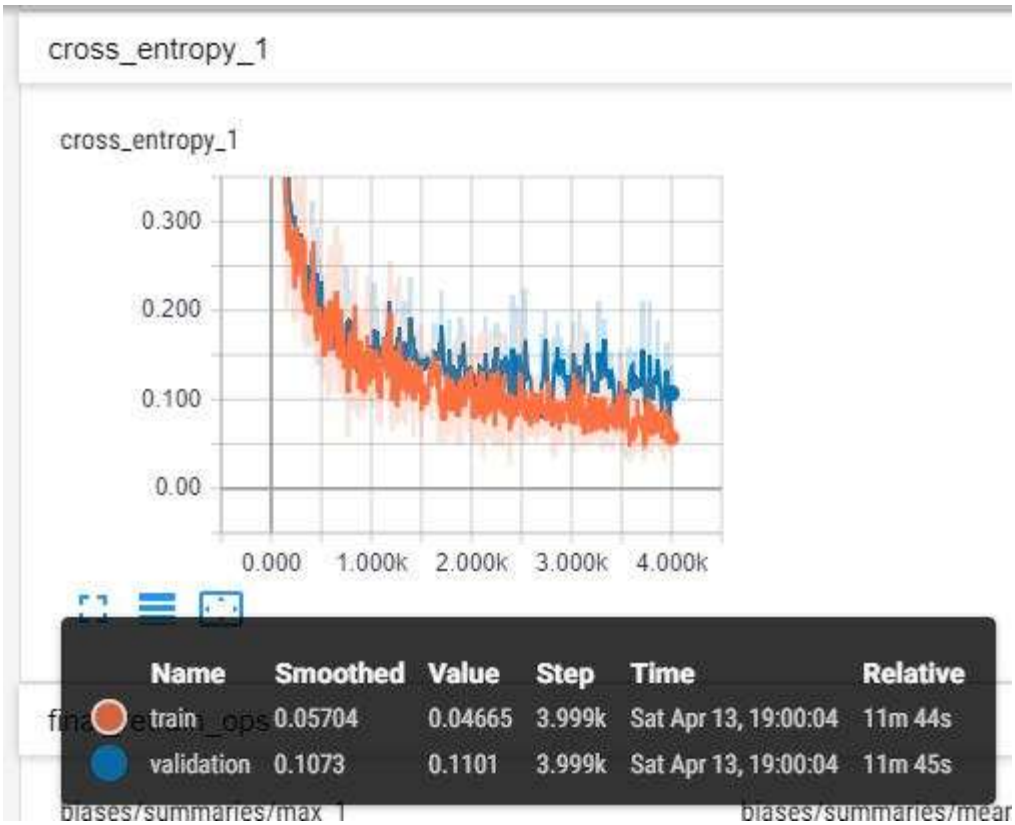


Figure 5.12: Cross Entropy Graph

The final test accuracy achieved 97.3% as shown below in Figure 5.13 below.

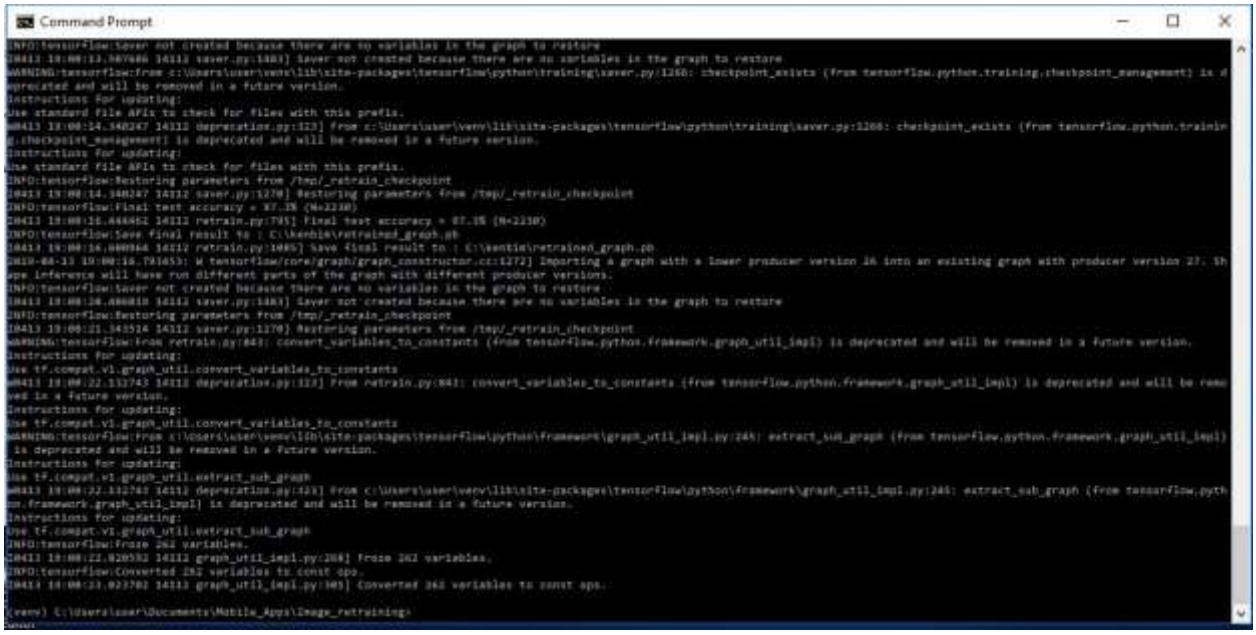


Figure 5.13: Final Test Accuracy

Finally, the researcher tested the resulting model via a Python command line tool called label_image.py. The model returned correct results as shown in Figure 5.14 below.

Figure 5.14: Training and Testing Accuracy Example

5.5.2 Image Classification in the Mobile Application

The image classification feature of the mobile application was tested on an actual mobile device by feeding it with images and seeing what classification the application would give them. The researcher selected colour and greyscale images and images that were not birds. Figure 5.15 shows the test results. The colour image was correctly labelled with a high confidence of 0.787. The greyscale image was also correctly labelled but with a lower confidence of 0.757. The image that was not a bird was also correctly labelled as not being a bird, with a 0.55 chance that it is

indeed not a bird. In all the image classification test cases, inference time was less than 400milliseconds.

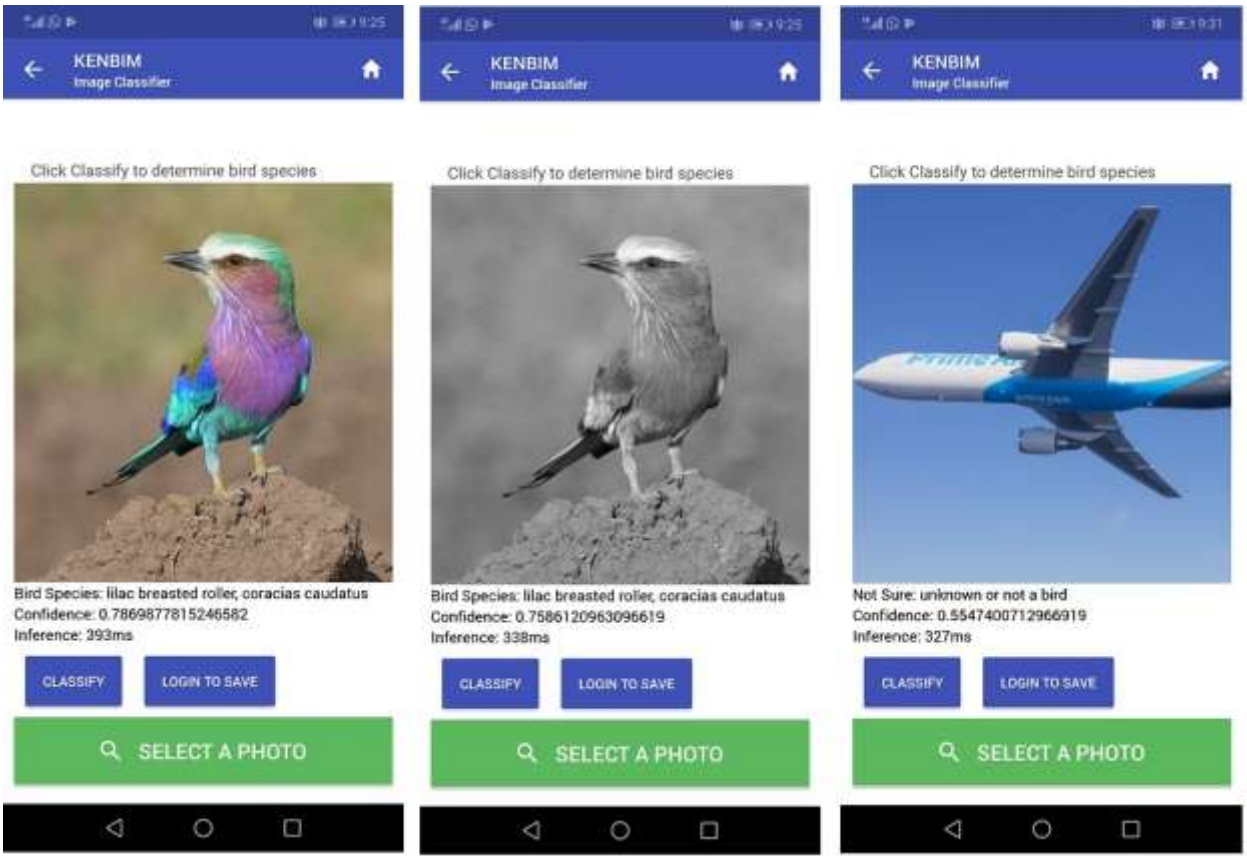


Figure 5.15: In-App Image Classification Tests

5.5.3 Storing Bird Observations

A registered user was able to save their bird observations in the Firebase cloud database. The mobile application would retrieve image metadata like location co-ordinates and timestamp the image was taken. If location co-ordinates where missing, the user would input them manually. Figure 5.16 below shows the user input prompt for entering missing location details. Figure 5.17 below shows an example of the Firebase backend with some saved observations.

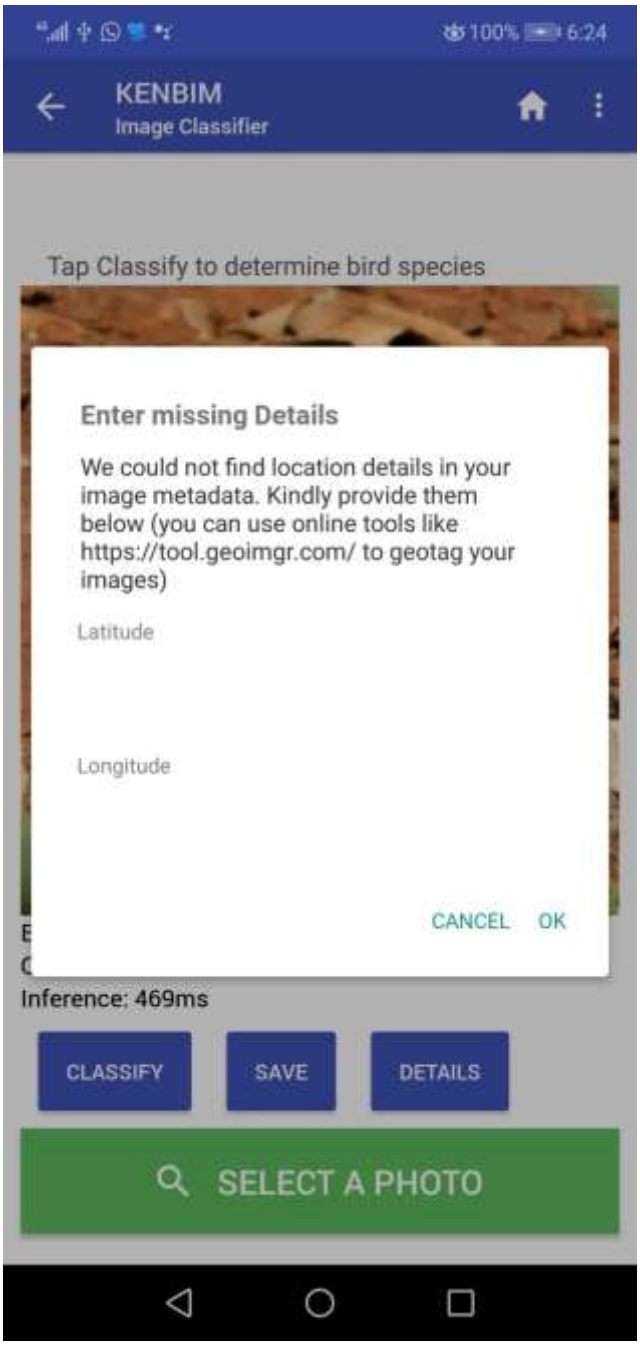


Figure 5.16: Getting Missing Details from the User

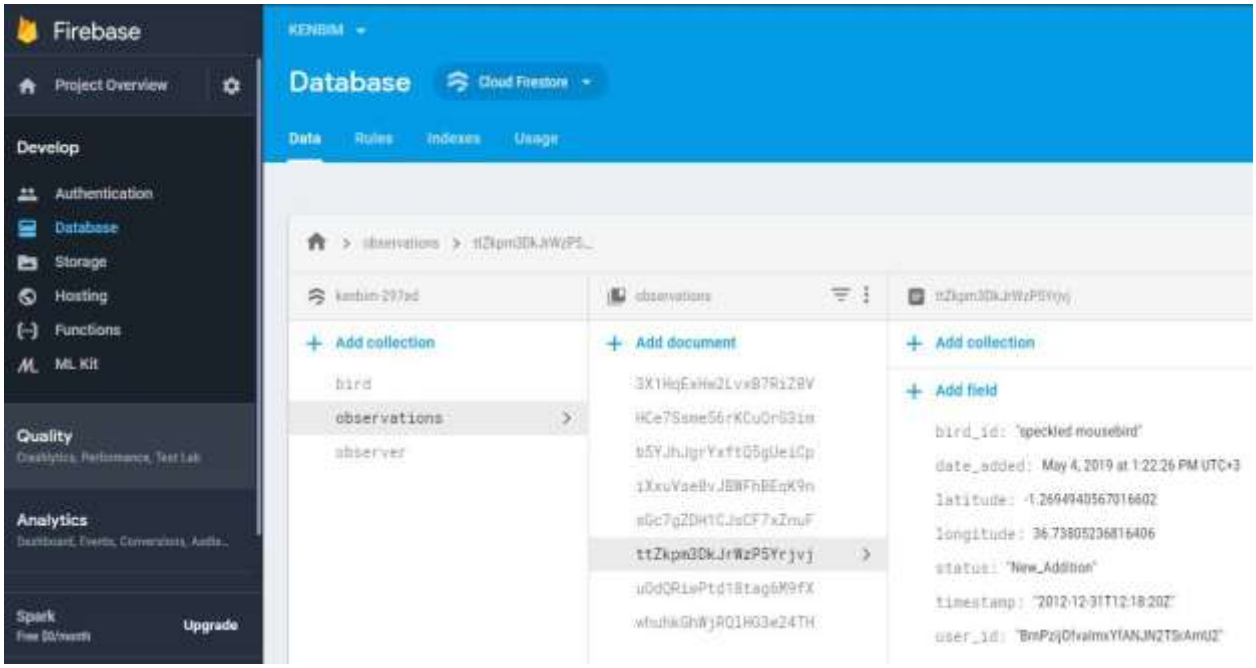


Figure 5.17: Saved Bird Details Example

5.5.4 Viewing the Bird Map

A registered user was able to view the bird observations that they have saved in form of a bird distribution map. The user would tap on “My Bird Map” button in the home screen shown in Figure 5.5 and their saved bird observations would appear as markers in an embedded Google map. Tapping on an individual marker revealed extra details about that particular bird observation. Figure 5.18 below shows an example of a user’s bird map.

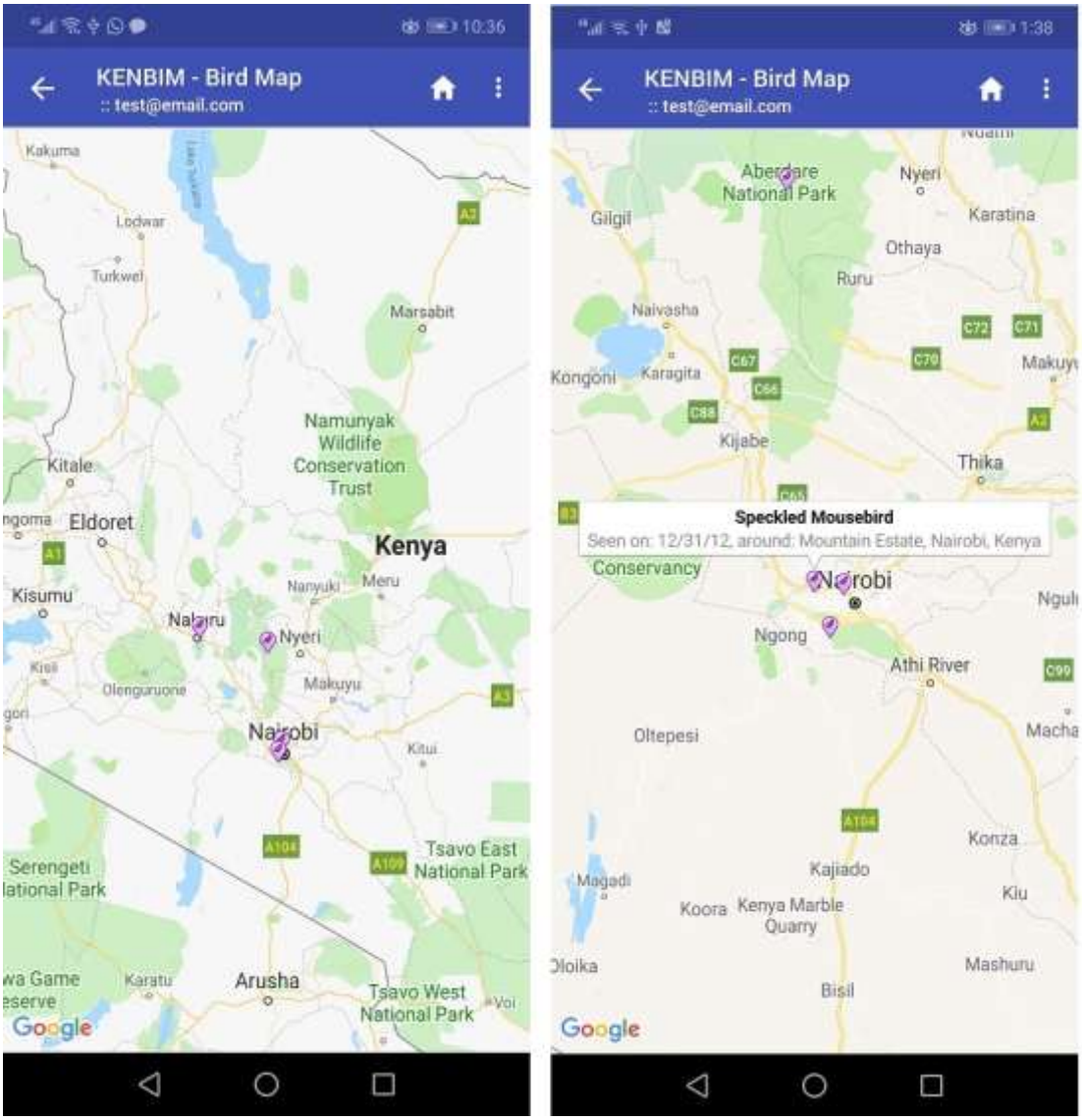


Figure 5.18: A User's Bird Map Example

Chapter 6
Discussions

6.1 Introduction

This chapter discusses the research findings, reviews the solution that was developed and whether the research objectives where achieved. This research study analyzed current methods of bird species identification and machine learning methods of image recognition. After reviewing these current methods, the researcher chose to focus on deep convolutional neural networks that are well suited for image processing tasks. An Android mobile application, with an embedded convolutional neural network that is suited for mobile devices that have constrained computing resources, was developed.

6.2 Model Validation

This study employed transfer learning in which a pre-existing convolutional network, MobileNet version 2 in particular, was retrained and fine tuned for fine-grained bird species recognition. The researcher used an image dataset of 39,031 images. These images were mostly sourced from Google Open Images as discussed in Section 5.4.2 above. 80% of the images were used to create the training dataset. The remaining 20% of the images were used for validation and testing. 10% of the images were used to run the model validation during the training process and the last 10% were used as a test dataset to predict the real-world performance of the resulting model. The true performance of the model was measured by measuring its performance on the test dataset, which was not contained in the training data. The testing accuracy represents how well the trained model would classify completely new images. The final test accuracy achieved was 97.3%. The test accuracy shows what percent of images used in the testing process were labelled with the correct label. From Figure 5.12 above it was shown that the loss function was dropping to almost zero. Thus high test accuracy and a low loss function indicate that the model learnt from the training data and can thus classify new images. The Figure 6.1 below shows the accuracy graph. Appendix G shows the final training metrics in graphical form.

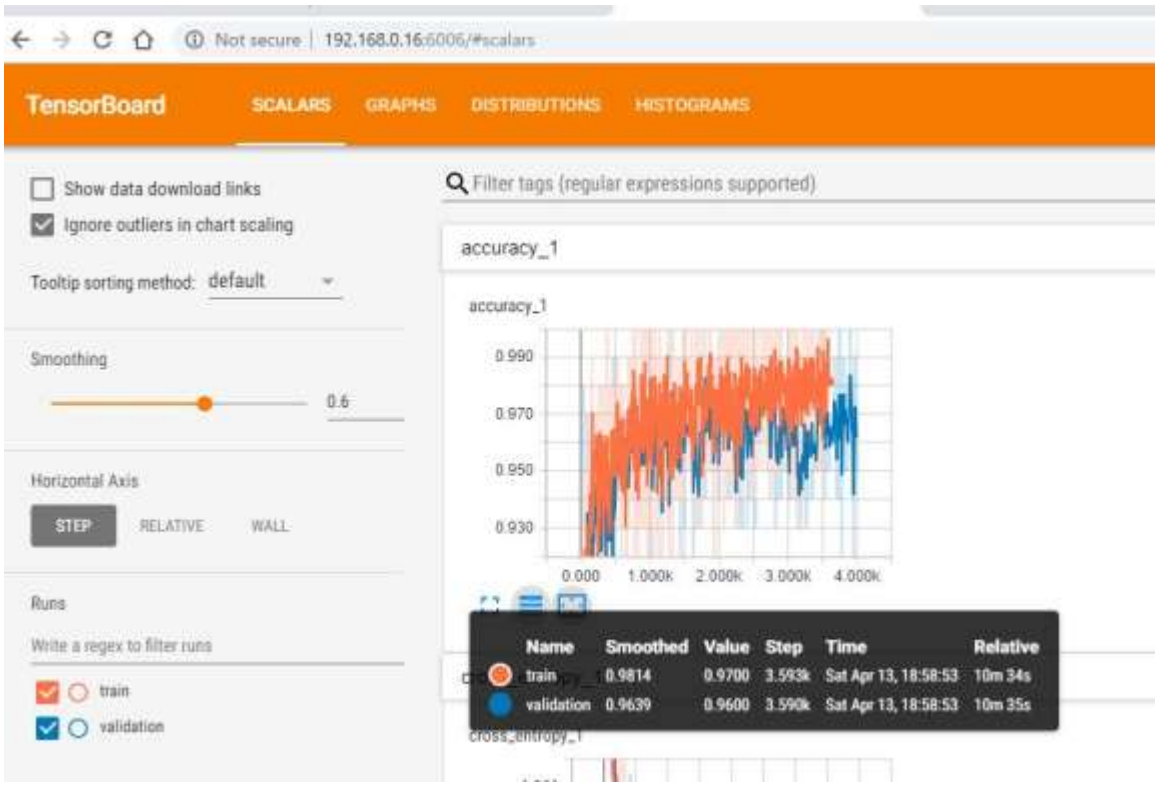


Figure 6.1: Training Accuracy Graph

6.3 Image Classifier Validation

Image classification validation was done using 15 different images of each of the 11 bird species, including 15 images of not-a-bird images. Therefore there were a total of 180 images. The Confusion Matrix was used to measure the performance of the classification system. The model performed well in determining which image had or did not have a bird as shown in Table 6.1 below.

Table 6.1 Bird Classification Confusion Matrix

	Classified as 'Bird'	Classified as 'Not a Bird'	
Bird	TP = 165	FN =0	165
Not a Bird	FP =1	TN = 14	15
	166	14	180

i. **True Positive (TP):** An image containing a bird is correctly identified as containing a bird.
ii. **False Negative (FN):** An image containing a bird is incorrectly identified as not containing a bird.
iii. **False Positive (FP):** An image not containing a bird is incorrectly identified as containing a bird.
iv. **True Negative (TN):** An image not containing a bird is correctly identified as not containing a bird.

The following metrics were generated from the confusion matrix.

i. **Accuracy** – Measures how often the classifier was correct.
   (TP + TN) / Total = (165 + 14) / 180 = 0.994
ii. **Miscalculation Rate** – Measures how often the classifier was wrong.
   (FP + FN) / Total = (1 + 0) / 180 = 0.006
iii. **Precision** – Measures when it predicts a bird, how often is it was correct.
   TP / Classified Bird = 165 / 166 = 0.994
iv. **True Positive or Recall Rate** – Measures when it is actually a bird, how often did it classify it as a bird. TP/Classified as Bird = 165/165 = 1
v. **True Negative Rate** – Measures when it is not a bird, how often did it classify it as not a bird.
   TN/ Classified not a Bird = 14/15 = 0.933
vi. **Prevalence** – Measures how often classified as bird occurred in the sample.
   Total Birds/ Total = 165 / 180 = 0.917

The confusion matrix for the respective bird species is shown below in Table 6.2

Table 6.2 Fine-Grained Classification Confusion Matrix

Actual Label	Selected Label												Accuracy
	Aberdare Cisticola	African Pied Wagtail	Crested Guineafowl	Common Ostrich	Greater Flamingo	Kikuyu white-eye	Lilac-breasted Roller	Ruppell's Robin-Chat	Speckled Mousebird	Speke's Weaver	White-browed Robin-chat	Unknown or Not a Bird	
Aberdare Cisticola	**13**	0	0	0	0	0	0	0	2	0	0	0	**0.983333**
African Pied Wagtail	0	**14**	0	0	0	0	1	0	0	0	0	0	**0.988889**
Crested Guineafowl	0	0	**14**	0	0	0	0	1	0	0	0	0	**0.988889**
Common Ostrich	0	0	0	**15**	0	0	0	0	0	0	0	0	**1**
Greater Flamingo	0	0	0	0	**13**	1	0	0	1	0	0	0	**0.983333**
Kikuyu white-eye	0	1	0	0	1	**11**	0	0	1	0	1	0	**0.966667**
Lilac-breasted Roller	0	0	0	0	0	0	**14**	0	1	0	0	0	**0.988889**
Ruppell's Robin-Chat	0	0	0	0	0	0	0	**10**	2	0	3	0	**0.961111**
Speckled Mousebird	0	0	0	0	0	1	0	0	**13**	1	0	0	**0.944444**
Speke's Weaver	1	0	0	0	0	0	0	0	0	**14**	0	0	**0.977778**
White-browed Robin-chat	0	0	0	0	0	0	0	1	1	2	**11**	0	**0.955556**
Unknown or Not a Bird	0	0	1	0	0	0	0	0	0	0	0	**14**	**0.994444**

The correct classifications are located in the diagonal of the table and highlighted in bold. From the above Table 6.2 the image classifier performed relatively well on all classes. Unique bird

species like the Common Ostrich had a perfect accuracy, that is being classified correctly in all tests and not being wrongly classified in any other test. Other bird species with unique characteristics like the Crested Guineafowl and the African Pied Wagtail had near perfect accuracies. Bird species with subtle differences like the Ruppell's Robin-chat and the White-browed Robin-chat had lower accuracies. The outlier case revealed is that of the Speckled Mousebird, which had the lowest accuracy of 94.4%. Nevertheless, this accuracy is still high and is acceptable for fine-grained image classification.

6.4 More Bird Details

The researcher was able to add extra bird details feature into the image recognition system. This enabled the user to get extra details i.e. thumbnail picture, taxonomy details and brief description about the bird species identified without having to leave the system. This was done to enhance the user experience. Figure 6.2 below shows an example of extra bird details.

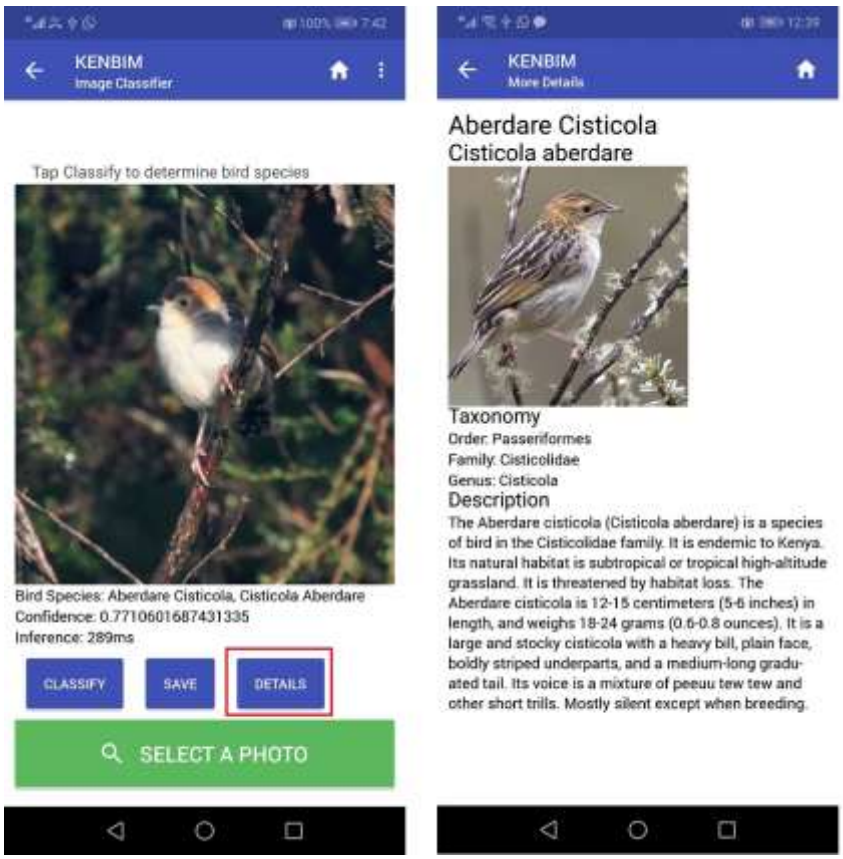


Figure 6.2: More Bird Details Example

6.5 Advantages of the Developed System to Current Systems

The image dataset created during this research study is the most significant advantage of the developed system to current systems. Current bird identification applications focus on American or European bird species. The developed image recognition system provided a solution to anyone wanting to determine the Kenyan bird species in an image. There is no human-in-the-loop methods employed. The user feeds the system an image and the system returns back a result. This improves the user experience especially for the beginners. This research study has demonstrated that the system administrator does not need to have the image dataset part annotated and bounding boxes set. The depth-wise separable convolutional neural network model created in this study is especially suited for mobile devices since it have low latency, uses less computational power while at the same time maintaining high accuracy. Finally, the developed system included a bird map feature, which none of the other systems have.

6.6 Shortfalls of the Research

The mobile-based image recognition system developed during this research study did not meet some of the expectations of the researcher.

i. The classifier would fail to correctly an image if two or more bird species were in the same photo.
ii. The image dataset did not have many bird species as earlier anticipated.
iii. The user feedback loop as envisioned in the conceptual framework was not actualized.

6.7 Challenges Encountered

It was a challenge getting good quality bird images, especially the rarer bird species like the Aberdare Cisticola or the Taita Apalis. In fact, the reason the Taita Apalis was no included in this prototype is because the research could not find sufficient images in the research time frame. It was time consuming sifting the collected images and removing unwanted or low quality images and sorting the collected images into the respective folders. The researcher trained the neural network using a Central Processing Unit (CPU) only since he had no access to a Graphics Processing Unit (GPU). A GPU performs matrix operations like convolution much more quickly than a CPU (LeCun, Bengio, & Hinton, 2015).

Chapter 7
Conclusions and Recommendations

7.1 Conclusions

The objective of this research study was to develop a mobile application that can identify Kenya bird species from a photograph and create a bird map of the observations. The problem of fine-grained visual categorization is difficult for both humans and machine alike. It is easy for a human to recognize that a given image is a bird. It is more difficult to determine the bird species in the image. Thus, this study began by reviewing current methods of bird species identification and their challenges. Machine learning techniques that support image recognition were reviewed and the research chose to focus on transfer learning and convolutional neural networks that are well suited for image processing tasks. This study showed that advances in computer vision have been able to create deep neural networks that can effectively and efficiently perform fine-grained categorization. This research then presented a detailed approach on how to solve the problem of bird species identification using transfer learning and came up with a mobile application for bird species identification.

The model developed performed satisfactorily well and was able to infer the bird species in an image provided by a user and provide more details of the inferred bird species. The user could then save the classification results into a cloud database and later on retrieve the same in form of a bird map. Combining bird identification, more details and bird distribution mapping in a single application is a novel approach introduced in this research study.

7.2 Recommendations

Based on the results of this research study, the researcher recommends the following

i. Breakdown the bird identification into further subclasses like male, female, juvenile, breeding female etc.
ii. Show possible suggestions if the bird species cannot be correctly identified.

7.3 Suggestions for Future Research

More research needs to be done on improving the fine-grained capabilities of the developed model. Also future research can explore identification of multiple birds in an image. Future research can also look into using *both* audio and visual data to correctly identify a bird species.

References

Arel, I., Rose, D. C., & Karnowski, T. P. (2010). Deep Machine Learning-A New Frontier in Artificial Intelligence Research. *IEEE Computational Intelligence Magazine, 5*(4), 13-18.

Basic Research Designs: CIRT. (2018, September 12). Retrieved from Center for Innovation in Research and Teaching: https://cirt.gcu.edu/research/developmentresources/tutorials/researchdesigns

Berg, T., Liu, J., Lee, S. W., Alexander, M. L., Jacobs, D. W., & Belhumeur, P. N. (2014). Birdsnap: Large-scale Fine-grained Visual Categorization of Birds. *2014 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2019-2026). Columbus, OH, USA: IEEE. doi:10.1109/CVPR.2014.259

Birdwatching tourism from Europe. (2017, November 13). Retrieved from Centre for the Promotion of Imports from developing countries (CBI): https://www.cbi.eu/market-information/tourism/birdwatching-tourism/

Branson, S., Horn, G. V., Wah, C., Perona, P., & Belongie, S. (2014). The Ignorant Led by the Blind: A Hybrid Human–Machine Vision. *International Journay of Computer Vision*(108), 3-29.

Brynjolfsson, E., & Mcafee, A. (2017, July). *The Business of Artificial Intelligence.* Retrieved from Harvard Business Review Website: https://hbr.org/cover-story/2017/07/the-business-of-artificial-intelligence

Brynjolfsson, E., & Mcafee, A. (2017, July 18). *What's Driving the Machine Learning Explosion?* Retrieved from Harvard Business Review Website: https://hbr.org/2017/07/whats-driving-the-machine-learning-explosion

Building Skills: The 4 Keys to Bird Identification. (2009, April 20). Retrieved March 18, 2018, from All About Birds Website: https://www.allaboutbirds.org/building-skills-the-4-keys-to-bird-identification/

Custom Image Augmentation: Towards Data Science. (2017, July 9). Retrieved September 18, 2018, from Towards Data Science: https://towardsdatascience.com/image-augmentation-14a0aafd0498

Dennis, A., Wixom, B. H., & Tegarden, D. (2009). *Systems Analysis & Design with UML version 2.0. An Object Oriented Approach 3rd Edition.* New Jersey: John Wiley & Sons, Inc.

Devikar, P. (2016, December). Transfer Learning for Image Classification of various dog breeds. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 5*(12), 2707-2715.

Ethical Considerations: CIRT. (2018). Retrieved April 2, 2018, from Center for Innovation in Research and Teaching: https://cirt.gcu.edu/research/developmentresources/tutorials/ethics

Filip, S. (2018). *Deep Neural Networks for Classification of Product Images.* Prague: Czech Technical University.

Horn, G. V., Branson, S., Farrell, R., Haber, S., Barry, J., Ipeirotis, P., . . . Belongie, S. (2015). Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 595-604). Boston, MA, USA: IEEE. doi:10.1109/CVPR.2015.7298658

How to Identify Birds. (2018). Retrieved from National Audubon Society Website: http://www.audubon.org/content/how-identify-birds

How to Identify Birds by Sounds. (2018). Retrieved from National Audubon Society Website: http://www.audubon.org/news/how-start-identifying-birds-their-songs-and-calls

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., . . . Adam, H. (2017, April 17). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.* Retrieved from arXiv.org: https://arxiv.org/abs/1704.04861v1

Identification of Birds. (2016). Retrieved from Ornithology: The Science of Birds Website: https://ornithology.com/identification/

IIango, G., & Kumar, S. (2017). Flower Species Recognition System using Convolution Neural Networks and Transfer Learning. *4th International Conference on Signal Processing, Communications and Networking* (pp. 1-6). Chennai: ICSCN.

Image Recognition: MathWorks. (2018). Retrieved from MathWorks Website: https://www.mathworks.com/discovery/image-recognition.html

LeCun, Y., Bengio, Y., & Hinton, G. (2015, May 28). Deep Learning. *Nature: International Journal of Science, 521*, 436-444.

Lee, A. (2016, January 21). *Traditional Computer Vision vs.Deep Neural Networks.* Retrieved July 07, 2018, from Swarthmore College Website: https://www.cs.swarthmore.edu/~meeden/cs81/f15/papers/Andy.pdf

Marketing Analysis of Bird-Based Tourism. (2011). Retrieved from Center for Responsible Travel Website: http://www.responsibletravel.org/docs/Market%20Analysis%20of%20Bird-Based%20Tourism.pdf

Merlin Bird ID. (2018). Retrieved from All About Birds Website: http://merlin.allaboutbirds.org/

Nielsen, M. A. (2015). *Neural Networks and Deep Learning.* Determination Press.

Okello, M. M. (2014). Economic Contribution, Challenges and Way Forward for Wildlife-Based Tourism Industry in Eastern African Countries. *Journal of Tourism & Hospitality, 3*(1), 122-134.

Olah, C. (2014, July 8). *Conv Nets: A Modular Perspective*. Retrieved from Colah's Blog: https://colah.github.io/posts/2014-07-Conv-Nets-Modular/

Ornithology Section: National Museums of Kenya. (2018). Retrieved from National Museums of Kenya Website: http://www.museums.or.ke/ornithology-section/

Robert, C. (2014). Machine Learning, a Probabilistic Perspective. *Chance, 27*(2), 62-63.

Sandler, M., & Howard, A. (2018, April 3). *MobileNetV2: The Next Generation of On-Device Computer Vision Networks.* Retrieved from Google AI Blog: https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2019, March 21). *MobileNetV2: Inverted Residuals and Linear Bottlenecks.* Retrieved from arXiv.org: https://arxiv.org/abs/1801.04381v4

Shaoli, H., & Tao, D. (2016, October). Real Time Fine-Grained Categorization with Accuracy and Interpretability. *Computing Research Repository (CoRR), abs/1610.00824*.

Sinclair, I. (1990). *Southern African Birds: A photographic Guide.* Cape Town: Struik Publishers.

Singh, H. (2018, September 5). *Reasons Why Use React Native for Mobile App Development.* Retrieved from DEV Community: https://dev.to/theninehertz/reasons-why-use-react-native-for-mobile-app-development-1idl

TensorFlow. (2018). Retrieved from TensorFlow Website: https://www.tensorflow.org/

The Internet Bird Collection. (2018). Retrieved from The Internet Bird Collection Website: https://www.hbw.com/ibc

U.S. Fish & Wildlife Service. (2016, August). *National Survey of Fishing, Hunting and Wildlife-Associated Recreation.* U.S. Fish & Wildlife Service. Retrieved from U.S. Fish &

Wildlife Service Website: https://wsfrprograms.fws.gov/subpages/nationalsurvey/nat_survey2016.pdf

Wachira, W., Jackson, C., & Njoroge, P. (2015, July). Kenya Bird Map: an internet-based system for monitoring bird distribution and populations in Kenya. *Scopus: Journal of East African Ornithology, 34*, 58-60. Retrieved from https://www.ajol.info/index.php/scopus/article/view/111996/101755

Wah, C., Branson, S., Welinder, P., Perona, P., & Belongie, S. (2011). *The Caltech-UCSD Birds-200-2011 Dataset.* California Institute of Technology. CA: Computation & Neural Systems Technical Report. Retrieved from http://www.vision.caltech.edu/visipedia/papers/CUB_200_2011.pdf

Why is Bird Watching So Popular? (2018, March 14). Retrieved from Seasonedtimes Website: https://www.seasonedtimes.com/why-is-bird-watching-so-popular-.html

Why we need Throw-away Prototyping. (2017, August). Retrieved from Software Engineering Stackexchange Website: https://softwareengineering.stackexchange.com/questions/253574/why-we-need-throw-away-prototyping

Zab, S. (2015, June 2). *The Growth of The Smartphone Market in Kenya.* Retrieved from Jumia Blog: https://www.jumia.co.ke/blog/whitepaper-the-growth-of-the-smartphone-market-in-kenya/

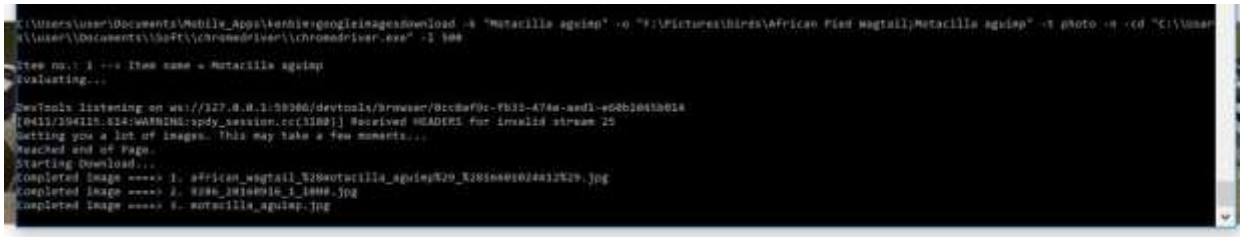
Appendices

Appendix A: Originality Report

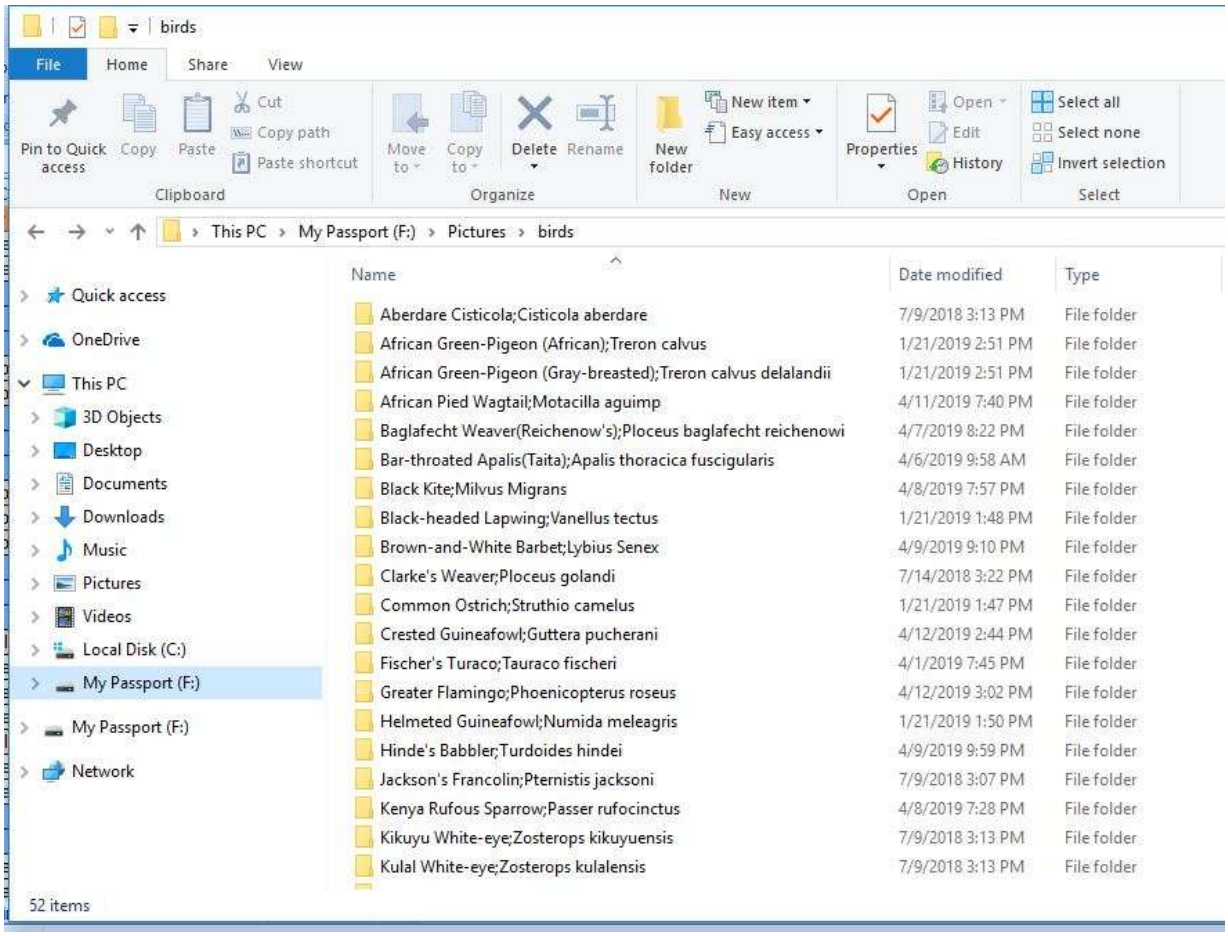
The Turnitin report summary

Appendix B: Data Collection Commands and Folder Structure

Example Command to download images from Google Open Images.



The dataset folder structure.



Appendix C: Image Augmentation Script

Python Script: Augmentor.py

```
"""
Image Augmentation script
by Gideon Nyaga

Usage: python /path/to/augmentor.py /path/to/image/folder [no_of samples]
"""
import sys
import Augmentor

image_path = sys.argv[1] #set the image path where the images are
no_of_samples = sys.argv[2] #set number of samples to be generated
#set default no of samples
if no_of_samples == 0:
    no_of_samples = 100
    pass
#create an Augmentor pipeline
pl = Augmentor.Pipeline(image_path)

#Add operations to the pipeline
pl.greyscale(probability=0.2) #convert images to gray scale
pl.flip_left_right(probability=0.5)# Now we add a horizontal flip
pl.flip_top_bottom(probability=0.5)# Now we add a vertical flip
pl.rotate(probability=0.7, max_left_rotation=10, max_right_rotation=10)
pl.zoom(probability=0.5, min_factor=1.1, max_factor=1.5)
pl.gaussian_distortion(probability=0.4, grid_width=7, grid_height=6, magnitude=6,
corner="ul", method="in", mex=0.5, mey=0.5, sdx=0.05, sdy=0.05)
pl.skew(0.4,0.5)
pl.skew_tilt(0.6,0.8)
pl.skew_left_right(0.5, magnitude=0.8)

# Now we can sample from the pipeline:
pl.sample(int(no_of_samples))
```

Appendix D: React Native App packages.json File

```
{
  "name": "kenbim",
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "start": "node node_modules/react-native/local-cli/cli.js start",
    "test": "jest",
    "android": "react-native bundle --platform android --dev false --entry-file
index.js --bundle-output android/app/src/main/assets/index.android.bundle --
assets-dest android/app/src/main/res && react-native run-android"
  },
  "dependencies": {
    "firebase": "^5.9.2",
    "native-base": "^2.12.1",
    "react": "^16.8.6",
    "react-native": "0.59.3",
    "react-native-gesture-handler": "^1.1.0",
    "react-native-image-picker": "^0.28.1",
    "react-native-maps": "^0.24.2",
    "react-native-status-bar-height": "^2.3.1",
    "react-native-tensorflow-lite": "^1.0.6",
    "react-navigation": "^3.6.0",
    "realm": "^2.25.0"
  },
  "devDependencies": {
    "babel-jest": "24.5.0",
    "jest": "24.5.0",
    "metro-react-native-babel-preset": "0.53.1",
    "react-test-renderer": "16.8.6"
  },
  "jest": {
    "preset": "react-native"
  }
}
```

Appendix E: Validation Script File

Python Script: run_test.py used to generate the multi-class confusion matrix.

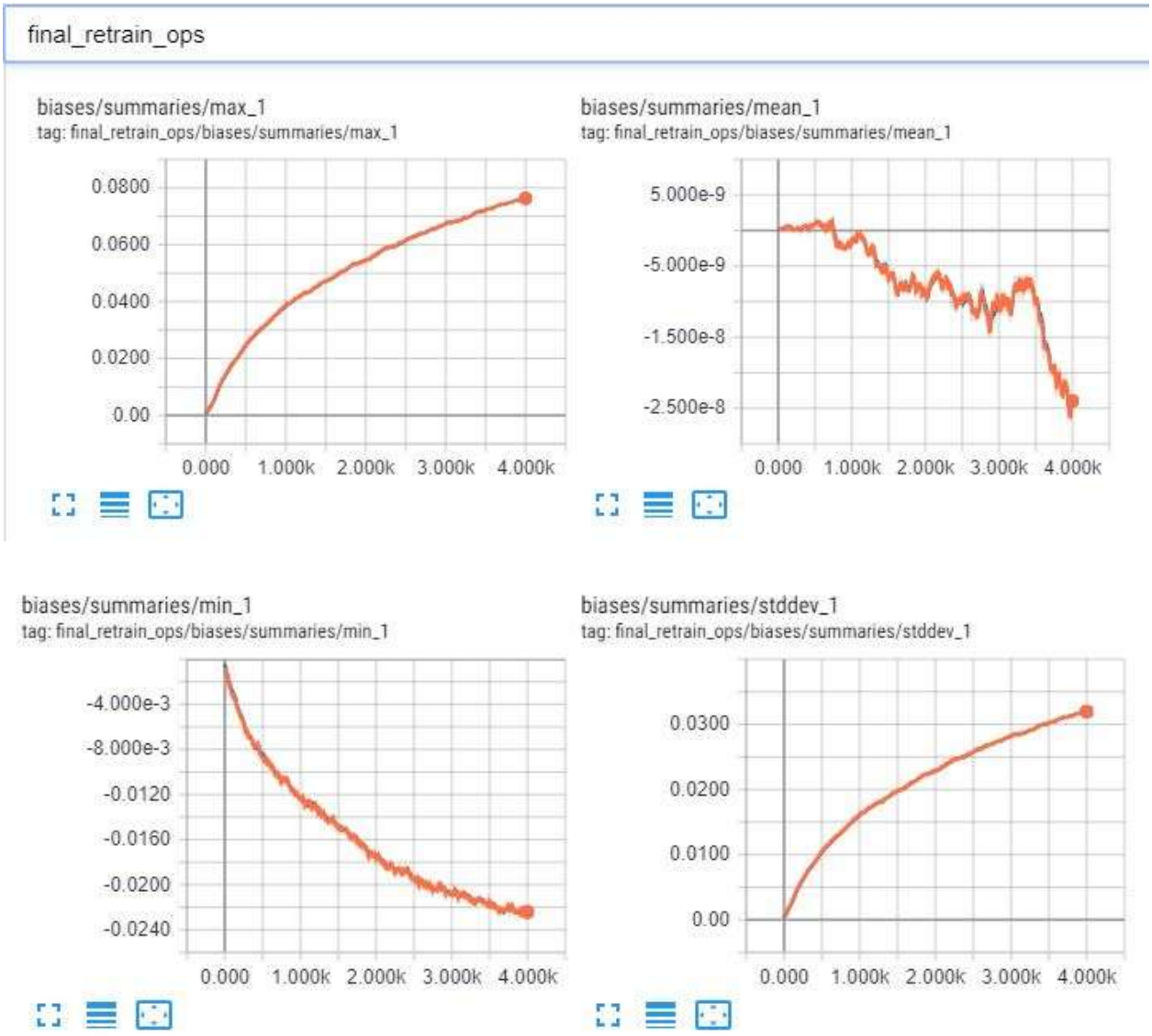
```
"""
Run automated validation tests on image classifier
by Gideon Nyaga

Usage: python /path/to/run_tests.py /path/to/image/folder output_file.txt
list_of_space separated filenames_to_be_included_in_array
e.g. C:\kenbim\scripts>python run_tests.py "C:\kenbim_data_train\Greater
Flamingo__Phoenicopterus roseus" flamingo.txt 488 499 504 975
"""
import sys
import os

image_path = sys.argv[1] #set the image path where the images are
outout_file = sys.argv[2] #set output text file
images = sys.argv[3:] #create list of images

for image in images:
    image_file = image_path+'\\'+str(image)+'.jpg'
    #print(image_file)
    os.system('python label_image.py --graph="c:\kenbim\\retrained_graph.pb" --
labels="c:\kenbim\\retrained_labels.txt" --input_layer=Placeholder --
output_layer=final_result --input_height=224 --input_width=224 --
image="'+image_file+'" >> '+outout_file)
```

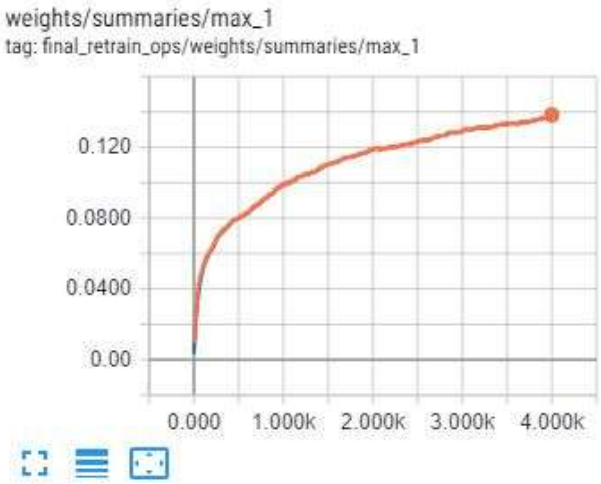
Appendix F: List of Bird Species

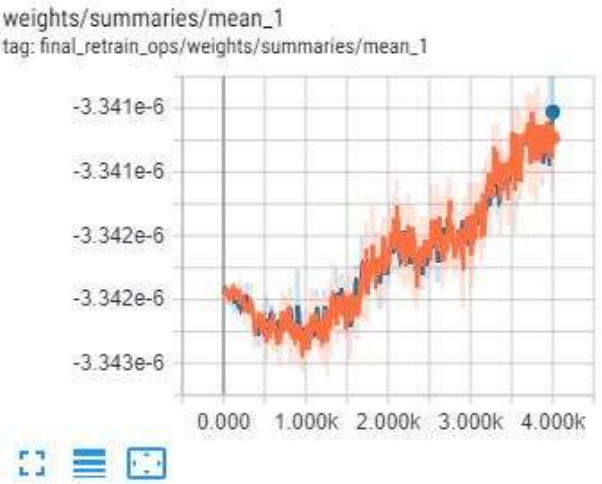
The list below contains the bird species that can be identified using this image recognition system as at the submission date of this paper. Note that the list may include more species in subsequent prototypes.

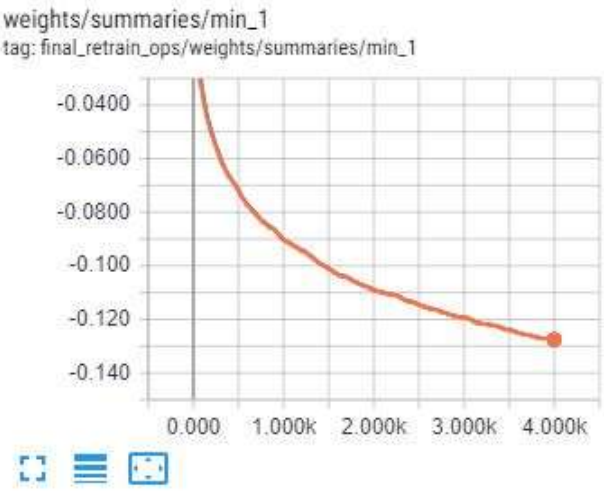
i. Aberdare Cisticola
ii. African Pied Wagtail
iii. Clark's Weaver
iv. Common Ostrich
v. Crested Guineafowl
vi. Greater Flamingo
vii. Grey-headed Kingfisher
viii. Kikuyu White-eye
ix. Lesser Flamingo
x. Lilac-breasted Roller
xi. Olive Thrush
xii. Ruppell's Robin-chat
xiii. Speckled Mousebird
xiv. Speckled Pigeon
xv. Speke's Weaver
xvi. White-browed Robin-chat

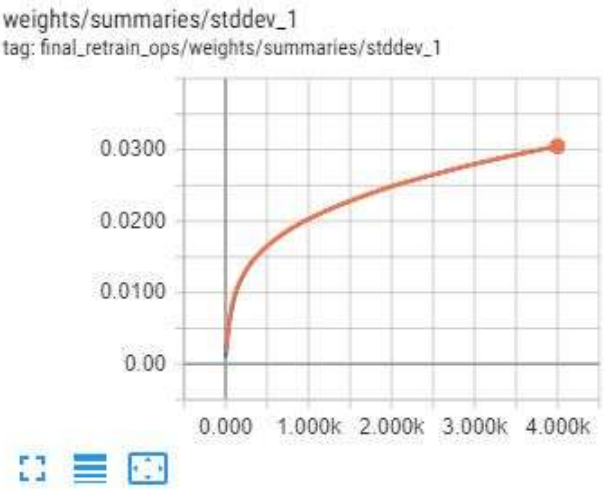
Appendix G: Re-Trained CNN Model Graphs




weights/summaries/max_1
tag: final_retrain_ops/weights/summaries/max_1
0.120
0.0800
0.0400
0.00
0.000
1.000k
2.000k
3.000k
4.000k


weights/summaries/mean_1
tag: final_retrain_ops/weights/summaries/mean_1
-3.341e-6
-3.341e-6
-3.342e-6
-3.342e-6
-3.343e-6
0.000
1.000k
2.000k
3.000k
4.000k


weights/summaries/min_1
tag: final_retrain_ops/weights/summaries/min_1
-0.0400
-0.0600
-0.0800
-0.100
-0.120
-0.140
0.000
1.000k
2.000k
3.000k
4.000k


weights/summaries/stddev_1
tag: final_retrain_ops/weights/summaries/stddev_1
0.0300
0.0200
0.0100
0.00
0.000
1.000k
2.000k
3.000k
4.000k